# Side Quest Generation using Interactive Storytelling for Open World Role Playing Games

by

**Sarah Noonan, B.A.I.**

**Dissertation**

Presented to the

University of Dublin, Trinity College

in fulfillment

of the requirements

for the Degree of

**Master of Science in Computer Science**

**(Interactive Entertainment Technology)**

**University of Dublin, Trinity College**

September 2015

# 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.

Sarah Noonan

August 30, 2015

# Permission to Lend and/or Copy

# Acknowledgments

First, I would like to thank my supervisor Mads Haahr, for providing valuable advice and assistance throughout the duration of this dissertation. I would also like to thank him, and the IET course, for allowing me the opportunity to work and research in such an interesting area, and inspiring a love of game AI and interactive storytelling.

I would also like to thank my friends Tony, Gio, and Clementine, for supporting me throughout what has been a stressful year.

I'd like to thank Jean and Daniel, for their invaluable proof reading skills.

And finally I'd like to thank my parents, for being there for me throughout my entire time in education. Without your unwavering support I would have given up a long time ago.

Thank you all.

Sarah Noonan

*University of Dublin, Trinity College*
*September 2015*

# Side Quest Generation using Interactive Storytelling for Open World Role Playing Games

Sarah Noonan
University of Dublin, Trinity College, 2015

Supervisor: Mads Haahr

This dissertation aims to explore the use of interactive storytelling techniques in the procedural generation of side quests for open world role playing games.

Side quests can play a vital role in providing players with a non-linear feeling during open world games. They do this while also providing goals, tasks, and rewards to keep players invested in the game. However, the prohibitive cost of designing and implementing a sufficient number of side quests to populate a large open world game often requires that the majority of such quests be overly simple and cheap to produce (such as the ever unpopular “fetch quests”).

Interactive storytelling has been used to control the pacing and difficulty of commercial games, however its full narrative potential has so far mostly been confined to academic and text-based games.

This dissertation presents a model that integrates interactive storytelling with procedural quest generation, allowing for the generation of engaging quests that have relevance to both the state of the game world, and the overall narrative.

The implementation of this model shows definite potential for the use of interactive storytelling in the procedural generation of quests for games in the future, in order to create more enjoyable, believable, and interactive experiences for players.

# Contents

**Acknowledgments** **iv**

**Abstract** **v**

**List of Tables** **x**

**List of Figures** **xi**

**Chapter 1 Introduction** **1**

1.1 Motivation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1

1.2 Objectives . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2

1.3 Dissertation Roadmap . . . . . . . . . . . . . . . . . . . . . . . 4

**Chapter 2 State of the Art** **5**

2.1 Believability of NPCs . . . . . . . . . . . . . . . . . . . . . . . . 5

2.1.1 Believability vs. Realism . . . . . . . . . . . . . . . . . . . 5

2.1.2 Modelling Human Behaviour . . . . . . . . . . . . . . . . . 6

2.1.3 Player Replacements vs. NPCs . . . . . . . . . . . . . . . . 7

2.1.4 Believable Agents . . . . . . . . . . . . . . . . . . . . . . . 7

2.2 Interactive Storytelling . . . . . . . . . . . . . . . . . . . . . . . 10

2.2.1 Patterns in Storytelling . . . . . . . . . . . . . . . . . . . . 11

2.2.2 Drama-Management Based Interactive Storytelling . . . . . . . 12

2.2.3 Character Based Interactive Storytelling . . . . . . . . . . . . 12

2.2.4 Multi-Solution Levels . . . . . . . . . . . . . . . . . . . . . 14

2.3 AI Architectures . . . . . . . . . . . . . . . . . . . . . . . . . . 15

2.3.1 Requirements . . . . . . . . . . 15
2.3.2 Finite State Machines . . . . . . . . . . 15
2.3.3 Behaviour Trees . . . . . . . . . . 16
2.3.4 AI Planners . . . . . . . . . . 17
2.4 NPCs in Games . . . . . . . . . . 18
2.4.1 Companions . . . . . . . . . . 18
2.4.2 Performance Issues . . . . . . . . . . 19
2.5 Side Quests in Open World Games . . . . . . . . . . 20
2.5.1 Definition and Purpose . . . . . . . . . . 20
2.5.2 Quest Generation . . . . . . . . . . 21
2.6 Conclusion . . . . . . . . . . 23

**Chapter 3 Design** **24**
3.1 Influences . . . . . . . . . . 24
3.2 Defining the Model . . . . . . . . . . 25
3.2.1 Quest System . . . . . . . . . . 25
3.2.2 Character Modelling . . . . . . . . . . 26
3.2.3 Drama Management . . . . . . . . . . 29
3.3 Illustrating the Model . . . . . . . . . . 31
3.3.1 Example Plot Fragment: The Jealous Suitor . . . . . . . . . . 31
3.4 Procedural Content Generation . . . . . . . . . . 34
3.5 Conclusion . . . . . . . . . . 35

**Chapter 4 Implementation** **36**
4.1 Platform and Existing Code Base . . . . . . . . . . 36
4.1.1 Unity Game Engine . . . . . . . . . . 36
4.1.2 Existing Code Base . . . . . . . . . . 36
4.2 Character Modelling . . . . . . . . . . 37
4.3 Quest System . . . . . . . . . . 39
4.4 Plot Fragments . . . . . . . . . . 42
4.5 Drama Management . . . . . . . . . . 43
4.6 Conclusion . . . . . . . . . . 44

**Chapter 5 Evaluation** **45**

5.1 Character Modelling . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 45
5.1.1 Mental/Physical State . . . . . . . . . . . . . . . . . . . . . . . . 45
5.1.2 Relationships and Relationship Traits . . . . . . . . . . . . . . . . 46
5.1.3 Character Traits . . . . . . . . . . . . . . . . . . . . . . . . . . . 46
5.1.4 Procedural Generation of Characters . . . . . . . . . . . . . . . . 47
5.2 Procedural Generation of Quests using Interactive Storytelling . . . . . 47
5.2.1 Quest System . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 47
5.2.2 Plot Fragments . . . . . . . . . . . . . . . . . . . . . . . . . . . . 48
5.2.3 Drama Management . . . . . . . . . . . . . . . . . . . . . . . . . 48
5.3 Extensibility and Ability to Integrate with Other Systems . . . . . . . 49
5.4 Effcient Implementation . . . . . . . . . . . . . . . . . . . . . . . . . . 50

**Chapter 6 Conclusion 51**
6.1 Main Contributions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 51
6.2 Limitations & Future Work . . . . . . . . . . . . . . . . . . . . . . . . 52
6.2.1 Limited Prototype . . . . . . . . . . . . . . . . . . . . . . . . . . 52
6.2.2 Integration . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 53
6.2.3 Events System . . . . . . . . . . . . . . . . . . . . . . . . . . . . 53
6.2.4 UI/Scripting Support for Non-Programmers . . . . . . . . . . . . 53
6.3 Final Thoughts . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 54

**Bibliography 55**

# List of Tables

2.1 Level-of-Detail(LOD) Levels in *Neverwinter Nights*[7] . . . . . . . . . . . 19

3.1 States - Ranges and Meanings [20] . . . . . . . . . . . . . . . . . . . . . . . . 27
3.2 Relationship Traits - Ranges and Meanings . . . . . . . . . . . . . . . . . . 28
3.3 Traits - Ranges and Meanings . . . . . . . . . . . . . . . . . . . . . . . . . . 29

# List of Figures

2.1 Freytag's Pyramid . . . . . . . . . . . . . . . . . . . . . . . . . . . 13
2.2 An example plot fragment from *UNIVERSE* [33]. This plot fragment attempts to achieve the author goal of making a married couple unhappy. 14
2.3 Example Behaviour Tree [58] . . . . . . . . . . . . . . . . . . . . . . . . 16
2.4 The *SQUEGE* process [47] . . . . . . . . . . . . . . . . . . . . . . . . . 21

3.1 The plot fragment selection process of the drama manager. . . . . . . . 30
3.2 The character constraints of the "Jealous Suitor" subplot. . . . . . . . 33

4.1 Open quest log displaying an active quest. . . . . . . . . . . . . . . . . 40
4.2 Quest star to indicate the NPC has a quest for the player. . . . . . . . 41

5.1 Unity's profiler showing McNulty's implementation. The performance shows a computation spike, which adversely affects the frame rate. This corresponds to 10am in-game time, when all NPCs go and eat breakfast. Other than this, performance is steady. . . . . . . . . . . . . . . . . . . . 49
5.2 A performance snapshot of the implementation taken in the Unity profiler. There is no single large performance spike, as NPCs do not have specific routines to follow and so do not all change activities at once. However, performance is less steady overall, as NPCs are changing tasks more frequently and at irregular intervals. . . . . . . . . . . . . . . . . 50

# Chapter 1

# Introduction

This dissertation explores the use of interactive storytelling techniques in the procedural generation of side quests for open world role playing games (RPGs). This project proposes a hypothesis that incorporating interactive storytelling techniques, such as author goals and sub-plot preconditions, into quest generation would create quests relevant to the NPCs, resulting in a more interesting and immersive player experience. This could also serve the purpose of weaving an underlying narrative that emerges organically as the player progresses through the game.

The hypothesis is tested through the creation of a prototype that demonstrates such a system. It is intended that this prototype will demonstrate the system's potential to produce side quests that are interesting and enjoyable for the player, while reducing the development time which would usually be necessary for the manual creation of side quests. It is expected that the model will remain versatile enough to be applicable to a wide variety of open world role playing games.

## 1.1 Motivation

Open world games, or "sandbox" games, are games with large scale, non-linear worlds that the player is free to explore. Open world role playing games typically take place in an environment that is populated with a sequence of quests that the player must complete in order to progress through the game's story. Along with these main quests, the player is given the opportunity to take on various side quests, which provide the player freedom to gain extra rewards and experience in the game without adhering to a linear storyline [47].

With the larger worlds and increased playtime demanded of recent open world role playing games, the number of side quests can grow to over a hundred[64][30]. The designing, writing, and scripting of these quests consumes a significant amount of development resources. This results in the need for cost saving measures, such as the reliance on a large number of "fetch quests", quests which merely involve fetching an item for an NPC. These quests are quick to develop, but can be boring for the player, and fail to reflect the state of the game world or the point in the storyline.

Interactive storytelling has been used in commercial games (such as *Left 4 Dead* [61] and *Far Cry 2* [43]) to control the pacing and difficulty[33]. However, the potential of interactive storytelling to control the narrative of the game has mostly been confined to academic and text-based games (such as *Façade* [2] and *Galatea* [57]).

## 1.2 Objectives

The main objectives of this dissertation are to:

1. Create a generalised character modelling system, on which interactive storytelling can be based.
2. Create a system which employs interactive storytelling to procedurally generate interesting and relevant side quests at run time.
3. Ensure system is extensible and can be integrated with other systems.
4. Provide an efficient implementation of such a system that does not have a significant impact on performance or memory consumption.

### Character Modelling

Character modelling is not the main focus of this dissertation, however, it is necessary for the interactive storytelling objective, given that unique and interesting characters are required in order to generate a narrative. Therefore, some model needs to exist to represent each NPC's unique personality, and their relationships with each other. In order to support the generation of a large number of NPCs, this character modelling should support procedural generation.

### Procedural Generation of Quests using Interactive Storytelling

The main objective of this dissertation is to research and develop a system to procedurally generate side quests at run time. This would reduce the development burden for side quest generation for large scale open world role playing games.

