Character Based Interactive Storytelling for Role Playing Games

by

Cian Kearney, B.Sc(Hons)

Dissertation

Presented to the

University of Dublin, Trinity College

in fulfillment

of the requirements

for the Degree of

Master of Science in Computer Science

University of Dublin, Trinity College

August 2012

Declaration

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

Cian Kearney

August 31, 2012

Permission to Lend and/or Copy

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

Cian Kearney

August 31, 2012

Acknowledgments

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Mads Haahr, for his suggestions and advice throughout the project. I would also like to thank my family and friends for all of their support.

Cian Kearney

University of Dublin, Trinity College
August 2012

Character Based Interactive Storytelling for Role Playing Games

Cian Kearney
University of Dublin, Trinity College, 2012

Supervisor: Mads Haahr

ABSTRACT

One of the main advantages of computers or games consoles over other entertainment media is that they provide the user with a means to interact with the system. Video games as an entertainment medium provide an excellent opportunity to tell a truly interactive story in which the players actions and interactions with non-player characters affect the entire development and structure of the story. Stories in modern video games are mostly linear with occasional branching of story lines based around a couple of key choices presented to the player. Character based interactive storytelling is mainly concerned with creating believable non-player characters who act based on internal motivations and desires.

This dissertation introduces the concept of character based interactive storytelling into role playing games through non-player characters that dynamically interact with

each other and the player. This is a relatively new concept and was implemented through a personality model and social relationship model and Goal Oriented Action Planning.

This project has shown that Goal Oriented Action Planning can be successfully combined with character based interactive storytelling techniques to dynamically generate non-player characters behaviours within a role playing game environment. This dissertation demonstrates unequivocally that use of character based interactive storytelling techniques can be of great benefit in the future development of RPGs.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Video games have been commercially available since the early 1970's. Since then people have been trying to embellish games with stories. These stories were initially put into the game documentation and merely served to give context to the game and weren't actually a part of it. Nowadays nearly all games have a story which is interleaved with the gameplay. These stories are mostly linear with occasional branching of story lines based around a couple of key choices presented to the player. One of the main advantages of computers or games consoles over other entertainment media is that they provide the user with a means to interact with the system. Since the invention of video games there has been massive advancements in graphics and gameplay but the stories in these games have been lagging behind.

The stories in most modern Role Playing Games (RPGs) are predominantly linear with occasional branching of story lines based around a couple of key choices presented to the player. Some RPGs such as Mass Effect 2[6] execute this very well and give the player a real sense of control over the story's outcome. Video games as an entertainment medium provide an excellent opportunity to tell a truly interactive story in which the players actions and interactions with non-player characters (NPCs) affect the entire development and structure of the story. The recently released Mass Effect 3[7] even used "Interactive Storytelling" as one of its main selling points.

Interactive Digital Storytelling is a promising research area within Computer Sci-

ence, sociology, psychology and linguistics[30]. A typical interactive storytelling system would be based on autonomous virtual actors that generate the plot through their real-time interaction. The user should be allowed to interfere with the on-going action, thereby altering the plot as it unfolds[14]. An integral part of creating interactive stories is the compromise between providing a degree of freedom for players and a degree of coherence to deliver the main storyline. If the system provides too much freedom to players, it may fail to deliver the intention of an author; otherwise: players will lose their freedom if the system places a great deal of weight on delivering authors purpose which could cause them to lose their interest in the game[18].

1.2 Objectives

Traditional RPGs use linear scripted storylines with basic branching of the storyline based on key decisions made by the player. In most RPGs, the player can only interact with some of the NPCs and these NPCs are simply there to provide the player with information and scripted quests. They generally act in a predefined scripted manner and do not have the ability to interact with each other in a meaningful way that could alter the course of the game.

The objective of this dissertation is to introduce the concept of character based interactive storytelling into RPGs. This would create RPGs where the NPCs can interact dynamically with each other, and the player, to alter the course of the game. This is moving away from the traditional scripted nature of interactions in RPGs. This is a new concept in RPGs.

Within this objective, this dissertation must investigate the application of various character based interactive storytelling techniques and how they can be applied to an RPG. Some of these techniques include, defining NPCs with personalities and social relationships and how these could be used to guide an NPC's behaviour. It must investigate the use of a planners for guiding an NPCs actions as used in interactive storytelling.

The aim is to produce a working prototype demonstrating these techniques in action and the applicability of them in future RPG design.

This dissertation will review the state of the art techniques and research in interactive storytelling and planning algorithms related to this project. Some of the major

design decisions made during the course of this project will then be explained, as will the reasoning behind them. This will be followed by a detailed account of the development and implementation of the system. An evaluation of the system will then be presented followed by conclusions drawn from this evaluation. Finally, the authors own observations and possible directions for future work will be discussed.

1.3 Document Roadmap

This document is structured as follows:

Chapter 2 presents a "state of the art" literature review of research carried out in interactive storytelling and planning algorithms. It will also review the current state of interactive storytelling within the commercial games industry.

Chapter 3 reviews and explains the design decisions made during this project. It will also present the framework developed for implementation.

Chapter 4 discusses the details of implementing the system proposed in chapter 3. It will also provide some examples of the code.

Chapter 5 presents an evaluation of the system in terms of the objectives that were initially put forward.

Chapter 6 discusses what the project achieved and presents areas to be explored with future work.

Chapter 2

State of the Art

Currently there are a number of approaches being used in creating Interactive Storytelling applications. These approaches can generally be broken down into plot based and character based. Planning algorithms are also a widely researched topic within interactive storytelling. These can be used to decide how a character will achieve their goals. They can also be used to track progress and reroute the direction of the plot if necessary.

In the following sections, I will begin by giving an outline of some research that has taken place within academia regarding plot based, character based and hybrid approaches to interactive storytelling. I will then outline the use of planning algorithms and then discuss the current state of interactive storytelling in the commercial games industry. I will conclude by discussing some of the issues within the interactive storytelling domain.

2.1 Plot Based Interactive Storytelling

Plot based approaches to interactive storytelling focus on the structure of the plot and maintaining the coherence of the story. This leads to less opportunities for interaction with the game world and its inhabitants but ensures that the plot is coherent and engaging. Content/story production is a major bottleneck in the entertainment industry. In dramatic writing, stories are thought of as consisting of events that turn (change) values. A value is a property of an individual or relationship, such as trust, hope, etc

[17]. The most common approach to plot based interactive storytelling is to create an omnipotent Game Master or Drama Manager that monitors the state of the system and injects events or guides the player to progress the story along an author defined path.

Crawford[15] proposes a drama manager that Listens, Thinks and Speaks. The listening aspect of the drama manager monitors the overall state of the story world and recognises patterns of behaviour that differentiate drama from tedium. The simplest method of accomplishing this relies on overview variables which are used to get an overall sense of the world state. Once the drama manager has developed a coherent interpretation of the events, the thinking aspect must determine how the story will progress from that point. This can be done by matching the current story to a set of predefined 'dramatic templates' and the closest match then becomes a guide for the drama manager. Another approach is the use of 'story grammar' which is a set of rules governing the sequencing of events in a story. Vladimir Propp's[25] work on decomposing a story into its basic common elements is the most common example of 'story grammar'. The drama managers final task is translating its determinations into a form that changes the game world to evolve the story in a desired direction. A drama manager can do this in many ways, including environmental manipulation, injecting a new goal into an actor, shifting an actor's personality to encourage them to make a decision in a desired direction or simply the passage of time. All of these methods can be very effective in progressing a story.

Another example of plot based interactive storytelling is Fairclough et al[**?**] case based reasoning approach to create a story adventure online role playing game. Their system handles multiple users and directs the NPCs to perform for parallel user storylines, interweaving character roles in each story. The story is told through a 'narrative of actions' and automatically generated dialogue. An omnipotent director agent is used to control the story. The story director notes character attitudes towards the player which are changed every time the player interacts with them or hears about an interaction involving the player. The expert knowledge of the system is that form Propp's work[25] defined as a set of base cases. Each case represents one move of the story. They then use the k-nearest neighbour algorithm to find cases similar to the current state of the story world. This is then used to select the next case which is converted to a set of goals that are then assigned to the NPCs. This allows for the creation of

dynamic stories and quests based on the current state of the world.

