October 2014

Development of an Emergent Narrative Generation Architecture for Videogames

Nicholas A. Schudlo
The University of Western Ontario

Supervisor
Michael James Katchabaw
The University of Western Ontario

Graduate Program in Computer Science

A thesis submitted in partial fulfillment of the requirements for the degree in Master of Science

DEVELOPMENT OF AN EMERGENT NARRATIVE GENERATION ARCHITECTURE FOR VIDEOGAMES

Thesis format: Monograph

by

Nicholas Schudlo

Graduate Program in Computer Science

A thesis submitted in partial fulfillment
of the requirements for the degree of
Masters of Science

The School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

Abstract

Emergent narratives can enhance a video gaming experience. However, the use of this storytelling form in videogames is in its infancy. Many existing emergent narrative generation systems are limited in authorial creative control, story world flexibility, user freedom and/or general storyline maintenance. In this thesis, we designed a robust architecture for generating emergent narratives to use in videogames, and tested the architecture in a prototype game simulation. The architecture continuously presents pre-written plot fragments with fulfilled preconditions, providing authorial control over the story's components and general direction. The user navigates the plotlines and shapes the story through their unrestrained decisions. Architectural components monitor these actions and respond appropriately to ensure a cohesive story. Yet, the narrative generation framework remains separate from specific Game World mechanics, maintaining compatibility with any story world. The proposed architecture offers flexible emergent narrative generation and can provide a framework for future emergent videogame development.

Keywords

Narrative generation, video game development, emergent storytelling, emergent narrative.

Acknowledgments

First and foremost, I would like to express sincere gratitude to my supervisor, Dr. Michael Katchabaw, for his dedication and support throughout my masters. Not only did he guide and support my research, but also encouraged and motivated me to persevere when I felt doubtful or discouraged.

Gratitude is also expressed to my committee members, Dr. Hanan Lutfiyya and Dr. Mahmoud El-Sakka, and external examiner, Dr. Jacquelyn Burkell, for their helpful comments and suggestions.

I would also like to thank Larissa Schudlo for her editorial contributions as well as helpful guidance during the writing of this thesis.

Table of Contents

List of Tables

List of Figures

List of Appendices

Chapter 1

1 Introduction

1.1 Motivation

Videogames are a popular form of entertainment. Unlike books and film which are considered mature forms of media, videogames are relatively new and are still in a state of evolution[1]. This includes the narrative aspect of videogames. This powerful medium for storytelling offers unique opportunities to authors that are unavailable with other traditional methods of narration [1]–[3]. Narratives conveyed through, for example, books, television and film, are often linear, having a highly structured and predetermined storyline, and require little or no input from the audience. Contrarily, videogames have the advantage of telling a story in an interactive manner. However, games that follow a single path still limit the influence the user has on the overall plotline. This means that the same basic narrative is told each time anyone plays through the game, regardless of a player's specific actions. An emergent narrative is one that grows naturally, or *emerges*, from an individual's particular actions. Because the course of the narrative is influenced by one's actions, each telling of the narrative yields a unique outcome founded in the user's specific decisions. In comparison to a linear approach, an emergent-based narrative heightens a videogame's level of interactivity and permits highly individualized storylines. This allows each player to foster a greater, more exclusive connection to the characters in the game [1], [4]. Furthermore, each play through the game provides a different experience, further increasing its playability and value [1]. Although this method of dynamic storytelling has great potential to enrich and reform the gaming experience, employing emergent storytelling in videogames has many challenges, and its development is still very much in its infancy. This thesis will explore the use of emergent narratives in videogames and the developmental architecture needed to integrate such narratives into games.

Despite rapid advances in videogames, the use of emergent storytelling in game development has been limited largely due to the way in which the narrative portions of

videogames are typically developed. When a videogame is in development, adding many different storylines can become expensive and time consuming for the developers. Therefore, it is usually easiest to cut down on story elements, which cuts down on production costs and allows the game to release earlier [1]. Because videogames are a relatively new form of storytelling, the optimal method for writing a story that fits this particular medium is still not well understood. Many studios approach storyline generation for their videogames from a non-interactive perspective, much like a filmmaker approaches the storyline of a film[3]. Player interaction with the plot is not intentionally incorporated into the narrative, and even the slightest deviation in the story can induce significant flaws in the intended narrative. Furthermore, videogames employing linear narratives are not as developmentally intensive as those employing an emergent narrative, since the storyline is highly constrained to a single plotline that permits minimal deviations. A dynamic storyline that supports substantial influence from the player is much more difficult to test, as there are exponentially more outcomes the narrative can have. The traditional way in which storylines are created and tested for videogames must be adapted if the narratives are to support any form of player influence on the plotline [1].

In addition to the challenges associated with storyline creation, several technological shortcomings limit the degree to which a story can be emergent. One of the greatest restrictions is the limited number of plotlines a game can support. To date, most emergent narrative generation systems utilize a pool of pre-authored fragments of information from which the system must choose to create plot points[5]–[10]. Though this offers greater plot flexibility than a strictly linear narrative, the number of possible outcomes available to the player is still constrained. While pre-authored plot points are not ideal for a truly emergent narrative due to the fact that they are still limited in choice, to date there is no better solution to constructing narratives. In order for unique plot lines to be created dynamically a near human like artificial intelligence would be needed to react to player decisions and interactions with the game world.