Rather than relying on repetitive fetch quests, it is hoped to integrate interactive storytelling techniques into the quest generation system, in order to create interesting and relevant quests.

### Extensibility and Ability to Integrate with Other Systems

If the system is to be versatile and suitable for use in a wide variety of games, it must be able to be used alongside other AI and NPC systems, such as activity selection and relationship systems. Additionally, it must be extensible, to support the generation of quests for different genres of open world role playing games.

### Efficient Implementation

AI is typically afforded a low budget for performance and memory in games, compared with other systems such as rendering and physics simulation. Millington et al [42] state that on previous generation hardware (Playstation 3 and Xbox 360), AI was typically afforded only 8MB of memory, out of a total of 512MB available. Therefore a model will be more likely to be used in a variety of applications if it remains light-weight and efficient.

## 1.3 Dissertation Roadmap

In chapter 2, the state of the art in the industry will be reviewed, including the believability of NPCs, interactive storytelling, and methods for reducing the footprint of generating large numbers of side quests.

Chapter 3 will outline the design of the model in general terms, along with how it was influenced by the current state of the art.

Chapter 4 will describe the specifics of the prototypes implementation. Specifically, how the character modelling and quest generation components were developed.

In chapter 5 the prototype will be evaluated by how well it has fulfilled the objectives outlined in the previous section.

Chapter 6 will draw conclusions about the merits and limitations of the model, as well as possible future improvements.

# Chapter 2

# State of the Art

In this chapter, a review of the state of the art of the industry will be presented. Research was conducted into the areas of believability of NPCs, interactive storytelling, and side quest generation.

## 2.1 Believability of NPCs

NPCs are a staple in any open world role playing game. In order for a world to feel alive and interesting, it must be populated with believable NPCs. In this section, research on the believability of NPCs will be reviewed.

### 2.1.1 Believability vs. Realism

In the casual discussion of games, the quality of the physics or rendering is often measured in terms of how "realistic" it is. However, it is important to make a distinction between the terms "believability" and "realism". Realism refers to how close something is to real life, while believability refers to how plausible something is for the given situation. For example, it would not be realistic for a human agent to have to ability to fly. However, if the agent was framed as a mage or wizard, it may be believable for them to be able to fly, within the context of the game. For agents, the goal is believability, not necessarily realism.

### 2.1.2 Modelling Human Behaviour

It is impossible to create believable human agents that emulate human behaviour without first defining and modelling this behaviour.

McCrae et. al [38] outlines the *Five Factor Model*, a model that describes personality according to 5 properties.

**Openness to Experience** A person's curiosity, and their appreciation for art, emotion or adventure.

**Conscientiousness** A person's tendency to be organised or dependable, self-disciplined, to prefer planned behaviour over spontaneous.

**Extraversion** A person's energy levels, sociability, tendency to seek the company of others.

**Agreeableness** A person's tendency towards compassion and cooperation. Also describes how well-tempered and trusting they tend to be.

**Neuroticism** A person's tendency to experience unpleasant emotions easily, such as anger or anxiety. Also refers to the degree of emotional stability and impulse control.

### 2.1.3 Player Replacements vs. NPCs

Agents in games may be intended to act as either a player replacement or an NPC. A player replacement is an agent that is designed to emulate a real human playing the game. These tend to be used in multiplayer games, for example *Dota 2* [14], to take the place of human players in the case of no internet connection, or if the player merely wishes to practice. Creating believable player replacements was the subject of the *BotPrize* competition [28], which aimed to create agents to play the game *Unreal Tournament 2004* [23] that were indistinguishable from human players. Player replacements are not the subject of this dissertation.

In contrast to player replacements, NPCs are not intended to act as if a human was controlling them, but instead act as if they are their own living entity within the game world. This allows the agents to act in ways that are not necessarily plausible for a human player to perform, as the player has a willingness to suspend disbelief in order to immerse themselves in the game world.

### 2.1.4 Believable Agents

Unlike most areas of AI, agents for games do not need to able to accomplish useful tasks, or be effective problem solvers. Instead, their main purpose is to be believable, as real characters within the game world.

Loyall [36] outlines the properties required for believable agents. He draws on work from character artists, such as animators [63] and writers [19].

**Personality** Personality is regarded as the single most important requirement for believable agents [36].

> *"For a character to be that real, he must have a personality, and, preferably, an interesting one."* - Thomas, Johnston. The illusion of life : Disney animation.[63]

Personality can be described as the characteristics that define an agent as an individual, and that brings them to life.

According to Egri [19], convincing characters must be “tridimensional”; they must be specified along three dimensions: physiology, sociology and psychology.

**Emotion** Believable agents must have emotional reactions, and be able to express these emotions. The emotions of an agent can be viewed as an extension of their personality, and so the expression of these emotions must be true to this.

**Self-Motivation** An agent must be self-motivated, acting of its own accord, rather than solely responding to external stimuli.

For example, a person that is reading does not merely sit silently until interrupted, but may move position periodically, or go and pursue another activity if they grow bored.

Self-motivation ensures that agents are seen as having their own thought process.

An common example of self-motivation used in games are *barks*. Barks are utterances from companion NPCs during idle time, such as grunts, sighs, or comments. They ensure that the companions do not seem lifeless when not directly involved in combat or dialogue.

**Change** In order for an agent to be believable, it should grow and change with time. Again, these changes must not be arbitrary, but must be in line with the agent’s established personality.

An example of change could be an agent growing to like or dislike the player, depending on their actions (and how the agent’s personality views these actions).

**Social Relationships** Humans are social animals, so human agents must interact with other agents. These interactions should be influenced by their relationship, and in turn this relationship should be influenced by their interactions.

Relationships can not merely be described in simple terms such as "friends" or "enemies". Just as every personality is different, so must every relationship be different.

**Consistency of Expression** Humans have a number of ways of expressing thoughts or feelings, such as words, facial expression, posture, movement, etc. In order for an agent to be believable, all of these must be consistent with each other.

**Illusion of Life** This last requirement of believable agents that Loyall outlines is actually a collection of requirements. These include properties that may be taken for granted in other art forms (for example, in acting, it is taken for granted that characters can walk and talk at the same time because the human actor can), but for the creation of agents must be explicitly stated.

**Appearance of Goals**
Agents must appear to be working towards a goal or goals.

**Concurrent Pursuit of Goals**
Agents must be capable of pursuing multiple goals at once, rather than just one at a time.

**Parallel Action**
Agents must be able to do two things at once (for example, walking and talking).

**Reactive and Responsive**
Agents must react to stimuli and respond at reasonable rates.

**Situated**
Agents must seem to be aware of their environment and adapt to the current situation.

**Exist in a Social Context**
Agents must be aware of the social context and act in accordance with social and cultural conventions.

**Broadly Capable**

Broadly capable means that agents should be capable of all actions that a real human is, such as thinking, sensing, talking, listening, etc.

**Well Integrated (Capabilities and Behaviours)**

Often in the creation of agents, different capabilities of the agent are handled by separate components, for example there might be a separate sensing component, a locomotion controller, etc. If these components are not well integrated with each other, there may be a noticeable pause or abrupt change when an agent changes from performing on action to another. It is therefore required that all capabilities and behaviours of an agent are well integrated with each other in order to appear natural and believable.

## 2.2 Interactive Storytelling

Typically, story-driven games follow a linear storyline, for example *Bioshock: Infinite* [24] and *The Last of Us* [17]. Kline refers to these games as "rollercoaster" games, as the story is a journey to be experienced by the player, but they can have no influence on it [33]. The story is limited to cut-scenes, with pre-recorded dialogue and no input from the player. This allows the writers to have full control of the narrative, and produce a high quality story, but limits interactivity, as the player has no impact on the story of the game, even though they are playing as the main character. This can also introduce a disparity between the actions of main character when it is controlled by the player, and when it is in a cut-scene This is referred to as *ludonarrative dissonance* [29] and leads to a reduction in believability.

Interactive storytelling involves creating stories that can be changed and influenced by the player. This increases interactivity and immersion. It also improves re-playability, as different actions taken by the player will influence the story and possibly result in a different outcome. However, since the writer does not have full control of the story, it can result in low quality stories. Also, creating a story with multiple outcomes and consequences increases the writing and programming time required in development.

### 2.2.1 Patterns in Storytelling

In the study of narrative and narrative structure, it has been observed that stories can often be found to adhere to specific patterns. These patterns can be found in stories across multiple storytelling mediums. A notable example of this is the "Hero's Journey" [9]. This pattern consists of 17 stages, and involves a hero going on an adventure, overcoming an obstacle or enemy, before returning home victorious. The "Hero's Journey" has been identified in stories throughout history, from *Moby Dick* [41] to *Star Wars* [37].

**Event Patterns** In *Morphology of the Folktale* [54], Vladimir Propp decomposes Russian folktales into patterns. He outlines 25 "functions" (patterns), which describe the main events of the story in terms of interchangeable variables such as characters, objects, locations, etc.

For example, the "absentation" function: "One of the members of a family absents himself from the home." This function, in practice, could take many forms. The family member could be a parent, a sibling, or child, while the type of "absentation" could be going travelling, or to work, or even death.

Propp also describes the various sequences of these functions that are commonly used.

**Plot Patterns** While *Morphology of the Folktale* deals with patterns that describe the events in the story, in *The Seven Basic Plots: Why we tell stories* [5], Booker outlines patterns that describe the overarching plot of the story as a whole‘, for example, "Voyage and Return" as well as the classic "Comedy" and "Tragedy". These plot patterns are outlined as a series of events, which in turn take the form of event patterns, such as those described by Propp.

### 2.2.2 Drama-Management Based Interactive Storytelling

In order to allow the game designer to retain control during interactive storytelling, the story can be controlled by a *drama manager*. During development, the designer can specify a "policy", or a desired story plan. The drama manager can then monitor the progress of the interactive story at runtime, and can use "story moves" to attempt to direct the story towards the specified policy [33].

**Left 4 Dead** In combat games with enemies spawning in waves, typically these spawns are location dependent, ie. they are triggered when the player reaches the area. However, in *Left 4 Dead* [61], the enemy spawns are instead controlled by a drama manager. In this case, the policy is the desired tension and difficulty of the game. The story moves used to alter the tension and difficulty to attempt to adhere to the policy are enemy spawns and equipment drops.

For example, if there has not been any enemy spawns in some time, the drama manager may try to increase the tension by spawning enemies. Similarly, if the difficulty is above that specified in the policy, equipment drops may be triggered.

**Façade** The drama manager in *Façade* uses a policy that specifies a desired tension arc. This arc is based on *Freytag's Pyramid* [22] (Fig. 2.1), a pattern of tension observed in classic literature. A story that follows Freytag's pyramid consists of tension gradually rising until it reaches a climax, and then decreasing as the story resolves.

The story moves used by *Façade*'s drama manager are "beats". These beats are minor actions or conversation topics that the NPCs in the game can introduce. For example, if the drama manager wishes to increase tension, it could select a beat that introduces a controversial topic of conversation, to create tension between the characters.