2.2 Character Based Interactive Storytelling

One of the main aspects of character based interactive storytelling is character believability. In order for a character to be believable, they will need to respond to any likely situation in a manner that is consistent with their personality, as well as their position within social hierarchies[22]. This means that the character actions appear to be motivated by agents' internal beliefs and desires[26] (personality). The goal of character based interactive storytelling is create agents that appear reactive, goal-directed, emotional, intelligent and capable of using natural language. The agents are supposed to provide some signs of internal goals, reactivity, emotion, natural language ability and knowledge of agents (self and other) as well as those of the simulated physical world[30]. Character based approaches focus on modelling characters as believable autonomous agents with stories emerging from the players interaction with these agents. When this approach is adopted, it is easier to implement direct interaction with the characters, but harder to keep the story coherent[13]. Each actor has its own profile, which includes states, desires and preferences.

Sung Hyun Moon et al[21] introduce "properties" which are assigned to NPCs to represent their characteristics and status and their tendencies towards certain behaviours (personality). They discuss using an approach where the characters personality leads to them having different probabilities to perform certain actions. This might include a character taking a violent action against another if they have a personality which is geared towards violence whereas a kinder character may take a different approach. To produce the behaviour of the NPCs they adopt a "Constraint Based Narrative Structure". This is divided into constituent declaration and constraint declaration. The constituent declaration defines the building blocks for the story such as Actors, Action, Property, Stage and Props. In constraint declaration the author can declare various conditions which control the flow of the story. The role of constraints is to guide the story and to prevent the system from generating absurd stories. Below are two tables describing the constituent and constraint properties.

Table 2.1: Constituent Declaration

Name	Description
Actor	The subject or object in the story events.
Action	The behaviours or motions performed by an actor.
Property	The characteristics of the actors, actions, and props.
Props	Only the objects of actions.
Stage	The spaces where the story proceeds.

Table 2.2: Constraint Declaration

Name	Description
Ending	A designer can decide how the story ends by writing Ending constraint
Transaction	Transaction constraint is defined to when two events occur consequently.
Transition	If a certain event occurs, the values of specific properties are modified.
Induction	When the value of a certain property satisfies some conditions, a specific event occurs.
Must	If pre event occurs, post event must occur at any time before the end of story.
Ordering	Ordering constraint can be declared to adjust logical sequence of events in a story.
Stage Change	This is a constraint that shifts current stage.

To test their system they created a basic scenario consisting of five soldiers and a shopkeeper in a shop. The shopkeeper then suggests an item to each of the soldiers based on their perceived needs. Each time the simulation is executed the shopkeeper may suggest a different item if he perceives that the soldiers needs have changed.

Mosher et al[22] propose a framework for creating believable characters. The first part of this framework specifies character traits which provide a template for defining a characters personality. The aim for the template is to create consistent and coherent characters. They state that if the inherent properties of the agents and the world are not subject to drastic change without reasonable cause then consistency is maintained. Coherence is maintained if there is a logical explanation for events given the information available to the user. In a broader sense, they aim to model reality as closely as is practical. At the core of this template system is a set of traits. They use a method

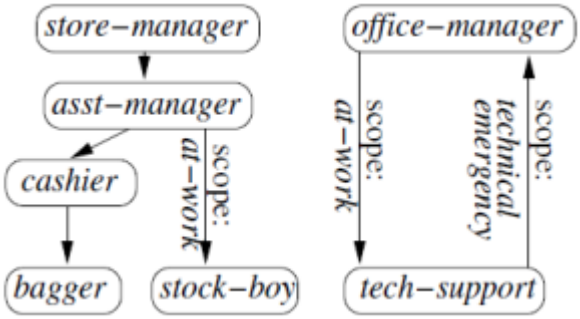
known as the Five-Factor model (FFM) which was initially a theory from psychology that was put into practice by Gordon Allport and H. S. Odbert in 1936. This model has since been refined numerous times to reach its current state which is a widely accepted set of factors and facets for describing personality. This model groups specific traits into one of five factors. Below is a table listing the Five-Factors and some associated traits.

Table 2.3: Five-Factor Model

Factors	Facets
Neuroticism	Anxiety, Angry Hostility, Depression, Self-Consciousness, Impulsiveness, Vulnerability
Extroversion	Warmth, Gregariousness, Assertiveness, Activity, Excitement-Seeking, Positive Emotions
Openness	Fantasy, Aesthetics, Feelings, Actions, Ideas, Values
Agreeableness	Trust, Straightforwardness, Altruism, Compliance, Modesty, Tender-mindedness
Conscientiousness	Competence, Order, Dutifulness, Achievement Striving, Self-Discipline, Deliberation

Along with the personality traits, the template specifies physical traits such as name, age, weight, sex, etc. It also allows for the specification of unique traits which include anything that doesn't fit into the other trait categories. These unique traits are generally skills (expert sailor, mediocre fisherman, etc.). They use stereotypes to help with the creation of their synthetic characters. Stereotypes provide default values for the character template based on their social role or background. These default values can then be tuned as the author desires. To further refine the personality they propose placing the characters into roles within a social hierarchy. This can greatly enhance believability as social standing plays a large role in human interaction. To handle the complexities of a social hierarchy they propose using a directed graph that includes scope indicators such as the one shown below.

Figure 2.1: Scope Indicators



Items such as relationships and emotions are not specified by this template. The system described is yet to be implemented but I believe it possesses great potential for quickly and easily defining believable characters for an interactive storytelling application.

Crawford[15] also proposes a method for creating a personality model for believable characters. Crawford's system is a very detailed approach at creating autonomous story characters. He states that the factors used in personality modelling for interactive storytelling should completely address all of the behaviours you the author wishes to invoke. The personality model should be kept as concise as possible to allow easier characters creation. Variables in the personality model should also be orthogonal. If two variables are orthogonal that means they have nothing in common, for example it would be pointless to use variables named 'Good-Humored' and 'Friendly' as they overlap quite a bit and it is hard to imagine a case where one factor would apply and the other one wouldn't. Every personality trait in the personality model should be created with an eye on what behaviour it might control. He proposes using five broad types of variable in a personality model; intrinsic, mood, volatility, accordance and relationship. Intrinsic traits are traits associated with any character such as greed, pride etc. Mood comprises the variable emotion states that people are subject to such as anger and joy. Volatility variables govern the readiness with which mood variables can change and accordance variables govern the readiness with which relationships change. Relationship variables model the feelings between two characters such as the affection or trust between them. He suggests that for every relationship variable there should be a corresponding accordance variable. For trust, the corresponding accordance variable

would be honesty. The choice of which variables appropriately model human personality for a simulation is an artistic choice and depends heavily on the type of story required. Crawford proposes a method of automatic relationship adjustment to model how characters are perceived by others within the story world. If one character punches another then a witness to this event might perceive that this character has a low virtue value as he is willing to punch someone. This perceived virtue is then stored and updated when necessary and will shape the interactions between the two characters. This technique relieves the story builders of the need to specify an actor's emotion reaction to every event they witness. The act of specifying a characters motivation automatically reveals character traits at work. Once the personality model is created, these variables need to be used to make decisions and choices. Crawford also proposes a model of tracking the history of each characters behaviour in order to prevent repetitive behaviour. He proposes using a 'History Book' to store this information. This 'History Book' is simply a temporal record of events in the story world. In computer programming past events can be stored as flags (Boolean values) but if this is used, there needs to be a flag for every possible eventuality. If Crawford's method is used then a simple search of the 'History Book' would replace this. This may take more processing time but it can provide extra information such as causality. He also suggest a method for implementing gossip and lies within the story world. This gossip and lies can have a dramatic impact on characters relationships and how they achieve their goals.

2.3 Hybrid Approaches

Most attempts at interactive storytelling combine aspects of both plot and character based approaches. Faade[20] is probably the most widely known attempt at this. In Faade you play the role of a friend of a couple experiencing marriage difficulties during a visit to their apartment. On a moment-by-moment basis, Faade is a character based simulation. It has a simulated virtual world with objects and behaviour-based autonomous agents. The simulation offers a high degree of freedom and local agency to the player. This is also an additional invisible agent called the drama manager. The drama manager continuously monitors the simulation and proactively adds and retracts behaviours and discourse contexts by which the autonomous agents operate.

These updates are organized into 'story beats'. A beat is the smallest unit of dramatic action that moves a story forward. These beats are a collection of behaviours tailored to a particular situation. Although it has some drawbacks such as the natural language understanding, the authorial effort required to create the characters and story beats, Faade is an excellent example of combining plot and character based approaches to create an interactive drama.