A limited number of systems have attempted to dynamically extend their starting pool of plots by modifying similarities between plot points to adapt to a new situation [5][6][11].

However, this approach fails when the system is unable to appropriately evaluate the manipulations made to plot points, and can lead to farfetched plot lines. To overcome this obstacle, some developers offer downloadable content post-release in order to continuously increase the number of plotlines available. To ensure the viability of this solution, the game must supports the addition of new plotlines, and for further flexibility, support these additional points once game play has started.

In addition to offering a highly flexible and mutable plot, a truly emergent narrative should allow any player action and be equipped to respond to these actions, much like a situation in the real world. Indeed, with the current state of technological development some parts of a videogame can be fully emergent. For example, many physical simulations are highly predictable, allowing variable outcomes to easily be incorporated into a game. The trajectory of a ball hit with a certain material and mass at a specific velocity can be accurately simulated without the need to hardcode every possible statistical combination. However, other aspects of a videogame, such as a human's response to certain events, cannot be mathematically determined. Similarly, when a player performs a unique action or exhibits a particular reaction to an event, the surrounding videogame environment should also respond to the change. This requires the creation of a realistic reaction in real time that would allow a player to fully delve into the experience of the videogame. For example, if a player wishes to make their human character flap their arms like a bird, not only would the game have to allow the player to perform this action, but also provide a reaction by surrounding characters and/or environment to this unusual event. Realistic reactions can be difficult to produce without using predefined actions or phrases that have been incorporated into the game. The infinite number of possible actions, reactions and outcomes poses a significant challenge for the development of a fully emergent narrative in a videogame. Furthermore, limitations in dynamically created content, such as voice assets, art, and music, do not allow for gameplay to venture significantly from human created content. Currently, character animations and speech are primarily limited to pre-generated material.

While a fully emergent narrative should support any action or event, it is important that the game does not allow a player to over-influence, and in turn, degrade the narrative of

the game. With complete control over the game environment, not only does a player have the opportunity to forge a unique storyline, but also to miss parts of the intended narrative. Yet, forcing a player to simply watch the predetermined narrative, without any true influence on the overall outcome, hinders the player's freedom in the game. Either of these extremes can detract from the player's overall gaming experience. A suitable balance between narrative control and player flexibility within the game world must be determined to ensure an interesting narrative is conveyed. However, setting this trade-off appropriately can be difficult, and varies for different game types [1][2].

Although incorporating an emergent narrative into a videogame can enhance the overall gaming experience for an individual, many challenges accompany this approach to game development. Consequently, progress in this area of game development has been limited thus far. Many of the existing interactive narrative generation systems are restricted in either the level of creative control granted to the author, degree of flexibility in creating unique story worlds, the combination of user freedom and the extent to which the system can maintain the general direction of the story, or a combination of several of these metrics. The objective of this thesis is to develop and test an architecture for generating continuous, interactive emergent narratives that offers creative freedom to the author, easily adopts any type of story domain, and maintains the general intended storyline, yet does not restrain the user's actions and effect on the story world. This type of architecture offers greater flexibility in emergent narrative story generation than existing narrative generation systems and would provide users a highly interactive and immersive gaming experience. The proposed architecture can provide a framework for further development of emergent videogames in the future.

1.2 Objectives

In order to develop a practical architecture for videogame development that allows an emergent narrative to be integrated into the game, the immediate objectives of this thesis were to:

1. Design an architecture that is capable of generating a video game narrative with the use of pre-authored plot points. The architecture should offer a high degree of

freedom to the user while preserving temporal and spatial consistency within the narrative. Additionally, the author should be given discretion as to their level of control over the narrative, allowing them to express their narrative ambition easily.

2. Design a system that manages plot points in such a way that the narrative is able to recover and continue even in the event that a user's interaction with a plot is undesirable. The system should recognize repeated failure from a user, and avoid exhausting or repeating plotlines.

3. Design an architecture that can accommodate a variety of different game world types, namely open world or level-based, without requiring specific game world mechanics to be integrated. This would allow the architecture to be domain independent and easily adapted to any game style.

4. Design a system that permits the addition of new plotlines, either by expanding upon the existing pool of plot points or increasing the number of plot points, even after the narrative has begun. This important criterion would allow the players to retain progress already made in the story, without needing to restart in order to experience the newest plotlines added to the system.

1.3 Thesis Outline

Following this introductory chapter, Chapter 2 presents background information regarding storytelling, narrative development, the current state of storytelling in videogames, and the benefits and use of emergent narratives in videogames. Chapter 3 details the architecture for emergent game development that was developed during the course of this thesis, while Chapter 4 details a prototype that was developed. Chapter 5 describes a number of case studies output by the prototype system. Chapter 6 presents some discussions regarding the architecture as well as the case studies presented in Chapter 5. Finally, Chapter 7 presents the contributions this thesis has made, as well as future work.

Chapter 2

2 Background and Related Work

This chapter will begin by addressing some common terms used when describing stories, and giving the definitions of these terms as used by this thesis. Section 2.2 will follow with definitions of common terms found in narrative generation systems. In section 2.3 we will be looking at a brief history of interactive storytelling, including an early evolution of stories in video games. This will include the current state as well as some difficulties faced when developing a story for a videogame. Section 2.4 will take a look at a number of narrative generation systems, providing short summaries as well as comparing a number of key aspects of each.