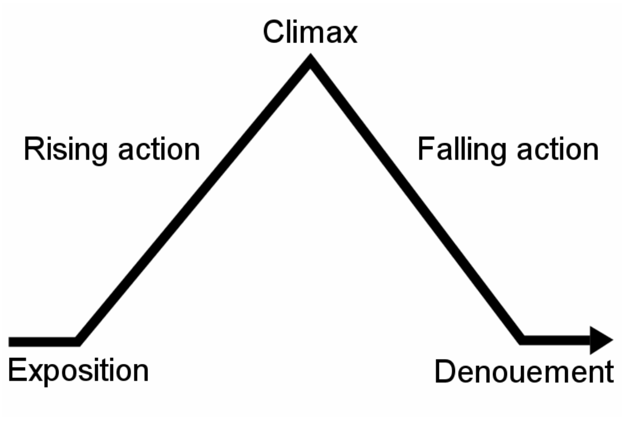


**Figure 2.1:** Freytag's Pyramid

### 2.2.3 Character Based Interactive Storytelling

Character based interactive storytelling involves modelling NPCs, and the emergent events that occur form the story. *TALE-SPIN* [40] was an early interactive storytelling application that implemented this approach, along with a world model, in its text-based environment. Cavazza et al. [11] also implemented this approach in a more modern engine, and added user interactivity.

This emergent approach creates opportunity for great interactivity, and also increases believability, as the NPCs are following their own goals and the world changes based on the actions they take.

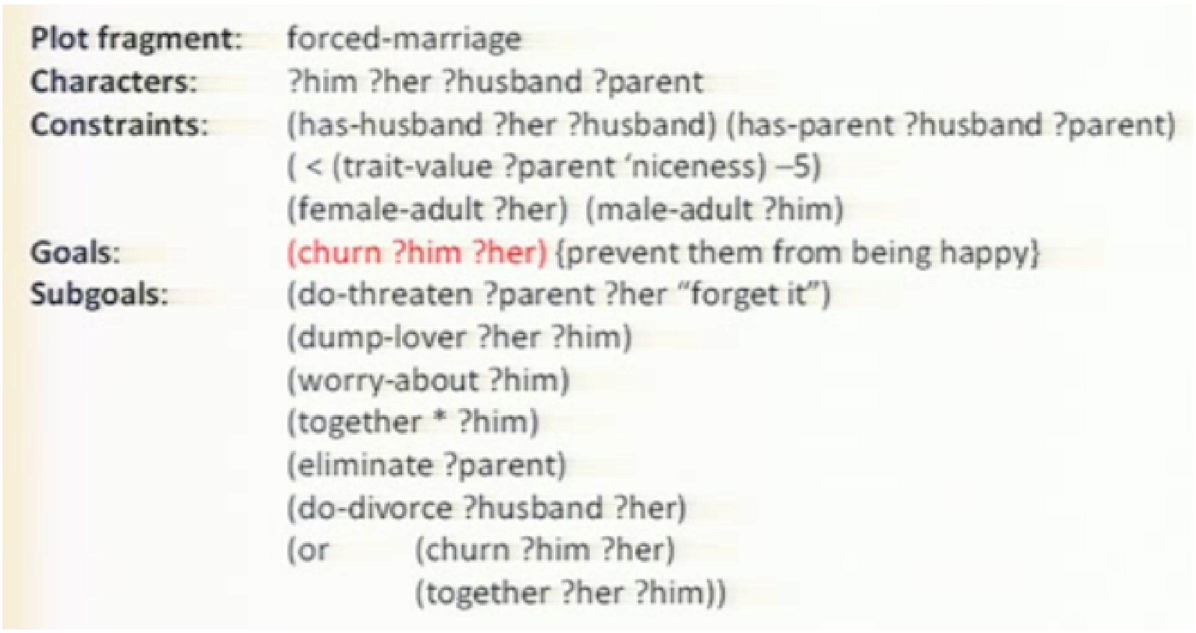
However, unlike the drama manager based approach, the game designer has little control over the story as it evolves at runtime. This can lead to emerging stories that have no structure, and are uninteresting [33].

**Author Modelling** Author modelling extends character based interactive storytelling by adding author goals and plans.

Author goals are goals that exist to add dramatic interest to the story of the world. They may not be rational goals for the characters themselves to have. For example, an author goal may be for a character to be unhappy, while it is not rational for a character to wish unhappiness on himself.

Author plans are plot fragments that can be used to achieve the author goals.

*UNIVERSE* [34] was a system that used this approach to create "soap-opera" type stories (Fig. 2.2).



```
Plot fragment:  forced-marriage
Characters:     ?him ?her ?husband ?parent
Constraints:    (has-husband ?her ?husband) (has-parent ?husband ?parent)
                ( < (trait-value ?parent 'niceness) –5)
                (female-adult ?her)  (male-adult ?him)
Goals:          (churn ?him ?her) {prevent them from being happy}
Subgoals:       (do-threaten ?parent ?her "forget it")
                (dump-lover ?her ?him)
                (worry-about ?him)
                (together * ?him)
                (eliminate ?parent)
                (do-divorce ?husband ?her)
                (or        (churn ?him ?her)
                           (together ?her ?him))
```

**Figure 2.2:** An example plot fragment from *UNIVERSE* [33]. This plot fragment attempts to achieve the author goal of making a married couple unhappy.

### 2.2.4 Multi-Solution Levels

Many open world games such as the *Assassin's Creed* series [44], the *Hitman* series [31], and the *Metal Gear Solid* series [32] have quests that may have multiple solutions, often with different solutions to the same quest having different consequences. This form of interactive storytelling allows the player to be more creative and original, rather than following a predetermined path to the solution to the quest.

However, these multi-solution levels can be complicated and difficult for level designers to balance. For example, a level may be designed to be difficult, however there may be easier solutions than the designers planned, and all these solutions must be discovered and evaluated.

Pizzi et al. [51] propose a method that uses heuristic search planning to generate level solutions, and presents them as a storyboard for ease of understanding. The system also allows level designers to alter the world state of the game as well as existing level constraints, and observe the consequences on the level's solutions. The system was tested on levels from the game *Hitman: Blood Money* [31], and generated similar solutions to those found in online walkthroughs and by the level designers. This method allows multi-solution levels to be created by designers with a greatly lessened overhead on testing and evaluation.

## 2.3 AI Architectures

NPCs can be controlled by a number of different AI architectures. In this section the requirements for an AI architecture for NPCs will be outlined, and the architectures commonly used will be reviewed.

### 2.3.1 Requirements

According to Cutumisu et al [16], suitable AI architectures for game NPCs must be able to support behaviours that exhibit the following properties:

- **Responsive** – can react quickly to the environment,
- **Interruptible** – can be suspended by other behaviours or events,
- **Resumable** – can be continued after interruption,
- **Collaborative** – can initiate and participate in joint behaviours with other agents,
- **Generative** – are easy to create and implement by non-programmers.

### 2.3.2 Finite State Machines

Finite State Machines (FSMs) are the most common architecture for AI in games [26]. An FSM consists of a number of states, to represent actions, that an agent may be in. States may have a number of transition criteria, which control when the agent should switch from one state to another. An agent may only be in one state at a time.

FSMs have a number of advantages [8]. They are quick and easy to implement, as well as being easy to debug. They are efficient, intuitive and easy to understand, and flexible.

However, FSMs also have a number of drawbacks [12]. They are not resumable, do not work well with concurrency, and scale poorly. Additionally, the nature of fixed states with fixed transitions means that NPCs that use FSMs can become easy to predict.

**Variations on FSMs** FSMs have been altered in an attempt to support more complex behaviours. Some examples are *Stack-based FSMs*, which are able to support resumable behaviours, and *Hierarchichal FSMs*, which use super-states to share and reuse state transitions.

### 2.3.3 Behaviour Trees

Behaviour Trees (BTs) are trees of hierarchical nodes, each representing a behaviour. The behaviours are in the tree according to priority, so to find the most appropriate behaviour, the tree is traversed, and the first valid behaviour is executed.

To illustrate, see Figure 2.3. For any nodes marked “Sequence”, all child nodes are executed one after another, from left to right. For any nodes marked “Selector”, only one child node is to be executed: the left most valid child node.

So for this tree, the agent first walks to the door. Then, it either opens the door, unlocks the door and then opens the door, or smashes the door, whichever is the first possible action. Then the agent walks through the door, and finally closes the door.

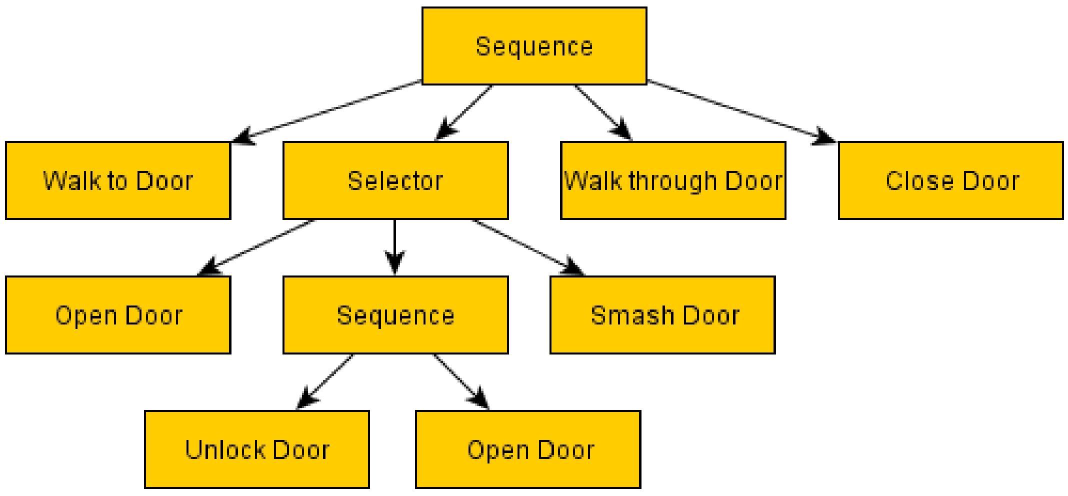


**Figure 2.3:** Example Behaviour Tree [58]

Behaviour trees have been used for the NPCs in *Project Zomboid* [59] [58].

### 2.3.4 AI Planners

AI planners are an architecture that more accurately reflect the human thought process. Two different AI planners are described here.

#### STRIPS

The STanford Research Institute Problem Solver (STRIPS) is an AI planner that was originally proposed in 1971, to aid in problem solving for robots [21].

STRIPS involves defining the relevant properties of any moment in time as a "world model", and any possible actions as "operators". Operators are defined by their necessary requirements in the world model, as well as their effect on the current world model.

To plan the best actions to reach the desired goal, the goal state as well as the current conditions are expressed as world models. Working backwards from the goal world model, a heuristic search algorithm is carried out on possible operators in an attempt to reach the current world model.

#### GOAP

Goal Oriented Action Planner (GOAP) [50][49] is an AI planner first used in the game *F.E.A.R.* [53].

Similar to STRIPS, possible actions are defined, as well as the desired goal, and a search is performed to find the best sequence of actions for the scenario.

More recently GOAP was used in the game *Tomb Raider* [18][13].

## 2.4 NPCs in Games

An NPC refers to any character that is not controlled by the player. This includes enemies, companions, and background characters.

**Enemies** Enemies are technically classified as NPCs, but are mostly not referred to as such. This dissertation is not concerned with enemy NPCs.

### 2.4.1 Companions

Companion NPCs are NPCs that accompany the player as they move through the game world.

In the past, companion NPCs have had a history of being "annoying" [35]. Players are forced to complete "escort missions", where they must escort an NPC to safety. Often these NPCs are helpless, and have poor AI that results in them running into enemies or into the line of fire. An example of these types of companion NPCs are Ashley Graham from *Resident Evil 4* [10] and Natalya from *GoldenEye 007* [55].