2.4 Automated Planning Algorithms

Interactive Storytelling has adopted AI planning techniques to control behaviours or actions of characters with narrative structure. There are numerous planning algorithms that can be used in interactive storytelling. Some of these algorithms are discussed and compared in [12]. The most popular structures are HTN(Hierarchical Task network)[14], HSP(Heuristic Planning) and POP (Partial Order Planning)[21]. Plot generation starts by inferring goals for the various characters (and for the story as a whole). Given this initial input, a planner can be used to insert events or actions for these characters into the story plot in order to fulfil the goals[13]. The Related Work section of [26] describes how previous attempts at interactive storytelling use planning algorithms to achieve their goals. One of the big problems with using planning algorithms in interactive storytelling is that several actors, or the user himself, might interfere with one agents plans, causing its planned actions to fail. Hence, the story can only carry forward if the agent has re-planning capabilities[5].

A number of games have implemented characters with goal directed decision-making capabilities. Goal-Oriented Action Planning (GOAP) is a decision-making architecture that takes the next step, and allows characters to decide not only what to do, but how to do it[24]. GOAP was developed by Jeff Orkin from Monolith Studios during the creation of the First Person Shooter First Encounter Assault Recon (F.E.A.R.). A planning system such as GOAP tells the A.I. what its goals and actions are, and lets the A.I. decide how to sequence actions to satisfy goals.

The GOAP algorithm is based on a modification of the STRIPS algorithm. STRIPS was developed at Stanford University in 1970, and the name is simply an acronym for the Stanford Research Institute Problem Solver. STRIPS consists of goals and actions, where goals describe some desired state of the world to we want to reach, and actions

are defined in terms of preconditions and effects. An action may only execute if all of its preconditions are met, and each action changes the state of the world in some way. [27, 23, 29].

The current state of the world is represented by a set of variables that collectively describe the world known as world state variables or symbols. Each symbol represents a single piece of information about the world, e.g. if an NPC wants to kill another NPC, it could have an isTargetDead symbol which would either be true or false. When looking for a solution to a goal, the planner must examine which world state symbols are unsatisfied between the agents current world state and the goal world state and look for actions that have effects that bring the two states closer to one another[28].

The A* algorithm is used to plan the correct sequence of actions that form an NPCs plan. The NPC then proceeds to step through and execute the plan, one action at a time, until there are no more steps left. GOAP differs from STRIPS in that a cost per action was added. This cost is used to guide the A* search toward the lowest cost sequence of actions to satisfy some goal. GOAP has been used in many games such as F.E.A.R.(2005), Fallout 3(2008) and Just Cause 2(2010)[1]. In all of these games, it has been used quite successfully for combat mechanics but stories and more detailed character interactions remain scripted.

2.5 Interactive Storytelling in the Games Industry

Interactive storytelling is still a relatively new idea within the games industry. Many games allow the user to directly influence its plot. However this is usually only at certain points with specific decisions that cause a noticeable predefined branching of the plot. Games such as Heavy Rain[2], Mass Effect 3[7] and Left 4 Dead[4] have some attempts at creating truly interactive stories. Heavy Rain is an interactive drama video game. The story is a dramatic psychological thriller featuring four playable characters involved with the mystery of a serial killer. The player interacts with the game by performing actions highlighted on screen related to motions of the controller or button presses, and in some cases, performing a series of quick time events during fast-paced action sequences. The player's decisions and actions during the game affect the narrative; the main characters can be killed, and certain actions may lead to different scenes and endings[3]. The game is a very good example of the plot based interactive

storytelling approach. Due to the fact that the game is focused on maintaining a coherent plot there is a lot less interaction than in a traditional video game with most of the key events being decided by quick time events or simple button presses. There are however between 18 and 22 possible endings and the player gets a sense that they are in control of each characters destiny. Mass Effect 3 is a 3rd person action Role-Playing Game (RPG). You play the role of Commander Shepard who has to save the earth from an alien invasion. The game had a much hyped interactive storytelling component. The game take into account all of the decisions that you made in Mass Effect 1 & 2 and feeds them into the narrative system. For example if any of your squad mates were killed in the final mission of previous games, they would remain dead which may have consequences for your current war effort. These decisions, along with any of the decisions made throughout the game decide whether or not the alien invasion could be stopped and which characters on your team live and die. This then leads to around 16 unique endings. The interactive storytelling approach used in Mass Effect 3 is also a very plot based approach. You are simply presented with decisions in some of the cut scenes and a single button press could dramatically alter the outcome which again appears to sacrifice interactivity for plot coherence. Left 4 Dead is a cooperative First-Person Shooter(FPS) video game. Set during the aftermath of an apocalyptic pandemic, the game pits its four protagonists against hordes of the infected. It features an AI called the "Director" which controls level pacing, difficulty and item placements, in an attempt to create a dramatic dynamic experience and increase replay value. The Director also creates mood and tension with emotional cues, such as visual effects, dynamic music, and character communication[5]. While this approach is not concerned with the plot of the story it was a very successful attempt at creating a dynamic game environment through its manipulation of dramatic elements such as pacing and enemy placement.

Chapter 3

Design

This chapter will discuss some of the major design decisions presented by this project. Firstly the game scenario will be described followed by a discussion on the choice between a plot based approach and a character based approach to interactive storytelling. The choice and use of a planning algorithm will then be explained. Finally the decision of traits for a personality model and social relationship model and some of the smaller design choices will be discussed.

3.1 Game Scenario

Typical Role Playing Games are usually made up for several small villages and towns. It is within these settlements that most of the interaction with NPCs takes place as they are usually concentrated in a small area. These settlements generally contain characters with a diverse set of backgrounds and occupations. It was decided to implement one of these small settlements for this project as all of the characters are within a small area and will have various different personalities, goals and motivations. The town also provides a closed and stable environment where all of the characters are free from any possible external influences. Most RPGs are set in a medieval, futuristic or post apocalyptic style fantasy world. A medieval style world and characters were selected for this project as it provides a large amount of different occupations and character types to choose from. Excluding the player, a total of ten other characters were chosen to populate the world.

Three 'good' farmers are designed with varying personalities although they are generally very friendly or have a high integrity. One antagonist farmer was added with high aggression, greed and low integrity. Therefore, this farmer is more inclined to steal from or attack other NPCs. A land baron was designed to be the ruler of the town. His main tasks are to buy and sell land and collect taxes from the other villagers. A blacksmith and a shopkeeper were designed whose main roles are simply to buy from and sell items to the player and the NPCs. A sheriff was devised to keep order in the village by investigating robberies and murders and arresting the suspected culprits. A hunter was designed to teach the player how to fight but would only do so if the player had a good reputation amongst the villagers. A town drunk was envisioned mainly to provide information to the player.

Each character owns some gold, seeds, land and product. It is mainly the sale and acquisition of these commodities that drives the games action. Characters start with a certain amount of these four commodities. Characters can acquire more gold by selling product, land or seeds to another NPC. Product is acquired through planting seeds and farming their land. The amount of product received from farming is dependent on how much land is owned by the NPC. Seeds can be purchased from other NPCs but are mainly sold by the shopkeepers. Land can also be purchased by characters in the game. The land baron owns most of the land and will be happy to sell a certain amount of it, however all of the farmers also have the ability to sell land. Characters may also steal seeds, product or gold if their personality dictates. Most of the goals and actions in the system relate to the sale or acquisition of these commodities and as such, it is these commodities that drive the games action.

3.2 Interactive Storytelling Method

As was discussed in Chapter 2, the implementation of a plot based approach or a character based approach to interactive storytelling results in very different user experiences. Plot based approaches lead to a very coherent story which allows the author to portray their intent quite clearly but allows for much less user interaction. Character based approaches on the other hand result in a much more interactive experience for the end user but may result in a loss of coherence and may fail to deliver the intent of the author.

A role playing game is a game in which the player controls the actions of a protagonist as this character lives immersed in a fictional world[9]. RPGs typically rely on a highly developed story and setting and also attempt to offer more complex and dynamic character interaction than what is found in other video game genres. This usually involves additional focus on the artificial intelligence and scripted behaviour of computer-controlled non-player characters[11, 16].

A plot based approach may result in a great story but if there is not enough user input and interaction, the player may become bored. Therefore it was decided to use a character based approach when designing the system as it fits best with the RPG genre, in that it allows for the most interactive user experience and provides more dynamic NPC behaviour. This approach warrants the creation of more realistic and believable NPCs. They should have the ability to act based on their own personal goals and desires. They must also be able to react dynamically to situations as they arise. One of the key aspects of character based interactive storytelling is that the NPCs must be able to handle interaction with the player. This may include talking to the player, asking for help or giving the player tasks to complete. These NPCs should dynamically interact with the player and other NPCs in the world. Creating NPCs that can react and adapt their behaviours in response to external factors such as the players actions or changes in the game world, will increase the players level of immersion with the game itself, as they are more likely to identify with the characters motivations and troubles[11], thus desiring to continue playing the game and getting more involved with those characters. It is from this interaction, the story should emerge.