2.1 What is a Story?

Storytelling is a natural human ability common in most of our interactions. Though it is a customary practice, there is no set of rules or particular formula that makes a story interesting or successful. Rather, the art of writing stories requires a focus on principles. In his book 'Story' [12], Robert McKee describes the difference between the two:

> "A rule says, "You must do it this way." A principal says, "This works …and has through all remembered time." Your work needn't be modeled after the "well-made" play; rather, it must be well made within the principles that shape our art."

Simply put, a good story is one that is worth telling. The specific definition of a story, however, is a point of contention. Although many writers and philosophers have tried to come up with a unified definition, no single definition has been fully agreed upon. Perhaps this is because several common terms often used when discussing storytelling are used interchangeably, making it difficult to converge on a unified definition. These terms include story, narrative, and plot. The definitions of these common terms, and some other terms commonly used in narrative generation systems for videogames, used throughout this thesis are described below. Although many exist, the definitions of these terms

considered for this work are those that are applicable to the context of a narrative system in videogames.

2.1.1 Story

A story "refers to almost everything that goes into a fiction and is most often used as an encapsulation of the narrative, plot, and the space in which they exist. Story is the world, the tone, the mise-en-scène, the emotional context. And most critically, story exists in pretty much everything, including those things we typically refer to as having 'no story'." [13].

As described in this definition, a story is the whole of what is being conveyed to the audience. While it may consist of various components, such as a narrative and plot, the compilation of everything the audience receives is the story. A story can be an account or recital of an event, or series of events, either true or fictitious. It can be simple or become complex by describing emotions or events that are never explicitly stated but interpreted and understood by the audience. Most importantly, at its core, a story is created by linking events together.

In creating a composition of events, a story should begin in one place and finish in another without interruption [14] [15]. This type of behaviour is commonplace in videogames. A character has a goal of reaching a new land, and the game follows them along that journey. The story (and the game) ends once they have overcome all obstacles and reached their goal. However, stories do not necessarily have to involve physical movement, from one physical place to another. The transition from one place to another can be any change in the world, such as a social, emotion or physical transformation, and arrival to a new state. This change is often observed in story world values.

2.1.2 Story Values

Story values are the qualities of human experience that can change within the story [12]. These changes can be from positive to negative or vice versa, and can occur from moment to moment. If a change from one place or state to the next occurs in a story, a change in a story value has ensued. The life of a character at the beginning of the story

should be different from the life of the character at the end, and the journey in between the initial and final conditions is what creates the story. Furthermore, the final condition must be complete and irreversible [12].

From this definition of *story values*, we define a *story* as having: (1) a starting state for the story world and its characters, (2) a series of events that leads to changes in the story values, and (3) an ending with irreversible and meaningful changes to some story values.

2.1.3 Plot and Plot Points

The plot of a story is the sequence of events that occurs to the characters or the story world. The plot is the way in which the story and story world are navigated through [12]. When a point is reached that requires an individual to choose one of several pathways, or branching choices, plotting is required to select the most appropriate path. Not only does a plot follow a writer's choice of events, it can also follow the character's choices. Plot points are the branching choices made by the characters throughout the story. These choices will influence the events that play out in the story, and in turn, create the overall plot. The type of events that transpire is dictated by the theme of the story.

Note that the plot is distinct from the story. The plot refers to the chronological events that take place in the story, whereas the story itself describes the overall change in story world values and induces an emotion in the audience.

2.1.4 Narrative

Narrative refers to the way in which the plot is communicated to the audience. This can be from a first- or third-person point of view, the protagonist, an all-seeing view, or any other form that presents the plot. A narrative can incorporate flashbacks or dreams to convey, for example, emotions or character traits, or can limit the information provided to keep the audience in suspense. The narrative is what is revealed to the audience and how these things are perceived.

While the plot must be chronologically linear, the narrative is not necessarily. The narrative can be creatively crafted to allow different event sequences to be described to the audience at different time throughout the story, or even left out completely. This

allows elements of the story to be left up to the imagination of the audience, or even allows them to infer missing parts of the story from what is revealed.

The narrative also dictates the pacing of a story. To understand what is meant by pacing, consider the movie ‘Slumdog Millionaire’ [16]. This movie follows a young Indian man as he competes on the Indian version of the trivia game show ‘Who Wants to be a Millionaire?’ The movie begins at the point in this man’s life when he has just won the game by answering all the questions correctly. The narrative employs flashbacks to show the key points in his life that helped him answer each of the questions correctly. The plot of the story follows his life chronologically through all the key moments that eventually lead him to competing on the show and doing well. However, by using flashbacks, the narrative attaches meaning to the events that occurred to the man throughout his life. Had the movie simply followed his life up to the point of competing on the game show, the curiosity and suspense created for the audience by wondering how the man came to know the answers to all the game show questions would not exist. By controlling the pacing, the narrative conveys the story to the audience in a meaningful way.