Even when companion NPCs are designed to be helpful to the player, rather than a hindrance, there can still be shortcomings. In *Skyrim* [60] the companions have weapons and abilities that aid the player in battle, but can often crowd the player and obstruct their path. The extent if this problem is evidenced by the fact that “Move it Dammit!” [52], a user-created mod that attempts to minimise this problem, has over 300,000 downloads.

Elizabeth from *Bioshock Infinite* [24] and Ellie from *The Last of Us*[17] are recent examples of successful companion NPCs. They are designed carefully to be likeable companions, rather than burdens [1]. To achieve this, they have use to the player in combat (Elizabeth can point out enemy snipers and loot drops, Ellie can shoot enemies), stay out of the player’s line of fire, and provide conversation during non-combat periods.

### 2.4.2 Performance Issues

In large scale open world games, the large number of NPCs required to populate the world can have significant performance costs. In order to reduce computation costs, often the world is segmented into areas, with the NPCs in an unused area being “switched off”. This can cause significant loading times when the player wishes to move between areas. In *Baldur’s Gate* [3], all players in the multiplayer party had to assemble in order to move between areas together. This made it impossible for individual exploration, and is regarded as one the games most hated features [7].

For *Neverwinter Nights* [4], *Bioware* (also the developers of *Baldur’s Gate* attempted to mitigate this problem by introducing *level-of-detail* into the NPC AI behaviour [7]. Level-of-detail is a common optimisation technique used in rendering or physics, where the model or simulations are simplified when far away or out of sight of the viewer. In *Neverwinter Nights*, it was used by altering the processing frequency, pathfinding, and combat rules of NPCs according to their proximity to the player.

The level-of-detail(LOD) is discretised into five different levels (Table 2.1).

For LODs 1, 2 and 3 (ie. every character that visible to a player), full pathfinding

**Table 2.1:** Level-of-Detail(LOD) Levels in *Neverwinter Nights*[7]

| LOD | Classification |
|---|---|
| 1 | Player Characters (PCs) |
| 2 | NPCs fighting or interacting with a PC |
| 3 | NPCs in view of a PC |
| 4 | NPCs out of view of a PC |
| 5 | NPCs in an area with no PCs |

(IDA*) is performed.

For LOD 4, the NPCs are out of view of the player, so full pathfinding is unnecessary. However, since they are still in the same area as the player, the paths they take can not be completely ignored. Since the terrain is split into tiles, a path can be found between the tiles (inter-tile pathfinding), and the NPCs are merely jumped tile to tile, without having to navigate within the tiles themselves (intra-tile pathfinding). This resulted in an over 90% decrease in computation time for the pathfinding of LOD 4 NPCs.

For LOD 5, the NPCs are in an area where there are no players, so their actual paths do not matter. They are transported from place to place with a delay corresponding to the direct distance, to simulate travel time.

Due to advancements in hardware and optimisation, it is now possible to have large scale open worlds populated with NPCs, that require no inter-area loading screens. An example of this is *The Witcher 3: Wild Hunt* [56] [27].

## 2.5 Side Quests in Open World Games

In this section, the purpose of side quests in open world role playing games will be explored, as well as methods to ease the generation of side quests.

### 2.5.1 Definition and Purpose

In open world RPGs, the player is generally not guided from place to place. Instead, they are presented an open environment to explore. Throughout the environment, they may encounter tasks to complete, in return for rewards such as gold or equipment, or merely to forward the story. These tasks are referred to as quests.

Often these quests are unrelated to the main storyline of the game. These are called *side quests*.

According to Onuczko et al [47], side quests serve four main purposes:

- to promote an open-world feeling,
- to provide opportunities to gain rewards and experience,
- to add interest and back story to the game (without over-complicating the main storyline),
- to reward exploration.

### 2.5.2 Quest Generation

It is not uncommon for large scale open world games to include dozens, if not hundreds, of side quests. This poses a major problem for developers. Each individual quest must be designed, programmed, have dialogue written and voice acting recorded for, etc. The amount of time and resources required to be spent on side quests can adversely affect the main storyline or other aspects of the game.

A number of attempts have been made to ease the side quest generation process.

#### SQUEGE

*SQUEGE* (Side-QUEst GEnerator tool) is a tool, created by Onuczko et al [47][48], to aid level designers in side quest generation. The designer creates “patterns”, activities

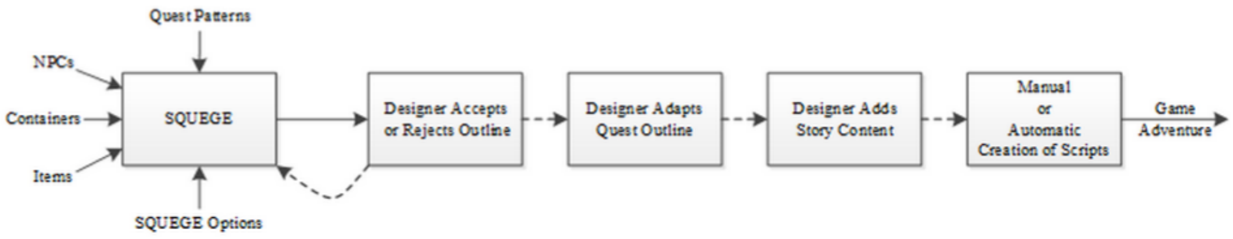

Quest Patterns
NPCs
Containers
Items
SQUEGE
SQUEGE Options
Designer Accepts or Rejects Outline
Designer Adapts Quest Outline
Designer Adds Story Content
Manual or Automatic Creation of Scripts
Game Adventure

**Figure 2.4:** The *SQUEGE* process [47]

to be completed with placeholders, as well as specifying game objects that can be used in these placeholders.

For example, a designer may create an "Acquire Item" pattern, which requires NPC, item, and item location placeholders. The designer then must specify a list of possible NPCs, items, and item locations that can be used within this pattern.

Patterns and game objects need only be specified once, but then can be reused across multiple quests.

Once all patterns and game objects have been setup, the *SQUEGE* system generates side quests be assembling patterns and game objects together (either randomly, or according to specified constraints). The designer than then accept, reject of modify the generated quest outline, and proceed to add story content (such as dialogue). Scripts can then be created by a programmer or script generator, and the quest can be added to the game. This process is summarised in figure 2.4.

*SQUEGE* is only capable of generating side quest outlines, and so there is still work required to produce a fully implemented side quest. However, it does ease the game design process. The system can result in quests that are of similar quality to manually scripted quests, as shown from user tests [47].

### Radiant Story

For *Skyrim* [60], *Bethesda* developed the *Radiant Story* [45] system. This system had a number of functions, including enabling NPCs to respond to a variety player actions, but also aided in the generation and flexibility of side quests.

Radiant quests would not have detail defined explicitly. Instead, similar to *SQUEGE*, there would be a pattern to be followed, with placeholders for game objects. In *Radiant Story*, these placeholders were referred to as aliases.

Radiant quests can be used in two different ways. Firstly, if an important person in a scripted quest becomes unavailable (eg. dies), this person can be swapped out for another person so that the quest can continue. Secondly, it provides an easy way to implement "job" quests, ie. standard quests associated with an in-game guild that always follow a specific pattern (eg. the assassin's guild has job quests that involve killing specific NPCs). Using *Radiant Story*, NPCs, items, and locations can easily be swapped into the job quests to provide a limitless number of quests, with no extra development time.

Unlike *SQUEGE*, which is a design tool to be used during game development, *Radiant Story* generates quests at run time. This is an advantage in many ways. Development time is further decreased, and the randomness introduced to the game provides replayability and limitless side quests. However, the inability for a designer to reject or modify the generated Radiant quests means that some quests can be incohesive with the rest of the game.

## 2.6 Conclusion

In this chapter, a review of research that has been conducted in the areas of NPC believability, interactive storytelling, and side quest generation has been conducted. It is clear from this research that there is still opportunity for advancements to be made in these areas.

# Chapter 3

# Design

In this chapter, the main influences on the model will outlined, before detailing the design. The model will then be illustrated with an example.

## 3.1 Influences

The initial influence on this dissertation came from the quest fallback system found in McNulty's residual memory model [39]. Other influences include Fallon's mental/physical states model [20], Onuczko's *SQUEGE* [48], and Lebowitz's *UNIVERSE* [34].

McNulty's quest fallback system was an extension of his memory/goals model. For example, if an NPC requires a hammer, but cannot find one, he creates a quest for the player, to find a hammer and bring it to him. McNulty's model is described in more detail in section 4.1.2.

Fallon's mental/physical states model is a system that attempts to model an NPC's current state of mind and physical status using a multidimensional approach. NPC's are not merely happy or sad, but can have multiple feelings at once, such as happy and angry, or sad and excited. Fallon's model is described in more detail in section 3.2.2.

*SQUEGE* is a design tool that can be used to decrease the amount of time it would take to design a large number of side quests, as was described in section 2.5.2.

*UNIVERSE* was a system that used both plot-based and character-based interactive storytelling techniques to attempt to create a soap-opera-style narrative. This was achieved through the use of author goals, as described in section 2.2.3.

## 3.2 Defining the Model

In this section, the components of the model will be explained in detail.

### 3.2.1 Quest System

The core components of the quest system of this model was originally based on the *SQUEGE* system [48]. Quests are defined by *patterns* and *points*, as described in the next sections.

#### Quest Patterns

Quest patterns are reusable outlines that describe the encounters and actions that must be completed in order to complete a quest. They consist of a a quest giver, as well as a number of quest points.

#### Quest Points

A *quest point* is a single, low-level objective of a quest (for example, "Acquire Item").

A quest point can be in one of three states: inactive, active, or completed. A quest point can only be completed when it is active.

Each quest point has a list of *enablers*, which are other quest points in the same quest pattern that must be completed to activate that quest point. The enablers for each quest point are specified in the quest pattern.

While a *SQUEGE* based system is a design tool used during the development cycle of a game to create quest patterns from defined quest points automatically (see section 2.5.2), in this model quest patterns must be defined in terms of quest points manually.

### 3.2.2 Character Modelling

Character modelling is necessary for the interactive storytelling component of the quest generation system. This is because unique and interesting characters are necessary in order to create a narrative around them.

For the current model, character modelling includes NPC's current mental/physical state, relationships between NPCs and the traits of these relationships, as well as character traits of individual NPCs.

#### Mental/Physical State

The mental and physical state of NPCs is modelled according to the model described in Fallon's research [20]. This is a multidimensional approach that describes various aspects of a characters mental/physical state using a *state vector*. This is a number of values, each of which represents a mental or physical state. An example of the states that could be used in a state vector, as well as the possible ranges, are shown in figure 3.1.

| -10.0 | State | +10.0 |
|---|---|---|
| Sad | MOOD | Happy |
| Tired | ENERGY | Rested |
| Hungry | SATEDNESS | Full |

**Table 3.1:** States - Ranges and Meanings [20]

Each NPC has a state vector. The values in the state vector are modified by *state modification vectors*. A state modification vector is made up of a number of values, which serve to modify the core state vector. Each NPC has a state modification vector associated with each activity. The state vector is modified by the associated state modification vector whenever the NPC performs an activity.

Along with modelling the current mental and physical state of an NPC, the state vector system affects the decisions NPCs. Along with the current state vector, each NPC has a *state threshold vector*, which store values that below which are deemed unsatisfactory for that state. For example, if an NPC's current "Energy" state is below the threshold, that NPC is tired.

The *delta* of a state is the amount by which the current value is below the threshold. The *delta sum* is the sum of the deltas for every state in a state vector.