3.3 Planner Design

Once a character based approach was chosen, a method of controlling the NPC behaviour was required. The traditional method of using a Finite State Machine(FSM) with scripted behaviours was initially considered. This method involves creating a FSM with all possible states defined and transitions between states. One complaint that was found from AI developers was that FSM becomes especially difficult to manage when the number of states increases[19]. Another drawback to this method is that a FSM can only move from state to state and has no real means of planning in advance what state it will go to next. It was for these reasons that this method was rejected

in favour of using an automated planning algorithm. After thorough research detailed in Chapter 2, Goal Oriented Action Planning (GOAP) was chosen. One of the main reasons it was chosen was that it mimics human reasoning. GOAP however comes with its own set of requirements.

3.3.1 World State

In order for the planner to search through the available actions, a method of representing the current state of the world is needed. This representation should allow for comparison in a way that lets it easily apply the preconditions and effects of actions, and recognize when it has reached the goal state. All of Orkins articles describing GOAP indicate that within a planning system each GOAP agent should maintain a world state which is a collection of world state variables or symbols that relate to the virtual world. Therefore, in order for the planner to function correctly, it was necessary to represent what was happening in the world symbolically. Each agent maintains a world state which is basically a fixed-size array of world state symbols. A world state structure was created to store these key pieces of information about the game world. The world state structure was implemented as an enumerated property variable and a value which is either a Boolean or integer value. An array of these world states is maintained by each NPC in order to condense masses of more complex variables into smaller and easier to manage properties which can be understood by the planner. These arrays are updated on a regular basis in order for the planner to have the most up to date information about the current state of the world.

```
struct WORLDSTATE{
worldProperty prop;
union VALUE
bool bValue;
int iValue;
value;
};
_worldState.prop = targetDead;
_worldState.value.bValue = false;
```

3.3.2 Goal and Action Sets

A global set of goals and actions needed to be designed for the NPCs. An NPC could then choose a goal, based on their needs and desires and then the planner would search through the available actions and find the most appropriate plan to achieve the goal.

Different agents in the system can have different capabilities. Each type of agent has its own goal and action set which define what goals and actions the planner can use when planning. Rather than hard-code this into the application, it was decided to keep this data separate from the C++ code. XML was chosen as the means to represent what actions and goals an agent could have. When an agent is first created, its own XML file is read in by the system and this then determines which actions and goals are allocated to it. One of the main reasons for choosing this method is to eradicate the need to recompile the code when actions or goals are added or removed from an agent. The benefit of using this system is that when agents with different goal and action sets are being created, the system simply reads in the XML file for that agent. Below is the goal and action set of Farmer 1 in its XML format. Appendix 2 contains sample XML file.

3.3.3 Actions

The design of the structure for the GOAP actions was influenced by the requirements specification document set-out by the working group on GOAP [7] along with other suggestions from Orkins various articles on the subject. Orkin states that GOAP actions should have the same general characteristics[24] so an abstract action class was designed that implements the basic functionality required for all actions. All the subsequent actions created, build on-top of this base class and extend its functionality. Each action is made up of a set of preconditions that must be satisfied before it can be executed and a set of effects that it has on the world once its execution is complete. These preconditions and effects are each represented by world state variables.

3.3.4 Goals

Goals are just as important as actions in a GOAP system as without them the planner has nothing to work towards. Orkin states that, in a similar way to actions, every goal should have the same basic structure[24]. Therefore an abstract goal class was also designed that implements the basic functionality required for all goals. All the subsequent goals created build on-top of this base class and extend its functionality. Each defined goal has a set of effects on the world. In a similar method to the actions, these effects are represented by world state variables. This is to allow the search algorithm of the planner to easily compare the effects of actions and goals. For a goal to be completed, the planner must find actions that satisfy these effects. World state symbol checks are used during planning to determine if a goal has become satisfied.

3.3.5 Goal Selection

In a GOAP system, each goal is assigned a relevancy variable. A goals relevancy is determined by the world state at the time and calculated within the goal object. The system proposed in this report takes a slightly different approach. When the NPC requests a new goal from the planner, it searches through the available goals and thresholds them based on predefined world states and personality traits. This allows the planner to choose goals not only based on necessity but also based on the personality of the NPC that requested it. The planner discards the goals that do not qualify and keeps track of the goals that pass the threshold. If more than one goal passes the threshold, it treats them as if they are equally important and randomly chooses between them. When a goal is selected as an agents current goal, the planner is instructed to create a plan. The planner is requested to run the search algorithm and find a series of actions that satisfy the goal.

3.3.6 Search Algorithm

In a traditional GOAP system, each action is assigned a cost. The planner then uses the A* search algorithm to find the best path from the current world state to the goal world state. A variation on this method was designed for this project to allow the NPCs to act based on their personalities. Instead of each action being assigned a

cost, each action contains a personality threshold. An action can only be considered for the plan if these threshold conditions are met. Once a plan is found by the search algorithm, it is returned to the NPC that requested it, ready for execution.

3.4 Personality Model

A personality model was designed to guide the actions and goals of the NPCs in the game world. A personality model is a data structure that contains the information needed to define a character. Crawford describes five factors in developing a personality model[15]. These factors are:

Completeness

The traits that make up the model must completely address all of the behaviours that can be evoked in the game world.

Conciseness

A personality model should be as concise as possible while still being able to cover the range of behaviours in the game world. If the personality model becomes too large, it can become difficult to decide which traits should be applied in a given situation.

Orthogonality

The variables in a personality model should not overlap. If traits overlap it can again lead to difficulty in deciding which one to apply. For two variables to be orthogonal, they should have nothing in common.

Tied to Behaviour

Personality models are built to determine an NPCs behaviour in a given situation. Every personality trait should be created with the goal of controlling a characters behaviour.

Overspecificity

Special case personality traits should not be created for individual behaviours. These behaviours should be controlled by more fundamental variables instead.

The personality model for this system was designed based loosely on Crawford's model discussed in the state of the art while taking the five factors discussed above into account. The aim was to keep it as concise as possible while at the same time being diverse enough to describe a character thoroughly. Each personality trait was designed to have a specific purpose and is stored as an integer with its value capped

between 0 and 10.

Friendliness(openness)

Friendliness describes how receptive the NPC is to being approached by the player and other NPCs. This trait is especially important if the NPC is interrupted while in the middle of an action.

Greed

Greed is a measure of how satisfied the NPC is with what they have got. A character with high greed will constantly try to get more land and gold whereas a character with low greed will be more likely to be content and simply work their land.

Aggressiveness

Aggressiveness defines how likely the NPC is to choose an aggressive goal or action such as attacking another NPC.

Activeness

Activeness describes how active or lazy an NPC is. A very active NPC will request goals and carry out all of their own tasks while a lazy NPC will try and ask the player or another NPC to carry out tasks and complete goals for them.

Easy Going

Easy going was included in the original design but its functionality was removed as there was too much of an overlap with the friendliness trait.

Emotional Instability

Emotional Instability essentially defines how much a positive or negative decision affects the NPCs happiness and social relationships with the player and other NPCs. This value scales the effect, a reaction to an event, has on the NPC's mood and friendships.

Integrity

Integrity is a measure of how loyal a character is. It also describes how likely they are to undertake the more immoral actions such as attacking or stealing to achieve their goals

Happiness

Happiness represents the general mood of the NPC. It affects the reaction of an NPC when they are interrupted during an action and also affects the type of actions and goals that the character will undertake. For example it would be unlikely that a very happy character would choose to kill another NPC.

3.5 Social Relationship Model

In a small town such as the one that is being created in this system, every character will know and have an opinion of every other character in the town. These opinions will generally affect how characters interact with each other. A social relationship model is required to describe these opinions in a way that can easily be understood by the system. The social relationship model defines and models the relationships between the NPCs in the game world. It was initially designed to be complex, consisting of attributes such as perceived aggressiveness, perceived openness, perceived loyalty and perceived activeness as well as a simple friendship variable. During the implementation it was discovered that, due to the basic nature of the system, most of the opinion traits were not necessary and were therefore reduced to simply using the friendship variable. The NPCs also keep track of their friendship with the player. This is then used by the NPCs to decide which jobs to give the player.