Narratives help to give plot elements meaning and context. They provide the audience with reasoning behind a character’s actions, and why the protagonist has a desire to continue forward. Narratives frame the plot points and have the power to control the way the audience interprets the story. The way in which a videogame conveys a story and its plot points is a specific variety of narrative. The narrative of a videogame is partly controlled by the player, giving them the power to setup a situation in many different ways, and even the potential to miss some elements of the story. Thus, it can be challenging for an author to completely control the narrative presented in a game each time it is played, even if the story is the same (*i.e.* linear).

2.2 Narrative Generation Terms

In the following sections, terms commonly used throughout story and narrative generation systems are defined.

2.2.1 The Story World

The story world is the representation of the world in which the story takes place. This can be a virtual world that characters walk around in, or an imaginary world described via text. The story world contains all characters, locations, objects, relationships, weather, *etc*. relevant to the story.

2.2.2 Plot Fragment/Plot Point

A plot fragment, also referred to as a plot point, is a section of the plot used in story systems. They are usually ordered or activated in a particular manner in order to increase the appeal of the narrative. A system may contain some pre-authored plot fragments or plot points that serve as a starting point for a story.

2.2.3 User

The user of a system is the audience to which the story is being told. In an interactive system, such as a videogame, the user can manipulate parts of the story world. The user starts the system and is engaged in the story presented. In a videogame, the user is commonly referred to as the player.

2.2.4 Characters/Agents

Characters or agents are elements that exist within a story world. This includes people or objects in the world (such as trees or cars). Agents can be autonomous, meaning they will act and move on their own based on some internal model.

2.2.5 Presentation

The presentation of a system is the representation of the story with which the audience interacts. This can simply be a text description of events, or as complex as a three-dimensional world that users walk around and interact with.

2.2.6 Director/Drama Manager

The Director or drama manager is a part of the architecture that controls the story world and agents within it. It is also responsible for presenting plot fragments to the user. It can

exert significant control over the story world, and is able to decide which stories or plot points occur and when.

2.2.7 Author

The author is the individual who inputs story content into the system. Story content can range from simply describing the domain of the story world, to premade characters and plot points the author wishes to portray.

2.3 Storytelling in Videogames

2.3.1 A Brief History of Interactive Fiction

In comparison to today's videogames, the earliest videogames developed in the 1950s were extremely primitive. They were primarily intended to show the capabilities of existing technology, rather than to entertain the masses. The computing power was minimal, the graphics were simple and the amount of player interaction was limited [17]. Consequently, the notion of incorporating a 'story' into a game was not considered. Throughout the next three decades, most of the innovation was invested into game play. Text-based adventure games became available for the computer, and although they were story driven, the story was not accompanied by any graphics. It was not until the early 1980s with Nintendo's 'Donkey Kong' that short cut scenes were used to incorporate some story narrative elements into a game [1].

Graphic adventure games arose with the increase in computing power. Through the use of images and animations, the stories of games were further developed. Without solely relying on one's imagination, solving puzzles and navigating around the story world become less difficult. Although most of these 'point and click' adventure games told traditional linear stories, some games began to incorporate branching storylines and multiple endings.

Further increases in computing power and memory afforded game developers the opportunity to create more in-depth stories for their games. This lead to longer conversations among characters within the game and branching dialogue scenes were introduced to allow players choice. In turn, player interaction with the game was

enhanced since they were able to decide what to say when interacting with another story character. Developers also began to pre-render scenes cut from full motion videos, permitting the stories to be told at a higher quality than game graphics allowed. Another significant advancement came with the inclusion of full voice acting throughout the entire game. With audible character voices, emphasis and emotion could be conveyed, further heightening the level of player immersion in the game.

2.3.2 Storytelling in Current Videogames

With present day technology, game developers are continually experimenting and enhancing all game play elements, including storytelling. For example, first person shooter games, which previously lacked great stories, have evolved to include well written storylines that accompany intense action scenes. Open ended and multi-branching stories are becoming increasingly feasible as more computing power and hard drive space becomes available for assets, making each play through unique.

Open world games with expansive terrain allow players to explore different environments on their own and play the game as they wish. This type of gaming framework opens the door for more open-ended methods of storytelling. Without the restriction of following sequential levels, stories can be told in a nonlinear fashion. The order that missions can be played, and even where in the story world they can be played can be rearranged. An example of this is in ‘Red Dead Redemption’[18]. The player controls a gun slinger in Western style America. Throughout the world, there are side missions available to the player, including wanted posters for capturing outlaws. These posters appear in different towns, each with a unique outlaw to capture. These appear periodically within the towns and can be played either between the main story missions, or ignored completely.

In providing a player optional levels and the freedom to choose when to play them, a feeling of control over the story is created. This allows for new opportunities in storytelling. However, it also creates obstacles in story development that must be overcome. For instance, the story world must remain consistent. As the missions or levels are played, the author must ensure the game facts are accurate throughout the different

paths, regardless of the order of play. To constrain this problem, the author can make sure certain levels are played before others by requiring prerequisites for later levels.

2.3.3 Difficulties in Developing Storytelling for Videogames

Unlike television or film, videogames have the unique advantage of being interactive. Without any interactivity or simulation, a videogame is still capable of communicating the same visuals and audio as a film. With the addition of interaction, a player of a videogame is able to directly manipulate the world they have become a part of, and become the character they are playing [19]. They can immerse themselves in the story even more so than with television or film, and make it their own. However, this advantage also comes with added obstacles. If incorporated inappropriately into a game, interactivity can ruin the story for a player. Graphical glitches or inconsistencies in the story world can hinder one's immersion in the game and the overall gaming experience.