If a state is at an unsatisfactory level, ie. the delta sum is greater than zero, the *activity selection manager* of the NPC selects a task that will increase the level of that state. For example, if an NPC is tired, they may select the "Sleep" activity, as it will raise their "Energy" state value.

This process is summarised in algorithm 1.

### Relationships

Each NPC has a *relationship manager* which maintains their relationships. This includes storing direct family connections (for example, parents, children, current spouse,

**Algorithm 1** Activity Selection

```
if every state in state vector is above threshold then
    apply state modification vector for current activity to state vector
return

for each activity do
    activity state vector = current state vector + activity state modification vector
    calculate activity delta sum for activity state vector

assign activity with minimum activity delta sum as new activity
```

former spouses), as well as a *relationship vector* for every NPC they know. This relationship vector stores the NPC's familiarity with the other NPC, as well as values for a number of *relationship traits* (Table 3.2).

| -10.0 | Relationship Trait | +10.0 |
|---|---|---|
| Enemy | FRIENDSHIP | Close Friend |
| Repulsed By | ATTRACTION | Attracted To |

**Table 3.2:** Relationship Traits - Ranges and Meanings

For every relationship, two relationship vectors are stored (NPC A stores a relationship vector for NPC B, and NPC B stores a relationship vector for NPC A). This allows for two NPCs to have an asynchronous relationship, which can lead to interesting scenarios such as unrequited love (NPC A would have a high attraction value for NPC B, while NPC B would have a low attraction value for NPC A).

### Traits

A number of character traits can be used to represent an NPC's personality and physical characteristics. These traits are chosen by the architect of the given system. An example of traits is shown in table 3.3. These traits are taken from those used in *UNIVERSE* [34] which is set in a soap-opera scenario. New traits can be added or traits can be taken away to reflect the needs of the desired scenario (for example, in a game that involves combat, NPCs may have a "Toughness" trait, which describes how much they are harmed by attacks).

| -10.0 | Trait | +10.0 |
|---|---|---|
| Submissive | AGGRESSIVENESS | Violent |
| Mean | NICENESS | Pleasant |
| Ugly | APPEARANCE | Attractive |
| Prudish | PROMISCUITY | Promiscuous |
| Poor | WEALTH | Rich |
| Stupid | INTELLIGENCE | Smart |

**Table 3.3:** Traits - Ranges and Meanings

### 3.2.3 Drama Management

The drama management system of this model was inspired by that used in *UNIVERSE* [34]. The drama manager chooses an author goals, and selects a plot fragment that both satisfies that author goal and has constraints that are satisfied (see section 2.2.3). The main difference between the drama management in this model and *UNIVERSE* is that in *UNIVERSE* plot fragments contain a number of actions for characters to perform if that plot fragment is activated, while in this system, plot fragments contain a quest for the player to complete, that is activated when the plot fragment is activated.

The components of drama management that will be described in this section are plot fragments, and the drama manager itself.

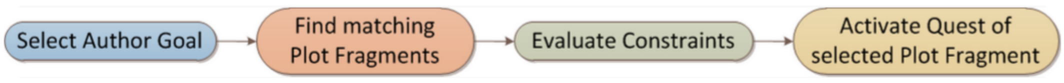

Select Author Goal
Find matching Plot Fragments
Evaluate Constraints
Activate Quest of selected Plot Fragment

**Figure 3.1:** The plot fragment selection process of the drama manager.

### Plot Fragments

Plot fragments are made up of three main components: author goals, constraints, and a quest pattern.

The author goals of a plot fragment are the author goals that the plot fragment satisfies (for example, a plot fragment which include a "Kill NPC" quest would satisfy the "Violence" author goal). The uses and purpose of author goals will be described in more detail in the next section.

The character constraints of a plot fragment are the conditions that must be satisfied in order for the plot fragment to take place. This provides context and relevancy to the quests of the plot fragments. Contraints include the necessary characters, the current mental/physical state of these characters, their traits, and the relationships between them. For example, a plot fragment that involves an NPC A wishing to cheer up NPC B, may have a constraints that NPC A has a relationship with NPC B that has a positive "Friendship" value, NPC A has a high "Niceness" value in their character traits, and NPC B currently has a low "Mood" state in their current state vector. If any of these constraints were not fulfilled, the quest would not be relevant or believable.

The quest pattern is the quest that is activated when the plot fragment is selected. Quests patterns have been described in detail in section 3.2.1.

### Drama Manager

The role of the drama manager is to select author goals, and select plot fragments to satisfy these goals.

Author goals can be general (for example, to cause violence, for the player to perform good deeds), or can be specific (for example, to cause the death of a specific character).

**Algorithm 2** Plot Fragment Selection

```
choose an author goal

for each plot fragment do
    if current plot fragment
    satisfies author goal and constraints are satisfied then
        add plot fragment to potential plot fragments

activate a potential plot fragment
```

Author goals can be used to control the trajectory of the game's dramatic arc. However, actual management of drama and tension in the narrative through selection of author goals was decided not a priority for this implementation, and was instead left as a possible avenue for future expansion.

Once an author goal has been chosen, an appropriate plot fragment must be selected. Firstly, a list is compiled of all available plot fragments that satisfy the current author goal. For each plot fragment in this list, the constraints are evaluated. Then a final plot fragment is selected from the plot fragments whose constraints are satisfied. Finally, the quest of this plot fragment is activated and assigned to the appropriate quest giver. This process is summarised in algorithm 2 and figure 3.1.

## 3.3 Illustrating the Model

In this section, the model that was outlined in the previous section will be illustrated using an example.

### 3.3.1 Example Plot Fragment: The Jealous Suitor

"The Jealous Suitor" is an example plot fragment designed to demonstrate the quest generation.

This plot fragment is designed to convey a narrative of a character (the eponymous "suitor") who has an unrequited infatuation with another character (the "love interest"). Due to a period of sadness, as well as their aggressive personality, the suitor decides to hire a hitman to kill their love interest's spouse (the "victim"). This hitman is the player.

The specific components of this plot fragment are detailed here.

### Author Goals

The general author goals for this plot fragment are evil, death, and violence. The **evil** author goal reflects the player performing evil deed (in this case, the murder of an innocent NPC). This could be useful if the game involves a moral element, such as that used in *Fable* [6]. The **death** and **violence** author goals reflect the attack and murder of an NPC.

The specific author goals for this plot fragment are the death of the victim NPC. This author goal could be used if the main narrative of the game required a character to be dead, or for the purposes of population control.

### Constraints

The constraints of this plot fragment are detailed in this section, and summarised in figure 3.2.

There are three characters necessary for this plot fragment: the suitor, the love interest, and the victim.

**Mental/physical state constraints:**
The **suitor** must be sad (ie. have a mood state vector value below the satisfactory threshold).

**Character traits constraints:**
The **suitor** must be aggressive (in this implementation, an NPC is deemed to be aggressive if their aggressiveness character trait is above 5).

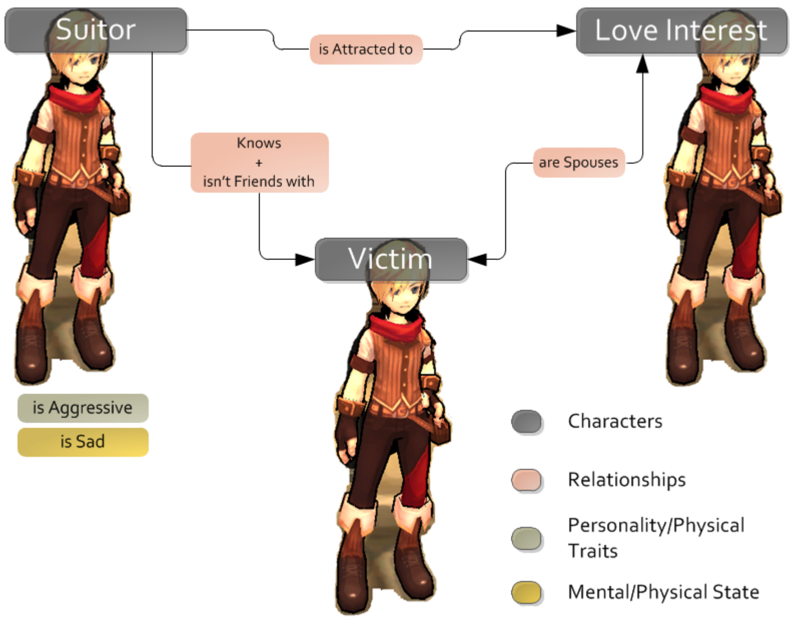


**Figure 3.2:** The character constraints of the "Jealous Suitor" subplot.

**Relationship constraints:**
The **suitor** must be attracted to the **love interest** (the suitor must have a relationship vector stored for the love interest, and this relationship vector must have a high attraction value). The **love interest** and the **victim** must be spouses. The **suitor** must know the **victem**, but they must not be friends (the suitor must have a relationship vector stored for the victim, but this relationship vector must not have a friendship value above neutral).

## Quest

The quest pattern for this plot fragment has the **suitor** as the quest giver, and is made from the following quest points:
Kill **victim**. Talk to **suitor**.

## 3.4 Procedural Content Generation

Procedural content generation is necessary for the generation of NPCs, as in a large scaled open world game, a large number of NPCs are necessary in order to populated it, and it would take a huge amount of time and resources to manually design each NPC.

**Character Traits and Relationships** Traits, relationships, and relationship traits can be randomly generated. This is advantageous as it can generate a large number of NPCs with unique characteristic and existing relationships quickly.

**Mental/Physical State** Each NPC's initial state vector can be randomly generated, as can their state threshold vector, which enables NPCs to have varying levels of satisfaction for states (for examples, some NPCs may have a higher "Satedness" threshold, and so will need to eat more food to feel full than others).

State modification vectors can also be randomly generated for each activity, and for each NPC. This enables NPCs to have different reactions to different activities, to reflect different personalities (for example, one NPC's mood may increase while fishing, implying they enjoy fishing, while another NPC's mood may decrease while fishing, implying they dislike fishing). However, it is possible to put restrictions in place on the modification of specific states by specific activities. For this model, the "Satedness" state must receive a positive modification from the "Eat" task, while it must receive a neutral or negative modification from all other activities. Similarly, the "Energy" state must receive a positive modification from the "Sleep" and "Eat" tasks, and would generally receive a neutral or negative modification by all other tasks. These restrictions must be put in place to maintain believability (for example, it would not be believable for an NPC's "Satedness" to decreased while they are eating).

## 3.5 Conclusion

In this chapter, the main design of the model has been described in detail, including character modelling, quest generation, and drama management. The opportunities for procedural content generation have also been identified.

In the next chapter, the specifics of the implementation of the prototype of this model will be explain.

# Chapter 4

# Implementation

In this chapter, the implementation of the prototype of the model will be described.

## 4.1 Platform and Existing Code Base

This section will outline the platform on which the prototype will be implemented, as well as the existing code base that was used.

### 4.1.1 Unity Game Engine

The Unity game engine [62] is a framework that is well suited to the quick generation of prototypes. Code is written in C#, while there is also an extensive editor that allows the user to quickly drag and drop components in a scene.

### 4.1.2 Existing Code Base

In order to maximise time spent on the model, it was decided to work on the existing code base from Tiernán McNulty's dissertation [39]. This model included a "village" environment, populated with items, buildings, and NPCs already in place.