The relationships between NPCs are affected by most actions in the world. For example if the player rejects a quest given to them by an NPC, the friendship between the player and this NPC will be reduced. Conversely if the quest is accepted and completed, the friendship will increase. These adjustments of friendship will affect the likelihood of the player receiving quests from this NPC in future. Similarly, if an NPC tries to buy an item from another NPC and the request is rejected, the friendship between the two will be reduced. This will then increase the possibility of these NPCs taking hostile action against each other.

3.6 XML Files

As was discussed earlier, the goal and action sets for the NPCs are read in through an XML file. The system was also designed so that each NPCs personality, initial friendships and initial possessions are specified in an XML file. This XML file is then read into the system when the NPCs are being created. This was a design choice that removed the need to hard code all of these values into the system. This also allows for easier testing of the system as it removes the need to recompile after changes. Furthermore it provides the possibility of procedurally generating characters in future iterations.

Chapter 4

Implementation

The previous chapter described most of the main design challenges faced during this project and this chapter will describe the architecture and functionality of the resulting system. Firstly the choice of development tools and development methodology will be discussed. The details of player interaction, the management of the state of the game world and the event messaging system will then be examined. This will be followed by an explanation of the goal and action sets and the planner implementation. NPC decision making and automatic relationship adjustment will then be discussed and finally the XML file format will be reviewed.

4.1 Development Tools

The system is written in C++ using the OGRE3D rendering engine for much of the visualisations of this project. OGRE is a scene-oriented, flexible 3D engine written in C++ designed to make it easier and more intuitive for developers to produce applications utilising hardware-accelerated 3D graphics[8]. Using OGRE and its supplied models allowed for the rapid development of a 3D game environment, with basic collision detection viewed from a basic 3rd person camera behind the player. It also provided a basic GUI system that was implemented and used to present in game quests and NPC interactions to the player

A basic debug screen was set up within the OGRE render window but it quickly became apparent that this was not sufficient and therefore a console window was attached

to supply more detailed debug information.

An XML file parser called tinyxml2 was used to parse the NPC setup files[10]. This XML parser was invaluable to the development process as it allowed the NPCs personalities, relationships and goal and action sets to be modified quickly and easily without the need to recompile the code.

4.2 Development Methodology

An agile approach loosely based on SCRUM was adopted for this project. The time available for the project was split between research and dedicated development time. The entire development of the system was broken down into small individual tasks and ranked in order of importance. These tasks were maintained on a task list spreadsheet with a status indicator, priority and the estimated time they would take to complete. The development time was then split into weekly SCRUM sprints. At the start of each sprint tasks were chosen based on their importance to the overall system. The goal was to have a new, working version of the system that could be tested at the end of each sprint. After each sprint, any new features were added and prioritised on the task list and a new set of tasks for the next week was selected.

4.3 3D Environment

The first part of the implementation phase of this project involved creating a basic 3D world environment within which the action would take place. This environment was created using OGRE3D. A ground plane was created and divided into a 48 chunks in a regular 6x8 grid. Each section in the grid is a chunk of land that that can be bought, sold and owned by the NPCs in the world. The next step was to add a player character into this world. A simple 3rd person camera was set up behind the player character in order to view the action. The player character was then given the ability to move around the scene in response to keyboard and mouse input. A basic NPC model was then added to the world and the method of player interaction was implemented. This involved creating a simple GUI that could display text to the player which would be used by the NPCs for presenting quests and information to the player. A yes and no

decision button was also included to allow the player to accept or reject the quests that were presented to them. As with a new system like this, debugging plays a large role in development and therefore a console window was also attached to the application to allow for the easy output of variables and messages.

4.4 Player Interaction

The player interacts with the system by 'talking' to the NPCs in the scene. The spacebar is used to initiate interactions between the player and the NPCs. Once the spacebar is pressed and the player is close enough to an NPC a message is sent to that NPC requesting an interaction. When the NPC receives this request they can respond in several different ways depending on whether they are in the middle of an action or idling. When the NPC is in an idle state and their personality is lazy, unhappy or friendly, they will search for a goal and present it to the player. If their personality does not fall into one of these categories, they will simply give you information. When the NPC is engaged in an action a mood score for that NPC is calculated. This mood score takes every personality trait and friendship variable into account and collates them into a single value. This mood score then represents how open the NPC is to being interrupted. It is calculated by checking their personality traits and opinion of the player against a maximum and minimum threshold. A value of 1 is added or subtracted if the personality trait or opinion is well above or below the maximum and minimum thresholds respectively. A value of 0.5 is added or subtracted if the personality trait or opinion is only slightly above or below the thresholds. Nothing is added or subtracted if the personality trait or opinion falls in between the thresholds. Once the mood score is calculated, if it is in between 0.5 and -0.5, the NPC will not change their goal but will not mind being interrupted and may give you information. If the mood score is greater than 0.5, the NPC will be happy to interact with the player and will search for a goal to be given to the player. Finally, if the mood score is less than -0.5, the NPC will be angry that they have been interrupted and their opinion of the player will decrease. The NPCs response to the interaction is then presented to the player via the GUI and any relationship adjustments resulting from the interaction are displayed in the console window.

Figure 4.1: Player Interaction

4.5 Messaging System

An event messaging system was created to handle and notify NPCs about events in the world. A message data structure was created to standardise the format of messages within the system. This message data structure is made up of four components; a sender, the recipient, the contents which is an enumerated value and an optional location. struct MESSAGE{

int sender;

int receiver;

int contents;

int location;

};

A message manager was also created to handle the delivery of these messages to the appropriate NPCs. The message manager consists of two vector containers, a

message vector and a delayed message vector. The message vector stores messages to be transmitted during the current update and the delayed message vector stores messages to be transmitted during the next update. During each update of the system, each NPC receives their messages and can then react accordingly. If a reply is needed, the NPC creates the reply message and adds it to the delayed message vector for transmission during the next update. This delayed message structure was necessary to avoid the constant transmission of messages and reply's during the same update as this could cause a bottleneck in the system and affect performance.

Each NPC maintains a pointer to the Message Manager. When an event occurs in the game world, it usually affects more than just the character who created the event. If an NPC creates an event that other NPCs need to be notified about or needs to communicate with another NPC in any way, that NPC creates a message and adds it to the message vector in the Message Manager. It is through this messaging system that the NPCs communicate with each other. For example, when an NPC attempts to buy land from a second NPC, a requestBuyLand message such as the following is added to the messaging system.

```
message.sender = ID;
message.receiver = targetID;
message.contents = BuyLand;
```

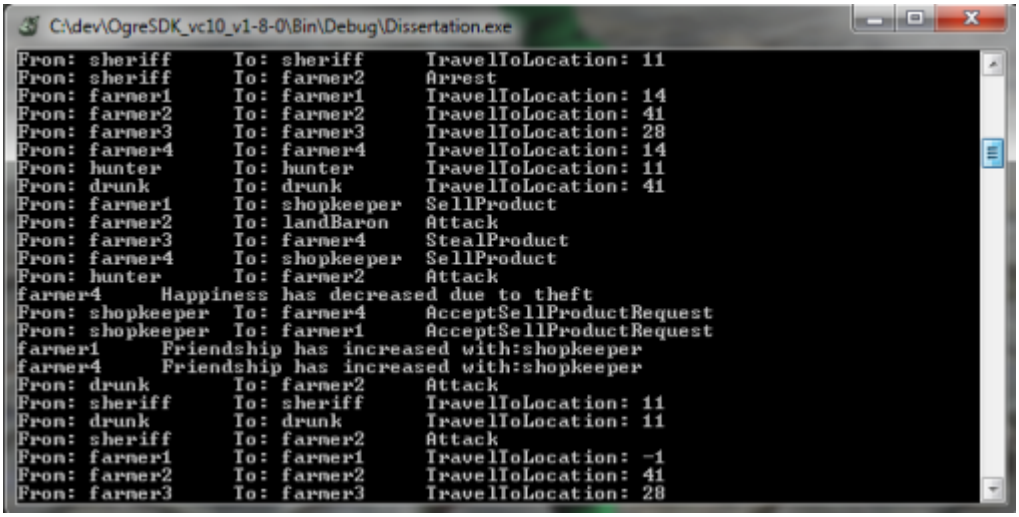
This message is then passed on to the recipient and dealt with appropriately. Once the message is received, the recipient can then make their decision on whether or not to sell the land. When a decision is made, the NPC creates a message to be sent back informing the first NPC of their decision. This message would be added to the delayed message vector for transmission during the subsequent update.

```
reply.sender = _ID;
reply.receiver = message.sender;
reply.contents = AcceptBuyLandRequest;
```

This message is received by the appropriate NPC and the automatic relationship adjustment reacts and items owned are adjusted depending on the contents of the

message. Every message that is sent by the NPCs in the system is outputted to the console window. This allows the developer to quickly and easily get an overview of exactly what is happening in the system.