Advances in game development not only foster a more interactive experience for players, but also impose more work for developers and testers. When games shift from a linear level composition to open, explorable worlds, developers need to consider the significantly increased opportunities that become available during play. Stories with open ended plots and multiple branching pathways must account for a player's adventure through any of the plotlines to ensure the story makes sense for every play. As an example, if a player in Red Dead Redemption decides to complete all the wanted poster side missions before proceeding with the main storyline, the developers must ensure that none of the wanted outlaws are involved in the main story. Capturing or killing a wanted outlaw who reappears later in the story creates inconsistencies in the plot. It is these types of plot holes, simple or complex, that need to be considered during game development.

Even with the advent of open ended play, games are still limited in the number of plotlines available to the user. The number of options a game can support will be infinitely smaller than the number conceivable in real life. Unfortunately, the advancement of artificial intelligence has yet to reach a point where computers can generate stories and reactions to all possible game scenarios. Thus, a player is limited to performing only the actions the design team has implemented in the game, and similarly,

the game can only react in a set number of ways [1]. This places a natural limit on the number of story options available. Furthermore, the story must be able to handle any possible inputs that it does receive.

Another issue with developing storytelling for games is consistency. The story must match the gameplay elements, and as more complex stories are incorporated into games, these two parts often clash. For example, a plotline may have a main character spare the life of an enemy because they felt the death was undeserved. However, if the gameplay mechanics permit the enemy, or even any innocent characters, to be killed without repercussions, the player may feel disconnected from the game because the story told and what the gameplay depicts as a harmless action is inconsistent.

2.3.4 Videogames as a New Form of Storytelling

Developers often try to compare storytelling in videogames to storytelling in traditional films [2]. It is common practice for a company to hire a screenwriter to compose the storyline for their videogame, and gameplay is adapted to the story. Because videogame development is still relatively new, developers attempt to exploit a style of story writing that is well developed and proven. However, subtle differences between the two mediums can significantly affect the way the story is conveyed and these differences must be considered. Videogames have mechanics unavailable to film, such as interactivity and choice. This is where the application of storytelling in film to videogames begins to fall apart. Storylines in a movie are fixed, and the actors perform the same actions identically each time the movie is viewed. In a videogame, however, a player may follow the story intended by the author, or they may miss events pertinent to the story, intentionally or unintentionally.

Because a videogame is quite different from traditional story writing, techniques that work well for books, plays, or film do not necessarily translate well to videogames [2], [3]. In a book, play or film, a character can give a long speech to convey their thoughts or emotions. However, a soliloquy given by a secondary character in a game consumes time that the player could otherwise be spending interacting with the story world. These long speeches are sometimes incorporated into a game using cut-scenes. Yet, too many cut-

scenes can make a game seem more cinematic than interactive. Another technique often used in traditional storytelling is ‘forced failure’ [3]. This is the practice of knocking a character down when things are going well for them, forcing them to recover from a hurdle before moving on. Forced failure can be frustrating in videogames since it appears that the player is simply being penalized for reaching a certain point in the story, rather than for making a mistake. After the player works to attain a particular goal, they may feel cheated by having their accomplishments taken away during a cut-scene.

Unlike a videogame, a film has a predetermined and fixed duration. This is a key reason as to why storytelling methods do not translate well between the two mediums [1], [20]. In a film, the pacing is strictly controlled and story is filled with carefully selected and meaningful plot points. Because videogames are often much longer than a film, their storylines must be longer, particularly in the middle portion of the game. Most games introduce the story and problem well at the beginning (act 1), and conclude with a strong solution to the problem at the end (act 3). However, it is in the middle (act 2), where the story must be developed and shaped, that most games struggle. Because this portion of the story makes up the majority of the game, it is often filled with meaningless tasks and missions in order to make the game longer. Making the story of a videogame meaningful, yet an appropriate duration, can be challenging and much more difficult than creating a story for film.

Videogames are far more comparable with seasons of television shows than with films, as they have similar constructs of time [19][21]. For a television show, the entire season consists of a major story arc, but each episode conveys a mini story. If a videogame were viewed in a similar manner, each play session, like a television episode, could contain a mini 3-act structure, and create more excitement in the ‘second act’ of the major story arc. This episodic format has been applied successfully to a few games to date, including. ‘The Walking Dead Season One’ by Telltale Games [22] which was released in 5 installments over the span of 8 months. This format requires a great deal of writing effort and manpower, and is the reason television shows often utilize a large team of writers. However, it may help to keep the audience engaged throughout the entire game and improve the lacking ‘second act’ of game design.

Although a critical component, the storytelling aspect of a videogame is usually one of the last things a developer works to improve [1], [19]. Large videogame companies are primarily concerned with developing games that sell. Therefore, they often do not try to alter a successful formula for franchise titles such as the Call-of-Duty or the Halo series. With the rise of independent games, this is slowly changing [23] [2]. 'Indy' gaming platforms, such as the 'Indy' sections of the of the Playstation Store [24] and the X-Box Games Store [25], as well as the Steam Green Light Program [26], allow small companies and individuals to introduce games to the general public. Independent developers are less restricted with the features and gameplay mechanics they can experiment with, and since a strong story or unique narrative can help boost a game's success, investing in story development is often a draw.