McNulty's memory model involves each NPC having memories. They can create memories when they encounter new objects and characters, and memories fade over time. The model also includes a goals system. Each NPC is assigned with a routine of high level goals (for example, eat at 12pm, work at 1pm, drink at 5pm, etc.). These high level goals are split into smaller intermediate goals to be completed (for example, the "sleep" high level goal would be split into: go to bed, sleep). The combination of the memory model and goals model creates a village of NPCs that are autonomous in their actions. This is useful as a baseline for this prototype as it shows how the proposed model could be used with an AI system that controls the behaviour of NPCs.

## 4.2 Character Modelling

The design for the character modelling used in this implementation is outlined in section 3.2.2.

### Mental/Physical State

State vectors, state threshold vectors, and state modification vectors for each activity, are randomly generated for each NPC. The specific states used for this implementation are listed in table 3.1.

State modification vectors are only assigned to final activities, such as sleeping, eating and working, and not interim activities, such as walking to a location or talking to an encountered NPC. This is to avoid scenarios where the journey necessary to partake in an activity could have unwanted and unplanned effects on the NPCs state vector. For example, an NPC is tired and so selects the "Sleep" activity. However, the "Walking" activity required to get to his bed decreases his energy further, and so it seems that sleep is not the best activity. This problem could be solved in more detail using an AI planner such as *STRIPS* (see section 2.3.4), however this is outside of the scope of the dissertation, and so the walking activity is simply not taken into account when generating state modification vectors.

For activity selection, each NPC is assigned an *update frequency.* This is the frequency at which the NPC checks their current state vector, and decides whether to select a new activity. NPCs are also assigned a *give up time.* This is the amount of time an NPC will try to accomplish their desired activity, before giving up and selecting the next best activity. This prevents an NPC getting stuck in a situation where their desired activity is impossible (for example, they desire to work at the blacksmiths, but there is not blacksmithing hammer available).

### Traits

The character traits for each NPC are stored as a series of randomly generated values between -10 and 10.

The traits used for this prototype are outlined taken from the *UNIVERSE* model, and are listed in table 3.3.

### Relationships

It was originally planned to create a relationship system that took advantage of the memory cue system in place in the McNulty's original model, such that when an NPC encountered a new NPC it would form a new relationship, meeting that NPC again would reinforce the relationship, and not seeing the NPC for a period of time would decay the relationship. However, it was decided that a relationship model of this complexity was outside the scope of this dissertation, and so a simpler model was created.

In this simple model, relationships between NPCs are generated randomly at the start of the program, and do not change over time. Each relationship has a random familiarity value between 0 and 10, and a relationship vector with random values between -10 and 10 for each relationship trait. The relationship traits used for this implementation are listed in table 3.2. Each relationship formed also has a chance to be a *connection*, ie. to be a parent, child, spouse, or former spouse relationship.

## 4.3 Quest System

The model for the quest system is described in section 3.2.1.

### Quest Completion Data

In order to check if quests have been completed, whenever a player performs a significant action (such as killing an NPC, or picking up an item), *quest completion data* is created with the details of this action, and is checked against the player's current active quests to check if that action has fulfilled a quest objective.

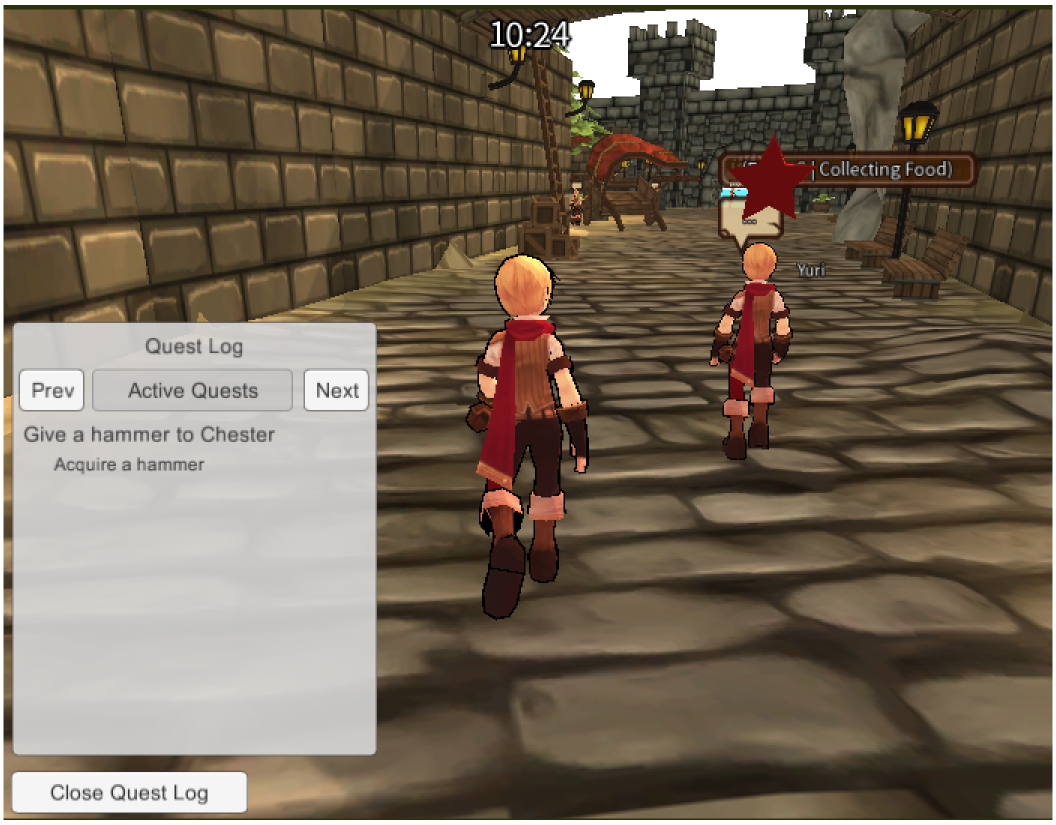
### Quest Points

Each quest point has the following main functionality:

- *Attempt Activation*: This involves checking if all the enablers for this quest point are completed, and it they are, activating the quest point.
- *Attempt Completion*: This involves checking *quest completion data* against this quest point to check if the objective of the quest point has been achieved.
- *Attempt Immediate Completion*: When a quest point is first activated, it is checked for immediate completion. Immediate completion is only possible for certain quest points (for example, the "Acquire Item" quest point would be immediately completed if the player already had the item in their inventory).
- *Ensure Possibility*: Some quest points may become impossible to complete (for example, the "Talk to NPC" quest point would no longer be possible if the target NPC has died). Therefore, active quest points must be periodically checked to ensure that they are still possible to complete, and if not, the quest is failed.

The following quest points are currently implemented in the model prototype:

- Talk to NPC



Give a hammer to Chester

Acquire a hammer

**Figure 4.1:** Open quest log displaying an active quest.

- Acquire Item
- Give Item to NPC
- Kill NPC

### Quest Patterns

Each quest pattern has functionality to cycle through all its quest points and carry out the quest point functionality detailed in the *Quest Point* section.

The following quest patterns (made up of the listed quest points) are currently implemented in the model prototype:

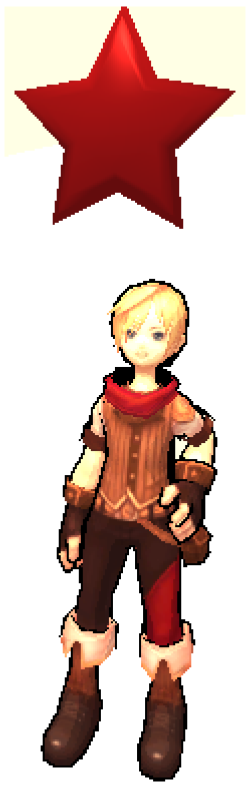


**Figure 4.2:** Quest star to indicate the NPC has a quest for the player.

- Give Item to NPC
  - Acquire Item
  - Give Item to NPC
- Kill NPC and Report to NPC
  - Kill NPC
  - Talk to NPC

### Quest Manager

The quest manager is responsible for managing all active quest patterns that the player has available. It takes in *quest completion data* and cycles it through all active quest patterns to implement the functionality listed in the quest pattern section.

#### User Interface

In order to keep track of their active quests, the player has a quest log that they can use to view each of their active quests, as well as the active quest points of these quests (Figure 4.1).

In order to facilitate this quest log, each quest point and quest pattern has a description attached to it.

#### Quest Givers

When an NPC has an available quest for the player, they become a *quest giver*, and have a visual cue (Figure 4.2). The player is then able to talk to the NPC in order to add that quest to their active quests.

## 4.4 Plot Fragments

Plot fragments are created manually using scripts. When a plot fragment is created, the following must be specified:

- *Author Goals*: Each plot fragment must store a list of the author goals that it fulfils.
- *Plot Details*: The *plot details* of a plot fragment contain the parameters specific to the quest of the plot fragment, such as the necessary characters, items, and locations.
- *Constraints*: The constraints for each plot fragment must be specified, in terms of character traits, relationships and relationship traits, and the mental/physical state of the characters involved.

- *Quest Pattern*: A quest pattern for the player to complete must be set for each plot fragment. An existing quest pattern can be used, or a new quest pattern can be created from the existing quest points.

Plot fragments have the following functionality:

- *Compare Author Goals*: This involves checking if the the plot fragment fulfils the current author goal selected by the drama manager.
- *Check Constraints Fulfilled*: If the plot fragment fulfils the current author goal, it is checked to see if there the constraints can be fulfilled. Any combination of characters that fulfil these constraints are stored as *possible plot details*. If there are more than one possible plot details, one is chosen at random to be the actual plot details that are used in the quest pattern for the plot fragment.

## 4.5 Drama Management

As discussed in section 3.2.3, controlling author goal selection in order to maintain a desired dramatic arc was not implemented in this model. Additionally, author goals have been restricted to be general, describing the general theme for the desired plot fragment, rather than specific, such killing specific NPCs or for the player to visit specific locations. For the purposes of this prototype, the drama manager randomly selects from a list of general author goals at a specified time interval. The possible author goals in this prototype are:

- Good
- Evil
- Violence
- Death

Once the current author goal has been chosen, a plot fragment must be selected. This involves comparing author goals, and then checking the constraints of plot fragments, as described in section 4.4. If more than one suitable plot fragment exists, a random plot fragment is selected. The quest of the selected plot fragment is then added to the appropriate quest giver (if they do not already have a quest).

## 4.6 Conclusion

This chapter has outlined the details of the implementation of the prototype. In the next chapter, this prototype, as well as the model outlined in the previous chapter, will evaluated on how well they satisfy the objectives outlined in chapter 1.

# Chapter 5

# Evaluation

In this chapter, the model, and the implementation of the model, will be evaluated, according to how well the objectives outlined in chapter 1 have been satisfied.

## 5.1 Character Modelling

### 5.1.1 Mental/Physical State

Fallon's state vector model [20] is a promising multidimensional approach to representing an NPC's current mental and physical state. The use of personalised state threshold vectors and state modification vectors for each individual NPC helps to convincingly create unique NPCs with individual preferences and interests.

Using state vectors for activity selection, rather than following a routine (as seen in McNulty's model [39]), allows for a visible representation of each NPCs uniqueness through their behaviours. It also helps to combat performance dips that may occur at certain times of the day (for example, at lunch or dinner time) when the vast majority of NPCs attempt to change their activities simultaneously.

A potential weakness of this approach, however, is that it does lead to NPCs performing activities at unconventional times, for example sleeping during the day. This could be acceptable for a small number of NPCs, however it would be far more believable for the majority of NPCs to perform certain activities such as sleeping or eating at more conventional times. This could potentially be achieved by attaching a timing multiplier to state modification vectors. Such an addition would allow for the introduction of more convincing temporal behaviour among NPCs.