Figure 4.2: Console Debug Window



4.6 Goals

A goal is something that the NPC wants to achieve within the game world. As was mentioned in Chapter 3, every goal should have the same basic structure. Therefore, a Goal abstract base class was created with all of the specific individual goals implementing this. Each goal contains a set of effects stored as world state variables in an array. These variables represent the effect that completing the goal will have on the current world state. An example of the desired world state for the GoalKillEnemy goal is shown below.

```
_numEffects = 1;
_effect[0].prop = targetDead;
_effect[0].value.bValue = true;
```

4.7 Actions

An action is something that an NPC can do in the game world that will have an effect on one or more of the world state variables. Again as it was mentioned in Chapter 3, every action should have the same basic structure. An Action abstract base class was created with all of the specific individual actions implementing this. Each action contains a set of effects stored as an array of world state variables. These effects represent the change that executing the action will have on the current world state. An action also contains a set of preconditions that must be met before the action can be executed. These preconditions are also stored as an array of world state variables. An example of preconditions and effects for the ActionAttack are shown below.

```
_numEffects = 1;
_effect[0].prop = targetDead;
_effect[0].value.bValue = true;
_numPreconditions = 2;
_preconditions[0].prop = atLocation;
_preconditions[0].value.bValue = true;
_preconditions[1].prop = hasWeapon;
_preconditions[1].value.bValue = true;
```

In addition to the effects and preconditions, each action has a personality threshold. This threshold is a set of values for each personality that an NPCs personality must exceed in order to be able to execute the action. It is used as the heuristic for the planner's search algorithm. If these threshold conditions are not met then the action cannot be selected. The following is an example of the personality threshold for the ActionAttack

```
_personalityThreshold.agressivness = 8;
_personalityThreshold.greed = -1;
_personalityThreshold.friendliness = -1;
_personalityThreshold.activeness = 7;
_personalityThreshold.integrity = 8;
```

This threshold means that the NPCs aggressiveness, activeness and integrity must be greater than these values in order for the action to be selected. The value of -1 means that the threshold need not be considered for these personality traits.

When an action is selected for a plan, its initialise function is called. This function is responsible for deciding the target NPC for the action, i.e. who the NPC will try to buy or steal from, sell to or attack. All of this information will be vital for when the action is actually executed by the NPC. This is done by searching through the world state of each NPC and deciding which one is most appropriate for the current goal. Once a plan is found, the NPC must then execute it. Each action has an execute function. This function is called by the NPC during the plan execution. It is responsible for sending any appropriate messages that are required by the NPC to begin and complete the action.

4.8 Planner

The planner is one of the key aspects of the implementation. The planner maintains a list of goals that the NPC has in the world and a list of actions that it can execute to achieve these goals. It also maintains a list of world state variables for each NPC. When an NPC requires a new goal, it requests one from the planner. The planner then searches through all of the goals available to the NPC and determines which ones are needed based on the NPCs current world state. If more than one possible goal is found, the planner treats them as equally important and simply chooses randomly from them.

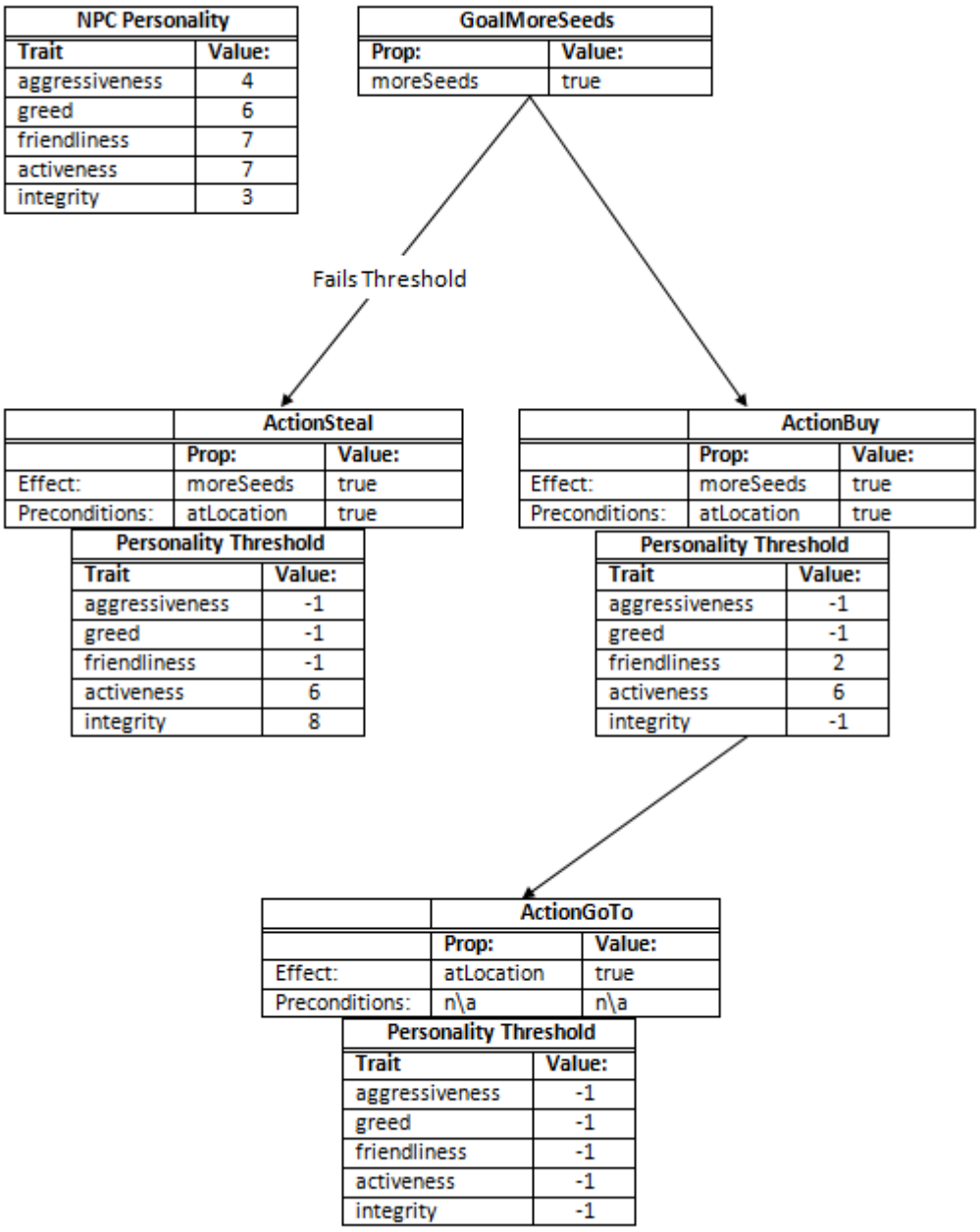
The goal is then passed to the findPlan() method. This method then performs a search similar to the A* search algorithm on all of the possible actions. The search begins by adding the goal's effect to a vector container called the Open List. The algorithm then searches through the effects that each action has on the world state and stores the actions whose effects match the effect in the Open List. Unlike the A* algorithm in which each node has a cost as a heuristic, each action has a personality threshold. The NPCs personality is then compared to the threshold of all the found actions and the most appropriate action is established.

The preconditions of this action are then added to the Open List. The effects in the Open List are then checked against the current world state and those that are already satisfied are removed. The algorithm then returns to the start and searches for actions

whose effects match those in the Open List until a complete plan is found.

This plan consists of a list of actions that once executed will change the current world state to that of the goal world state. The plan is stored in a vector container and returned to the NPC. Once the plan is returned to the NPC, it is activated and the first action is executed. Once this action is completed it is removed from the plan and the next action is executed. Once there are no more actions left in the list, the plan is finished executing and it is discarded.

Figure 4.3: Planning Sequence



4.9 NPC Decision Making

Many decisions need to be made by NPCs within the game world. These decisions range from selection of a goal to whether or not to sell an item. Each decision in the game world is made based on the NPCs personality and current world state. The following example will show that each decision is made based on a variety of factors from personality to necessity. An NPC may sell land based on their personality if they are not greedy, not very active and friendly with the NPC who wishes to purchase it. They may also sell land if they are greedy and lazy. The decision to sell land may become a necessity and be forced upon the NPC if they are low on gold and do not have enough seeds to farm their land.