In order to entertain and hold an audience, a well-crafted story in a videogame should induce significant emotions in the players. Although the focus of game development has been primarily technological, the future of game development lies is in the emotional content a game holds [23]. These emotions can be extreme feelings, such as screaming in pain, crying out of fear, or running in panic. However, it is the induction of subtle changes in the player that can truly capture the audience's attention, and should be a high priority of videogame storytelling [2]. Inducing mental stress and strain, causing a feeling of relief, and creating a sense of accomplishment when an obstacle is overcome or an opponent is overtaken yields an attachment to a character. It is the rollercoaster of emotions that a game can take you on, similar to an experience in the real word, that allows a player to become fully immersed in the story world and heightens the excitement of the game [2].

2.3.5 Techniques Used to Achieve Immersion in Videogame Storytelling

One goal of interactive drama is to allow players to choose how they want a story to unfold, or in terms of videogames, how to play the game. However, there must be measures in place that encourage the players to play in character so as to avoid a

phenomenon known as ‘desk jumping’[2]. This refers to an action a player would take that does not fit the persona of the game character. It occurs when the goals of the player deviate from those of the character. The conflict between a player’s desire and the author’s intent is a core dilemma in interactive storytelling [27].

While a possible means of avoiding desk jumping is to restrict player agency to disallow actions that are out of character, this solution can disengage a player from the game. The player loses a sense of freedom of doing what they want, and in turn, immersion in the story world. Permitting any actions and either ignoring them or responding to them in the story are also not optimal solutions. Ideally, the player’s goals become naturally aligned with that of the characters. Although this can be difficult solution to always achieve, a number of techniques have been developed to force the character and player aspirations to amalgamate.

Fast-paced action games often attempt to capitalize on the adrenaline rush players often feel in order to direct their actions. For example, games in the Call of Duty Series [28] use fast-paced action to instill the player with a desire to move to safety quickly and continue with their mission. The player feels the need to move to the next location because of the rush of battle created, while the goal of the character is to complete the mission in order to stay alive. The game creates a goal for a player that matches that of the character to ensure their personas align.

Imposing repercussions to certain actions is another method of forcing a player’s desires to align with those of a character. For example, in the Grand Theft Auto [29] series dying or being arrested means losing some of your money and guns. In Minecraft [30], everything in your inventory is lost when the player dies, and you must return to the location of death to retrieve it all. Furthermore, time limits and other game obstacles can result in permanent loss of inventory. These types of consequences to actions can cause hours of work to be lost, instilling players with a sense of care within the story world. While players have the option to play outside the boundaries of the intended story, there is an inherent incentive to remain in character.

The story world narrative can also be used to impose restrictions on the player in an open story. The world narrative is the story or history of a place, and the people and objects in it [2]. Rather than having a player witness a particular event, in which they could potentially interfere, they are informed of it afterwards by witnessing its resultant outcome. Sylvester gives an example of this type of story restriction [2]. If the player must be informed of a murder that took place, one method is to have the player witness the murder firsthand. However, with high levels of freedom, the player may interfere with the murder. If they prevent it from happening, the intended story must change. Instead, the narrative can be setup such that the player only learns of the murder by coming across the dead body. In this way, the player receives the story, but is unable to change the necessary plot point in any way.

An additional benefit of employing a world narrative to tell an interactive story is the unrestricted order that the story can be told. The player does not necessarily have to follow a specific path through the story, which gives them the freedom to explore and discover the story world on their own. In the crime solving mystery game 'L.A. Noire' [31], the player is a detective solving a new case with a new crime scene for each level. Each level has clues to be found and witnesses to be interviewed, and how this is done is variable. How the witnesses are questioned and which clues are found will determine whether the crime is solved. Because the world narrative is told using the clues and people in it, the levels can be played out many different ways.

2.4 Narrative Generation Systems

Narrative generating systems can be divided into two major categories – 'Story Generation Systems,' which generate a story for the audience to observe but not take part in, and 'Interactive Narratives,' which permit the audience to interact with and manipulate in some way.

2.4.1 Story Generation Systems

This section discusses two important story generating systems – the work of Michael Lebowitz and his system 'Universe' [5], [32], and the work of Scott Tumer and his system 'Minstrel' [6]. These were among the first systems created for computer

generated storytelling. Both systems automatically generate unique storylines by compiling story fragments, and employ dynamic methods to modifying these fragments in an attempt to expand the knowledge base of the system on its own.

2.4.1.1 Lebowitz's Universe

Lebowitz's storytelling system 'Universe' is an automatic narrative generation system that uses plot fragments to generate plot outlines of a story [5], [32]. The system takes reoccurring plotlines, or 'plot fragments' of a daytime television soap opera to create intricate and intertwined stories among a large cast of characters. Using a set of pre-defined plot fragments and characters, the system is able to create a wide range of extended stories that continue on with no fixed end point. Rather than directly using a human thought process to create stories, Universe is a way of developing a set of well-defined characters and plots that interconnected them.