### 5.1.2 Relationships and Relationship Traits

The relationship system (described in section 3.2.2) is extremely effective at representing the relationships between NPCs. The use of familiarity and relationship vectors allows for multidimensional relationships to be stored, which allows for more complex and believable interactions between NPCs. Having two relationship vectors stored per association allows for asynchronous relationships to be represented, such as unrequited love, which can lead to the generation of far more believable and engaging stories.

One limitation of the relationship vector implementation, however, is that it does not evolve over time, and relationships are not currently reflected in the actions of NPCs, though this is merely a limitation of the current implementation, rather than of the model itself. The relationship vector model could be used to drive and influence NPC actions and behaviour in the future, but such work is beyond the scope of this dissertation.

The relationship vector system as implemented serves its main purpose of providing character constraints on which plot fragments can be based.

### 5.1.3 Character Traits

The character traits system is a an effective way of representing multiple different aspects of an NPC's personality, physical appearance, etc. As with the relationship system, its current implementation is somewhat limited in that character traits have no effect on an NPC's actions, but it does provide a basic model that can be used as character constraints for plot fragments.

### 5.1.4 Procedural Generation of Characters

Support is provided for procedural generation throughout the character modelling system. The nature of the NPC model described lends itself very well to being partially or completely procedurally generated.

**Mental/Physical State** The state vector for each NPC can be randomly generated at the start of the program. This provides each NPC with a varying initial mood, energy level, etc. Additionally, the state threshold vector can also be randomly generated. This gives unique tolerances for certain states. Finally, the state modification vectors for each activity can be randomly generated for each NPC. This generates NPCs that react to different activities in different ways, and develop preferences for certain activities, which is reflected in their actions.

**Relationships** At the start of the program, relationships between NPCs can be randomly generated. This includes random familiarity, as well as random relationship vectors. Additionally, connections such as spouse-spouse, parent-child, etc. can be randomly generated. This automatically generates a family tree within the game world.

**Character Traits** The character traits for each NPC can also be randomly generated, giving each NPC a unique personality and physical characteristics.

## 5.2 Procedural Generation of Quests using Interactive Storytelling

### 5.2.1 Quest System

As previously outlined in section 3.2.1, the quest system is based on the research of *SQUEGE* [48] and is an effective way of creating quests of any size using single objectives. Patterns can be created from any number of quest points in any order, can be linear or non-linear, involve any number of NPCs, etc. It is worth noting however that the current implementation of the system does not allow for multi-solution levels. Such an extension was explored in *SQUEGE* [48], and a similar solution could be applied to this system in future work.

### 5.2.2 Plot Fragments

As described in section 3.2.3, the plot fragment and drama management model is based on the research of UNIVERSE [34]. The plot fragment model is what distinguishes this dissertation from the work of *Skyrim*'s *Radiant Story* [60][45]. Rather than just quests being generated independently, they are generated as part of a plot fragment.

The plot fragment encapsulates author goals that the quest will satisfy, which allows generation of quests congruous to the narrative of the game. The plot fragment also include constraints, which act as preconditions for the quest to take place. This gives quests relevance in the game world. Currently the constraints are limited to character constraints, but this could be extended in the future.

### 5.2.3 Drama Management

Using a drama manager for quest generation allows for the state of the game to be monitored, and for the pacing and narrative of the game to be controlled using author goals. These author goals could be used to mirror the main narrative of the game through the various side quests. The frequency at which quests are generated, as well as their relative intensity, can be altered to achieve the desired flow and emotional arc of the game. Author goals could also be used to affect other elements of the game, such as population control.

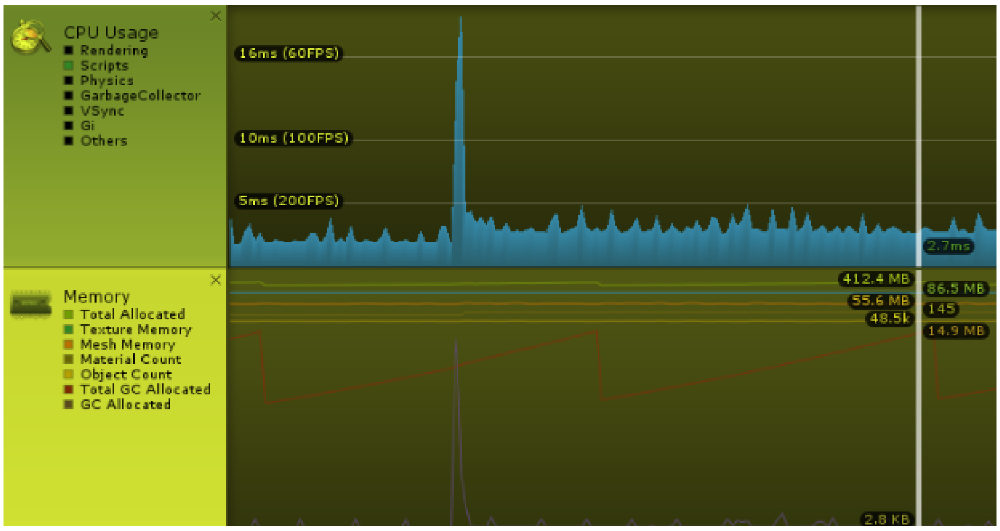


**Figure 5.1:** Unity's profiler showing McNulty's implementation. The performance shows a computation spike, which adversely affects the frame rate. This corresponds to 10am in-game time, when all NPCs go and eat breakfast. Other than this, performance is steady.

## 5.3 Extensibility and Ability to Integrate with Other Systems

The described model has been created with extensibility in mind. The quest system can be extended by adding more quest points, or by assembling them into patterns. Plot fragments can also be created quickly and easily. Technically, these components have been created generically, so the creation of new components does not require any additional developmental overhead.

The core quest generation model has already been integrated with a number of different systems in this implementation; McNulty's existing memory model [39], and Fallon's state vector model [20]. A logical next step would be to implement it with O' Connor's more sophisticated relationship model [46], however this was not possible due to time constraints.

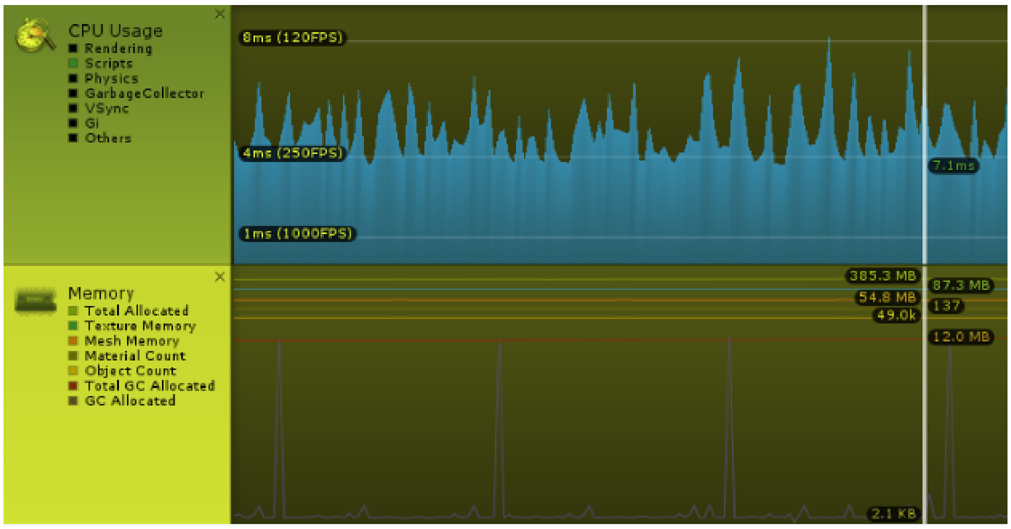


**Figure 5.2:** A performance snapshot of the implementation taken in the Unity profiler. There is no single large performance spike, as NPCs do not have specific routines to follow and so do not all change activities at once. However, performance is less steady overall, as NPCs are changing tasks more frequently and at irregular intervals.

## 5.4 Effcient Implementation

The current implementation is a prototype of how the model could be used, so and so is not a definitive reflection on the model's potential efficiency were it to be used for larger scale, more sophisticated projects. However, conclusions that have been drawn thus far do strongly indicate that the model does not adversely affect performance by a noticeable level.

Due to McNulty's model using a routine system, where NPCs would change tasks at specific times during the day, performance would dip significantly when the majority of NPCs would all change their current activity at once (Figure. 5.1). Migrating from this routine-based system to activity selection based on state vectors results in more frequent, but smaller and manageable performance spikes, as NPCs change tasks independently and more frequently (Figure 5.2). This is preferred over one single large spike, as the smaller spikes still keep the frame rate within an acceptable range, while the previous large spike caused noticeable lag when it occurred.

# Chapter 6

# Conclusion

This chapter draws conclusions on the dissertation by outlining the main contributions, the limitations of the existing implementation, as well as the possible future work to address these limitations, and final thoughts.

## 6.1 Main Contributions

The use of interactive storytelling techniques is an effective approach to procedural side quest generation. Similar to the commercial example of *Skyrim*'s *Radiant Story* [45](see section 2.5.2), it is capable of generating a theoretically limitless number of side quests, a task which would be impossible were they to be manually scripted. However, while quests generated with *Radiant Story* are unrelated to the state of the game world and the current state of the narrative, the quests generated with this current system are given purpose and relevancy through the use of author goals and constraints. The quest system used of patterns and points (see section 3.2.1) allows quests to be created quickly and easily during development.

Although the character modelling component of the system was not the main focus of this dissertation, and was merely a necessity for the interactive storytelling aspect of the main quest generation model to create a narrative, it does provide a method to create a large number of unique and interesting NPCs quickly. The mental/physical state modelling serves as a near-full implementation of Fallon's research [20], in a more general form than in his own dissertation, as his prototype was implemented in *Skyrim*'s modding engine. The use of this method for NPC activity selection also addresses the "Future Work" in McNulty's dissertation, which outlined the need for a more flexible model of activity selection rather than relying on set routines [39].

An efficient implementation of the model has been created in the form of a high fidelity prototype. Profiling the prototype has shown that the implementation is light-weight and should not have a significant impact on performance. This is of importance because of the low computation and memory budget typically afforded to AI in commercial games [42].

This dissertation is based on and integrates research from both interactive storytelling and quest generation, two areas which when merged show potential for the procedural generation of interesting quests that has not been largely explored to this date. It is hoped that this dissertation can provide sources and foundation for future research into this area.

## 6.2 Limitations & Future Work

In this section, some limitations of the model will be outlined, as well as future work that could address these limitations.

### 6.2.1 Limited Prototype

As the current implementation is a prototype and has various limitations, the system's potential uses and functionality have not yet been fully explored. The prototype has currently only four fully implemented quest points and two plot fragments, and while in theory it should be easily able to support a larger number, further experimentation is required to monitor the behaviour of the model, as well as the effect of a large number of quest components and plot fragments on efficiency.

### 6.2.2 Integration

The current system has been integrated successfully with McNulty's memory model [39] and Fallon's state vector model [20]. It was also hoped to integrate with O' Connor's relationship model [46] and Cullen's environment and temporal model [15], however due to time constraints this was not possible.

Integration with a more sophisticated author goal selection system, such as the drama managers used in *Façade* [2] and *Left 4 Dead* [61], would also be useful to showcase how the model can help with the pacing and narrative of a game.