4.10 Automatic Relationship Adjustment

A method of automatic relationship adjustment was implemented and used in this system. Every time an NPC or the player makes a decision in the world, it provokes an emotional reaction from the NPCs which affects their opinions of other characters. If an NPC attempts to buy an item from a second NPC and the deal is rejected, the first NPC's happiness will be reduced and the friendship between the two NPCs will also be reduced. Conversely if the deal is accepted, the first NPC will be happier and the friendship between the two will be increased. Automatic relationship adjustment also comes into play when a character has an item stolen from them. The happiness of this character is automatically reduced. The system also models a sense of achievement for the NPCs. This is also implemented with automatic relationship adjustment as once an NPC complete a goal, their happiness is slightly increased.

One of the most important scenarios where automatic relationship adjustment applies is when the player is dealing with an NPC. If the NPC is interrupted during a time when they are busy(executing an action) and they are not open to being approached(low mood score), that NPCs friendship with the player will be reduced. However if the player completes a goal for an NPC, the friendship between that NPC and the player will be increased. The emotionalInstability variable is responsible for scaling the severity of these reactions. This means that if an NPC is highly emotionally unstable, they will have a more severe reaction to an event than a more emotionally

stable character. This automatic adjustment method relieves the designer of having to specify emotional reactions to every possible situation. This also helps to expose how emotional the NPC is to the player.

4.11 XML Files

XML files are created for each NPC in the game world. These XML files define the characters personality, attributes, items owned, start location and land owned. It also defines the goals that the character has in the world and all of the actions they can take to achieve their goals. Finally it specifies the relationships between the NPCs. The benefit of using this system is that if agents with different goal and action sets need to be created then the program can simply read it in through the XML file for that NPC. The external XML files are processed using an open source plugin written in C++ called tinyxml2 that can read XML files and provides functions to parse them quickly and easily. During the initialisation of each NPC in the game world, the corresponding XML file is parsed. The values defined in the XML files are stored in the corresponding variables for each NPC.

Chapter 5

Evaluation

This chapter will present an evaluation of the system that was created and assess the extent to which the objectives were achieved.

The primary objective was to introduce character based interactive storytelling techniques to RPGs through NPCs that dynamically interact with each other and the player. This was implemented through the personality model and social relationship model and GOAP.

5.1 Personality Model & Social Relationship Model

To achieve the objectives a personality model was defined with six personality traits and a general happiness variable. This model influenced the interactions amongst the NPCs and with the player. It also guided their selection of goals and actions. These variables were defined in an XML file. This allowed the variable to be easily altered to evaluate the effect that changing the variables had on the course of the simulation.

This personality model was extremely successful in influencing actions, interactions and decisions. It was demonstrated that changing some personality trait e.g. integrity, that the entire flow of the game was drastically altered. This demonstrates unequivocally that use of character based interactive storytelling techniques can be of great benefit in the future development of RPGs. This is due to the fact that the game can be totally changed simply by altering a single personality variable.

5.2 GOAP

To achieve the objective of dynamic interaction GOAP was chosen to plan the NPC actions. This followed from extensive research into automated planning algorithms, including Hierarchical Task Networks, Heuristic Planning and Partial Order Planning. These are amongst the most widely used planners within interactive storytelling.

A global set of goals and actions was defined. The goals and actions for each character were read in through an XML file. This allowed goals and actions to be added or removed with ease. The planner finds an appropriate goal for the NPC and the GOAP search algorithm finds the best set of actions to achieve this goal guided by the NPCs personality.

The GOAP system was an excellent choice for producing NPC behaviour in RPGs. A goal-directed character displays some measure of intelligence by autonomously deciding to activate the behaviour that will satisfy the most relevant goal at any instance[24]. The goals that the system presents to the player conform nicely to the classic RPG quest types. Fetch, kill and deliver quests are all types of jobs that can be presented to the player by the NPCs. These quest are generated dynamically as opposed to the traditional scripted flow. This demonstrates more human like reasoning. This clearly proves its suitability for use in RPGs and should be considered as a foundation for future games.

5.3 XML

Separating the character traits and goal and action sets from the main program into an external XML file allowed for easy character customisation. This demonstrated its usefulness during testing and would also allow for quick and easy in game customisation of characters by the players or developers.

5.4 Implementation Considerations

One of the main drawbacks of the system is that actions are constantly being executed in the game without the players knowledge. This could lead easily lead to the player being confused by NPCs moods and discovering dead characters with no explanation

of what happened. The evaluation of the system required a console window interface to accurately keep track of what was happening in the game world. This is a problem that could easily be overcome in future iterations of the system by implementing a gossip system which is discussed in detail in Chapter 6.2.

Chapter 6

Conclusion

From the evaluation, it's clear that interactive storytelling has massive potential for the future of RPGs. As was described in Chapter 5, changing a single personality trait for one character produces a totally different game. This makes it an incredibly powerful tool for game developers.

This project has shown that GOAP can be successfully combined with character based interactive storytelling techniques to dynamically generate NPC behaviours within a RPG style environment. This appears to be a significant finding as GOAP has mainly been used for combat mechanics in first person shooter games. This is one of the first implementations of GOAP to plan character actions in an RPG. GOAP also shows massive potential for replacing traditional scripted behaviour in RPGs. It provides a much more dynamic and interactive environment for the player.

The system works as intended with the NPCs choosing actions and giving quests to the player based on their individual goals, personalities and social relationships.

The choice of using XML files to read in NPC personality traits, social relationships and goal and action sets was one of most crucial parts of the project. It removed the need to hardcode values for each NPC into the system which provides excellent flexibility for developers to add and alter characters and action and goal sets.

6.1 Drawbacks

The systems shows great potential for use in Role Playing Games although would not be suitable for generating the main storyline in its current state. Small towns are a common feature in RPGs. The current system could be run alongside the main story in order to generate these towns within the game world. This would lead to a more thriving town with constant action taking place. It would also produce endless side quests for the player.

6.2 Future Work

Some of the actions that were designed for the system have not yet been implemented fully. The land barons AskNicely and Intimidate actions have not been implemented and I believe this would add more depth to the system. The AskFriend actions have not been fully implemented either. They were initially set up with preconditions and effects and personality thresholds but the initialise and execution functions were not implemented and therefore the actions had to be omitted from the final demo. Full implementing these actions would allow for a more thorough evaluation of the systems storytelling capabilities.

A gossip system is another area for future work. A gossip system would allow the NPCs to inform their friends of actions that they have witnessed. This would allow NPCs to adjust their opinions of the player or other NPCs based on what they are told by friends. A massive benefit of implementing this system is would be that the player becomes more informed about what is happening in the game world. NPCs that are friendly with the player would be able to inform him about actions that have taken place in the world. This would help explain to the player why an NPC has suddenly become very unhappy or who killed an NPC.

The system designed and implemented is reasonably basic and acts as a proof of concept for using interactive storytelling techniques and GOAP in RPGs. The next logical step would be to expand system by adding more complex goals and actions. More NPCs with different occupations could also be created to add move variation into the game world.

Another area worth further investigation would be the procedural generation of

NPCs. Due to the fact that the personalities, social relationships, attributes, items and goal and action sets do not need to be hardcoded into the system, this presents the possibility for procedurally generating NPCs. This could be taken a step further by defining a range of personality values that describe simple stereotypes of "Good" or "Bad" characters. This could lead to the creation of a very diverse set of NPCs populating a town with minimal effort. This could save designers a great deal of time during the creation of unimportant background characters or if the system was not being used for progressing the main storyline and was running concurrently to give the impression of a thriving town.