In Universe, a story begins by generating a rich cast of characters and a detailed world. The story is filled with various plot fragments that align with the goals of the author. The plot fragments used by Universe do not, however, consider the goals of the characters. The goals and personalities of each character should be considered in order to determine how they would respond in different situations and maintain story consistency. However, the author's intentions should override those of the character's in order to introduce interesting twists to the story.

Story set-up in Universe consists of two phases. The first is a character generation phase [32], in which a complex set of interconnected characters is created. This involves playing out a character's life up until the point of the story, and creating all necessary relationships to other new or pre-existing characters. Following the generation of all main characters, story planning is conducted in the second phase of Universe. Plot fragments are chosen and arranged to create an interesting storyline, which includes actions of the characters and changes to the story world. Because the story is meant to be ongoing, the entire story is not planned at once.

The storytelling algorithm in Universe begins by choosing a goal from the goals queue that has no missing precondition, and then deciding which plot fragment can achieve that goal. Each of the matching plot fragments and characters that can potentially be used is checked for compatibility with the current story. The value of a potential plot fragment is evaluated based on the number of goals it can achieve (*i.e.* a plot fragment that achieves more goals has a higher value). This permits multiple goals to be pursued and more plotlines to become interleaved. Once a plan is complete, the story section is executed and new goals are added to the queue if necessary.

Upon initialization, the Universe system has the ability to generate many characters, each with unique traits and a rich back-story. Additionally, the various relationships among different characters are established. By interconnecting the characters with defined relationships, the system can not only track consistencies within a storyline, but can continually intertwine the plots of the different characters. Universe was the first storytelling system to devote special attention to character creation, and its ultimate goal was to allow user to create characters could easily be woven into the story at any point. Unfortunately, the goal was never achieved. New characters cannot easily be introduced, unless character generation is run again.

When the project ended, Universe consisted of 65 plot fragments [33]. Even with this limited database, a large number of stories can be created. However, after running the story for a long period of time, the quantity of new plots to run eventually becomes low. Lebowitz proposed that the existing plot fragment library could be expanded by relaxing certain conditions to generalize the existing fragments for future use [5]. For example, in the ‘forced marriage’ plot fragment, a husband and wife would be forced to stay together because of a threatening father-in-law. A simple generalization would be to generalize the condition to any couple in a positive relationship, not specifically those that are married. Generalization of plot fragments could extend the number of stories generated. However, there was no guarantee that the additional plot fragments would make sense in the context of the story world, an issue also present in Turner’s Minstrel system.

2.4.1.2 Turner's Minstrel

Turner's story generating system 'Minstrel' tells short stories about King Arthur and the Knights of the Round Table. The system starts with limited knowledge about the domain, as if having read only a few short stories about King Arthur, and goes on to generate a number of compete and incomplete stories from this knowledge.

The stories are created using a problem solving approach. When faced with a problem, Minstrel searches the database for a similar, previously used problem, and adapts that situation to the current one. To generate creative solutions, rather than simply reusing solutions to old problems, Minstrel uses 'search and adaptation' processes to transform the old problems called 'Transform-Recall-Adapt Methods' or TRAMS. The old problem is changed slightly, or *transformed*, into a new problem. Then, past problems that are similar to the newly transformed problem are *recalled* and the same operation as before is performed. Lastly, the similar, past solution is *adapted* to the new problem. TRAMS are an essential component in generating creative stories based on a limited base of prior knowledge.

Minstrel is a two-staged story building system. It consists of a planning stage and a problem solving stage. In the planning stage, the author's unsolved goals are evaluated, and are either broken down into smaller goals or passed directly to the problem solving stage. In the problem solving stage, the story segments are evaluated and matched using prior knowledge of stories and modifications of segments made by various TRAMS. Unlike Universe, Minstrel creates short stories with a single authorial theme with a defined ending. To this end, Minstrel's planning process must ensure a resolved ending, which limits its freedom in plot choices during the planning process.

Minstrel categorizes the various author-level goals it maintains. These categories include thematic, consistency, drama and presentation goals. Thematic goals are those that reflect theme that the chosen story attempts to teach the reader. Consistency goals ensure that the story is believable and logical. Drama goals use writing techniques to make the story interesting, which include foreshadowing, tragedy and suspense. Presentation goals are concerned with the way the story is delivered to the audience. A number of different

techniques are used to make the story pleasing to the reader. Minstrel must decide on the correct order of events, incorporating things like foreshadowing appropriately. Next, the words and phrases to describe the segments of action must be generated. Lastly, Minstrel must create new scenes to introduce new characters or topics in the story. Each scene has a different goal. For example, an introductory scene can be used to bring the reader into the story by introducing the main characters and genre.

Minstrel works by comparing the current situation to those in its existing bank of knowledge. If Minstrel comes to a problem which it has no prior knowledge for dealing with, a TRAM: Cross Domain Solution can be applied. For this, a similar situation from a different domain is found and applied to the current situation to form a solution. However, TRAMS do not necessary conform appropriately to every situation [34]. For example, if a knight pokes himself with a sewing needle, the TRAM:Similar-Outcome-Partial-Change recognizes that being killed is similar to being injured. In turn, this allows a sewing injury to be a means of committing suicide. The situations that the TRAMS model tend to be written for achieving very specific scenes and do not necessarily adapt to new situations well [34]. TRAMS can also be used in conjunction with each other to enhance the complexity of the creative process. However, as the level of complexity increases, there is a heightened risk of creating a story that is farfetched and unbelievable [35]. Unlike Lebowitz's storytelling system 'Universe', Minstrel attempts to recreate the human creativity process. However, the system does not support any form of human reasoning, which becomes a problem when a creative solution makes no sense to the story being told. Ideally, a system should be able to evaluate if the use or adaptation of a plot fragment is realistic before implementing it.