### 6.2.3 Events System

McNulty's model [39] included an events system, that affected the memories of the NPCs. It was originally hoped to integrate this system with the actions of the player, so that NPCs would be aware of the player and their relationship with the player would be affected based on their interactions. However, this, as well as a more sophisticated relationship model and traits model that change over time, was decided to be outside of the scope of the dissertation, and is left to future work.

### 6.2.4 UI/Scripting Support for Non-Programmers

While new quest points, patterns, and plot fragments can be created quickly and easily in the model, they are must be specified in code. This requires the user to be familiar with programming. Typically, writers and designers would be responsible for quest design, so requiring a programmer to implement these specifications is a waste of resources.

Developing a user interface or simple scripting system that could be used by writers or designers to create quests and plot fragments themselves could further save development time and resources.

## 6.3 Final Thoughts

As open world role playing games grow, both in popularity, and in size, the desire and necessity for an alternative to manual side quest generation increases. Procedural side quest generation has been used commercially in *Skyrim* [60], however the resulting quests were repetitive and disliked among many players.

The use of interactive storytelling has been limited in commercial games. Drama management has been used to great effect to control pacing and difficulty (as used in *Far Cry 2* [43] and *Left 4 Dead* [61] [33]), however interactive narrative is rare. Games that rely on a narrative experience (such as *The Walking Dead* [25]) often rely on a few yes-or-no choices from the player, which don't have more than a cosmetic effect on the narrative or gameplay.

This dissertation has proposed a model of using concepts from interactive storytelling to improve the side quest generation process. This is an area that has been largely unexplored to date. It is hoped that this dissertation demonstrates that using interactive storytelling in the procedural generation of quests is a viable option for the development of games in the future to create enjoyable interactive experiences for players.

# Bibliography

[1] John Abercrombie. Bringing BrioShock Infinte's Elizabeth to Life: An AI Devlopment Postmortem. `http://www.gdcvault.com/play/1020831/Bringing-BioShock-Infinite-s-Elizabeth`, 3 2014 (Accessed May 14, 2015). Talk at Game Developer's Conference AI Summit.

[2] Procedural Arts. Façade. [Download], 2005.

[3] Bioware. Baldur's Gate. [CD-ROM, Download], 1998.

[4] Bioware. Neverwinter Nights. [CD-ROM, Download], 2002.

[5] Christopher Booker. *The Seven Basic Plots: Why We Tell Stories*. Continuum, 2005.

[6] Big Blue Box. Fable. [CD-ROM, DVD], 2004.

[7] Mark Brockington. Level-Of-Detail AI for a Large Role-Playing Game. In Steve Rabin, editor, *AI Game Programming Wisdom*, pages 419–425. Charles River Media, Inc., Rockland, MA, USA, 2002.

[8] Mat Buckland. *Programming Game AI By Example (Wordware Game Developers Library)*. Jones & Bartlett Learning, 2004.

[9] Joseph Campbell. *The Hero with a Thousand Faces (The Collected Works of Joseph Campbell)*. New World Library, 2008.

[10] Capcom. Resident Evil 4. [DVD, Download], 2005.

[11] MO Cavazza, Fred Charles, and Steven J Mead. Character-based interactive storytelling. *IEEE Intelligent systems*, 2002.

[12] Alex J. Champandard. 10 Reasons the Age of Finite State Machines is Over. `http://aigamedev.com/open/article/fsm-age-is-over/`, 2007 (Accessed May 19, 2015.

[13] Chris Conway. Goal-Oriented Action Planning: Ten Years Old and No Fear! `http://www.gdcvault.com/play/1022020/Goal-Oriented-Action-Planning-Ten`, 20015 (Accessed June 10, 2015). Talk at Game Developer's Conference.

[14] Valve Corporation. Dota 2. [Download], 2013.

[15] Tony Cullen. Modelling Environmental and Temporal Factors on Background Characters in Open World Games. Master's thesis, University of Dublin, Trinity College, Ireland, 2015.

[16] Maria Cutumisu and Duane Szafron. An Architecture for Game Behavior AI: Behavior Multi-Queues. In Christian Darken and G. Michael Youngblood, editors, *AIIDE*. The AAAI Press, 2009.

[17] Naughty Dog. The Last of Us. [DVD, Download], 2013.

[18] Crystal Dynamics. Tomb Raider. [DVD, Download], 2013.

[19] L. Egri. *The Art of Dramatic Writing: Its Basis in the Creative Interpretation of Human Motives*. Touchstone, 1972.

[20] John Fallon. Believable Behaviour of Background Characters in Open World Games. Master's thesis, University of Dublin, Trinity College, Ireland, 2013.

[21] Richard E. Fikes and Nils J. Nilsson. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. In *Proceedings of the 2Nd International Joint Conference on Artificial Intelligence*, IJCAI'71, pages 608–620, San Francisco, CA, USA, 1971. Morgan Kaufmann Publishers Inc.

[22] Gustav Freytag. *Die Technik des Dramas*. 1863. (in German).

[23] Epic Games. Unreal Tournament 2004. [CD-ROM, DVD, Download], 2004.

[24] Irrational Games. BioShock Infinite. [DVD, Download], 2013.

[25] Telltale Games. The Walking Dead: Season One. [DVD, Download], 2012.

[26] M. Gonzalo and M.A. Gómez-Martín. *Artificial Intelligence for Computer Games*. SpringerLink : Bücher. Springer New York, 2011.

[27] Justin Haywald. Roam if you want to. `http://www.gamespot.com/articles/the-witcher-3-is-an-open-world-with-no-loading-tim/1100-6426896/`, 2015 (Accessed May 16, 2015). Interview with *The Witcher 3* developer.

[28] Philip Hingston. The 2K BotPrize. `http://botprize.org/`, 2013 (Accessed May 5, 2015).

[29] Clint Hocking. Ludonarrative Dissonance in Bioshock. `http://clicknothing.typepad.com/click_nothing/2007/10/ludonarrative-d.html`, 2007 (Accessed August 28, 2015).

[30] IGN. List of Side Quests in The Witcher 3. `http://ie.ign.com/wikis/the-witcher-3-wild-hunt/Side_Quests`, 2015 (Accessed August 28, 2015).

[31] IO Interactive. Hitman: Bloody Money. [CD-ROM, DVD, Download], 2006.

[32] Konami Computer Entertainment Japan. Metal Gear Solid. [CD-ROM], 1998.

[33] Daniel Kline, Micheal Mateas, and Emily Short. An AI Assist to Interactive Storytelling: Bringing Interactive Storytelling to Industry. `http://www.gdcvault.com/play/1013459/Three-States-and-a-Plan`, 2010 (Accessed June 10, 2015). Talk at Game Developer's Conference, AI Summit.

[34] Michael Lebowitz. Creating characters in a story-telling universe. *Poetics*, 13(3):171–194, 1984.

[35] Tim Lovett. The 15 Most Annoying Video Game Characters (From Otherwise Great Games). `http://www.cracked.com/article_15902_the-15-most-annoying-video-game-characters-from-otherwise-great-games.html`, 2008 (Accessed May 15, 2015).

[36] Aaron Bryan Loyall. *Believable Agents: Building Interactive Personalities*. PhD thesis, Pittsburgh, PA, USA, 1997.

[37] George Lucas. Star Wars, 1977.

[38] Robert R McCrae and Oliver P John. An introduction to the five-factor model and its applications. *Journal of personality*, 60(2):175–215, 1992.

[39] Tiernan McNulty. Residual Memory for Background Characers in Complex Enivronments. Master's thesis, University of Dublin, Trinity College, Ireland, 2014.

[40] James R. Meehan. TALE-SPIN, An Interactive Program that Writes Stories. In *In Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pages 91–98, 1977.

[41] Herman Melville. *Moby-Dick*. Richard Bentley, 1851.

[42] I. Millington and J. Funge. *Artificial Intelligence for Games*. CRC Press, 2009.

[43] Ubisoft Montreal. Far Cry 2. [DVD, Download], 2008.

[44] Ubisoft Montreal. Assassin's Creed IV: Black Flag. [DVD, Download], 2013.

[45] Bruce Nesmith. Radiant Story. `https://www.youtube.com/watch?v=Ou6SB8dWKjw`, 2012 (Accessed May 18, 2015). Talk at Game Design Expo, Vancouver Film School.

[46] Brendan O' Connor. A Relationship Model for Believable Social Dynamics of Characters in Games. Master's thesis, University of Dublin, Trinity College, Ireland, 2015.

[47] Curtis Onuczko, Duane Szafron, and Jonathan Schaeffer. Stop Getting Side-Tracked by Side-Quests. In S. Rabin, editor, *AI Game Programming Wisdom 4*, AI Game Programming Wisdom, pages 513–527. Charles River Media, 2014.

[48] Curtis Onuczko, Duane Szafron, Jonathan Schaeffer, Maria Cutumisu, Jeff Siegel, Kevin Waugh, and Allan Schumacher. A Demonstration of SQUEGE: A CRPG

Sub-Quest Generator. In *Proceedings of the Third Artificial Intelligence and Interactive Digital Entertainment Conference, June 6-8, 2007, Stanford, California, USA.*, pages 110–111, 2007.

[49] Jeff Orkin. Three States and a Plan: The AI of F.E.A.R. [AUDIO ONLY]. `http://www.gdcvault.com/play/1013282/Three-States-and-a-Plan`, 2006 (Accessed June 10, 2015). Talk at Game Developer's Conference.

[50] Jeff Orkin. Three States and a Plan: The AI of F.E.A.R. [SLIDES ONLY]. `http://www.gdcvault.com/play/1013459/Three-States-and-a-Plan`, 2006 (Accessed June 10, 2015). Talk at Game Developer's Conference.

[51] D. Pizzi, J.-L. Lugrin, A. Whittaker, and M. Cavazza. Automatic Generation of Game Level Solutions as Storyboards. *Computational Intelligence and AI in Games, IEEE Transactions on*, 2(3):149–161, Sept 2010.

[52] Pogee777. Move it Dammit! `http://www.nexusmods.com/skyrim/mods/4020/?`, 2011 (Accessed May 14, 2015). User mod created for Skyrim.

[53] Monolith Productions. F.E.A.R. First Encounter Assault Recon. [DVD, Download], 2005.

[54] V.IA. Propp. *Morphology of the Folk Tale.* University of Texas Press, 1968.

[55] Rare. GoldenEye 007. [CD-ROM], 1997.

[56] CD Projekt RED. The Witcher 3: Wild Hunt. [DVD, Blu-Ray Disc, Download], 2015.

[57] Emily Short. Galatea. [Web, Download], 2000.

[58] Chris Simpson. Behavior trees for AI: How they work, 2014 (Accessed May 19, 2015).

[59] The Indie Stone. Project Zomboid. [Download], 2013 (Early Access).

[60] Bethesda Game Studios. The Elder Scrolls V: Skyrim. [DVD, Download], 2011.

[61] Turtle Rock Studios. Left 4 Dead. [CD-ROM, DVD, Download], 2008.

[62] Unity Technologies. Unity Game Engine, v5.1.2f1. [Download], 2015.

[63] Frank Thomas and Ollie Johnston. *The illusion of life : Disney animation.* Disney Editions, New York, 1981.

[64] Elder Scrolls Wiki. List of Side Quests in Skyrim. `http://elderscrolls.wikia.com/wiki/Side_Quests_(Skyrim)`, 2011 (Accessed August 28, 2015).