Appendix

Goal and ActionSets

Goals

Goal: GoalKillEnemy
Effects: targetDead = true

Goal: GoalMoreGold
Effects: moreGold = true

Goal: GoalMoreLand
Effects: moreLand = true

Goal: GoalMoreProduct
Effects: moreProduct = true

Goal: GoalMoreSeeds
Effects: moreSeeds = true

Actions

Action: ActionArrest

Preconditions: atLocation = true
Effects: moreGold = true
Personality Threshold:
Aggressiveness: 5
Greed: -1
Friendliness: -1
Activeness: 7
Integrity: 7

Action: ActionSendFriendToArrest
Preconditions: atLocation = true
Effects: moreGold = true
Personality Threshold:
Aggressiveness: 5
Greed: -1
Friendliness: -1
Activeness: -1
Integrity: 7

Action: ActionAttack
Preconditions: atLocation = true, hasWeapon = true
Effects: targetDead = true
Personality Threshold:
Aggressiveness: 8
Greed: -1
Friendliness: -1
Activeness: 7
Integrity: 8

Action: ActionSendFriendToAttack

Preconditions: atLocation = true, hasWeapon = true
Effects: targetDead = true
Personality Threshold:
Aggressiveness: 8
Greed: -1
Friendliness: -1
Activeness: -1
Integrity: 8

Action: ActionBuy
Preconditions: atLocation = true, enoughGold = true
Effects: moreLand = true / moreSeeds = true
Personality Threshold:
Aggressiveness: -1
Greed: -1
Friendliness: -1
Activeness: 7
Integrity: -1

Action: ActionSendFriendToBuy
Preconditions: atLocation = true, enoughGold = true
Effects: moreLand = true / moreSeeds = true
Personality Threshold:
Aggressiveness: -1
Greed: -1
Friendliness: -1
Activeness: -1
Integrity: -1

Action: ActionGoTo
Preconditions: none
Effects: atLocation = true
Personality Threshold:
None

Action: ActionSell
Preconditions: atLocation = true, enoughSeeds = true / enoughLand = ture/ enough-Product = true
Effects: moreGold = true
Personality Threshold:
Aggressiveness: -1
Greed: -1
Friendliness: -1
Activeness: 7
Integrity: -1

Action: ActionSendFriendToSell
Preconditions: atLocation = true, enoughSeeds = true / enoughLand = ture/ enough-Product = true
Effects: moreGold = true
Personality Threshold:
Aggressiveness: -1
Greed: -1
Friendliness: -1
Activeness: -1
Integrity: -1

Action: ActionSteal
Preconditions: atLocation = true
Effects: moreSeeds = true / moreGold = true / moreProduct = true
Personality Threshold:
Aggressiveness: -1
Greed: 0
Friendliness: -1
Activeness: 7
Integrity: 8

Action: ActionSendFriendToSteal
Preconditions: atLocation = true
Effects: moreSeeds = true / moreGold = true / moreProduct = true
Personality Threshold:
Aggressiveness: -1
Greed: 0
Friendliness: -1
Activeness: -1
Integrity: 8

Action: ActionWork
Preconditions: atLocation = true
Effects: moreProduct = true
Personality Threshold:
Aggressiveness: -1
Greed: -1
Friendliness: -1
Activeness: 7
Integrity: -1

Appendix

<AGENT>
<PERSONALITY>
<GREED>2</GREED>
<FRIENDLINESS>6</FRIENDLINESS>
<AGRESSIVNESS>2</AGRESSIVNESS>
<ACTIVENESS>9</ACTIVENESS>
<EASYGOING>7</EASYGOING>
<EMOTIONALINSTABILITY>7</EMOTIONALINSTABILITY>
<INTEGRITY>2</INTEGRITY>
<HAPPINESS>8</HAPPINESS>
</PERSONALITY>
<ATTRIBUTES>
<HEALTH>100</HEALTH>
<TRAINED>0</TRAINED>
<WEAPON>1</WEAPON>
<SELLABLELAND>2</SELLABLELAND>
<SEEDSFORSALE>0</SEEDSFORSALE>
</ATTRIBUTES>

<ITEMS>
<GOLD>300</GOLD>
<SEEDS>150</SEEDS>
<LAND>4</LAND>
<PRODUCT>400</PRODUCT>
</ITEMS>

<LOCATION>
<STARTLOCATION>10</STARTLOCATION>
<HOMELOCATION>2</HOMELOCATION>
<VALUE>100</VALUE>
<LANDOWNED>1,2,9,10</LANDOWNED>
</LOCATION>

Bibliography

Bibliography

[1] Goal-oriented action planning(goap) [online] accessed: August 2012 available: http://web.media.mit.edu/~jorkin/goap.html.

[2] Heavy rain, quantic dream, sony computer entertainment, 2010.

[3] Heavy rain, wikipedia [online] accessed: August 2012 available: http://en.wikipedia.org/wiki/heavy_rain.

[4] Left 4 dead, turtle rock studio/valve south, valve corporation, certain affinity, valve corporation, 2008.

[5] Left 4 dead, wikipedia [online] accessed: August 2012 available: http://en.wikipedia.org/wiki/left_4_dead.

[6] Mass effect 2, bioware, electronic arts, 2010.

[7] Mass effect 3, bioware, electronic arts, 2012.

[8] Ogre3d [online] accessed: August 2012 available: http://www.ogre3d.org/.

[9] Role-playing video game, wikipedia [online] accessed: August 2012 available: http://en.wikipedia.org/wiki/role-playing_video_game.

[10] Tinyxml2 [online] accessed: August 2012 available: https://github.com/leethomason/tinyxml2.

[11] Ernest Adams. *Fundamentals of Game Design.* New Riders, 2006.

[12] L.M. Barros and S.R. Musse. Planning algorithms for interactive storytelling. *Computers in Entertainment (CIE)*, 5, 2007.

[13] Marcelo Camanho, Angelo Ciarlini, Antonio Furtado, Bruno Feij, and Cesar Pozzer. A model for interactive tv storytelling. In *Games and*

Digital Entertainment (SBGAMES), 2009 VIII Brazilian Symposium, Rio de Janeiro, Brazil, pages 197–206, 2009.

[14] Marc Cavazza, Fred Charles, and Steven J. Mead. Ai-based animation for interactive storytelling. In *Fourteenth Conference on Computer Animation, Seoul, Korea*, pages 113–120, November 2001.

[15] Chris Crawford. *Chris Crawford on Interactive Storytelling*. New Riders, 2005.

[16] Maria Cutumisu, Duane Szafron, Jonathan Schaeffer, Matthew McNaughton, Thomas Roy, Curtis Onuczko, and Mike Carbonaro. Generating ambient behaviors in computer role-playing games. *IEEE Intelligent Systems*, 21:19–27, 2006.

[17] Hussein Karam Hussein Abd El-Sattar. New framework for plot-based interactive storytelling generation. In *5th International Conference on Computer Graphics, Imaging and Visualisation, Penang, Malaysia*, pages 317–322, 2008.

[18] R. Figueiredo, A. Brisson, R. Aylett, and A. Paiva. Emergent story facilitated - an architecture to generate stories using intelligent synthetic characters. In *1st Joint International Conference on Interactive Digital Storytelling: Interactive Storytelling*, pages 218–229, 2008.

[19] D Isla and B Blumberg. challenges for character-based ai for games. *AAAI Spring Symposium on AI and Interactive Entertainment.*, pages 41–45, 2002.

[20] M. Mateas and A. Stern. Faade: an experiment in building a fully realized interactive drama. In *Game Developers Conference*, pages 4–8, 2003.

[21] Sung Hyun Moon, Minhyung Lee, Seokkyoo Kim, and Sangyong Han. Controlling npc behavior using constraint based story generation system. In *4th International Conference on New Trends in Information Science and Service Science (NISS), Gyeongju, South Korea*, pages 104–107, 2010.

[22] Robert Mosher and Brian Magerko. Personality templates and social hierarchies using stereotypes. In *3rd International conference on Technologies*

for Interactive Digital Storytelling and Entertainment (TIDSE), Darmstadt, Germany, pages 207–218, 2006.

[23] Nils J. Nilsson. *Artificial Intelligence: A New Synthesis*. Morgan Kaufmann, 1998.

[24] Jeff Orkin. *AI Game Programming Wisdom 2*. Cengage Learning, 2003.

[25] Vladimir Propp. *Morphology of the Folktale*. University of Texas Press, 1968.

[26] M. Riedl and R. M. Young. An intent-driven planner for multiagent story generation. In *3rd International Conference on Autonomous Agents and Multi Agent Systems, New York*, pages 186–193, 2004.

[27] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2002.

[28] Jeff Orkin (Monolith Studios). Symbolic representation of game world state: Toward real-time planning in games. In *AAAI Challenges in Game AI Workshop*. 2004.

[29] Jeff Orkin (Monolith STudios). Three states and a plan: The a.i. of f.e.a.r. In *Game Developers Conference*, 2006.

[30] Haitao Wang, Cungen Cao, and Yu Pan. Representing and using character feature rules in automatic story generation. In *5th WSEAS International Conference on Applied Computer Science, Hangzhou, China*, pages 619–623, April 2006.