2.4.2 Interactive Narratives

While a story generation system can plan out most, if not all, of a story before telling it, an interactive story can only devise a small portion of a story since the user's interaction is unknown. A story is something that is structured, with a sense of time and pace, whereas interaction is about doing what you want, when you want to do it. It is the goal of an interactive narrative to make these two conflicting ideas interact well in a story world. The following sections review interactive narrative systems as related to a

videogame environment. While not all of these interactive systems were developed with gameplay in mind, they all have aspects that are suitable for a flexible videogame environment.

2.4.2.1 Oz Project

The Oz Project is an interactive drama system that creates believable agents and an interactive narrative[7]. The story consists of pre-authored points, input to the system by the author, that make up the major transitioning points of the story. The author will have an ideal order of these plot point. However, rather than stating an explicit order, a function that evaluates the suitability of a given order of plot points is created and used by the drama manager. This function takes into account some aesthetics of what the author believes makes a good story. At the beginning of a story, the author's intended order of plot points should score high with the evaluation function. Within the given plot points, the user is able to move and act freely, with no interactions from the drama manager. At any given point in the story, where the user may deviate from the intended order, the score will change. Once a sequence of activities that constitutes a plot transition is detected, the drama manager begins to work. It is the job of the drama manger to determine which permutation of remaining plot point will produce the highest score in the evaluation function, and encourage the user along this path. By considering the previously visited plot points in conjunction with every possible permutation of remaining points and user's actions, every possible 'total history' is formed. Using the evaluation function, the drama manager decides which total history is most desirable, and in turn, which plot point should appear next. This system allows the user to deviate from the author's original story, while still maintaining the story path closest to their original vision.

2.4.2.2 Façade Architecture

Similar to the Oz Project, Façade integrates believable agents and drama management together [36][8]. It also incorporates natural language processing and simple graphics to create an experience of interactive drama for the user. The game involves a 20 minute play-through in which the user experiences a dinner party with a couple of friends

experiencing a rough patch in their relationship. The user interacts with the dinner scene by inputting text dialogue. The story can play out a number of different ways by selecting a unique combination of plot fragments known as ‘story beats’. There are approximately 200 unique story beats from which the system constantly chooses [8]. The beats are about one minute long and contain a main goal, as well as information about the characters and story world. The beats also contain information to decide how they will be played out which is determined by their context. For example, if a beat is used early in the sequence (*i.e.* early in the evening before things become tense), the context would be lighthearted. However, later in the sequence (*i.e.* later in the evening when tensions have risen) after many other beats have been placed, the context may be quite different.

Similar to the Oz Project, the Façade framework also allows for the author to easily organize content of their interactive drama by using a hierarchy of story beats. The beats are arranged by Façade’s beat sequencer based on their story tension effect, which is determined by its preconditions, effects on story tension and weight. At a given point in the story, the beat whose story tension matches best with the tension of the idea story arc would be given the highest score and inserted into the sequence. This is similar to the Oz method’s evaluation function for plot points.

Most research in the field of interactive drama is focused on small, specific areas, such as story design or integration, etc. However, Façade attempts to cover all aspects of an interactive drama. In doing this, the creators were able to coordinate the interaction of different parts of the architecture. By developing the system as a whole, the team was forced to address problems that are typically overlooked when parts of the architecture are developed in isolation.

Although player agency is high and the player is able to move around and interact with many aspects of the world in the Façade system, most of the interaction between the user and the story world is through text dialogue. Improvements in natural language recognition would allow the player to convey more, understandable phrases to the characters of the story [8]. When the system fails to understand a phrase, it can lead the story in an unintended direction, resulting in frustration for the user and a break in game

immersion. Additionally, the characters in the game tread toward the same topics of discussion, with little control given to the player. These topics are discussed as the game is played through multiple times, creating a forced story agency toward the story[8].

The architectures of the Oz Project and Façade system require a large quantity of authored content. This places a great burden on the author to create more story beats or plot points, as well as an optimal evaluation functions. The amount of authored content for future scenes or levels increases exponentially as the games become larger-scaled and the number of outcomes grow. The increase in complexity can require unrealistic amounts of content to be developed for a game. Furthermore, adding more content increases the chances of producing a story further from the intended direction. Generating a function to evaluate the hierarchy of plot points or beats would become increasingly difficult as more are added to the system.

2.4.2.3 The Virtual Storyteller

The Virtual Storyteller is a robust, interactive story generation system that aims to create virtual environments inhabited by autonomous agents with which a user can interact. Not only does the Virtual Storyteller seek to develop stories that are coherent, but also interesting [11]. Characters are allowed to set their own goals, and act based on their internal goals and personalities. Because the characters can act realistically, story consistency is maintained. However, this does not guarantee an interesting plot.

In an attempt to ensure an entertaining story by influencing a character's actions, while still permitting them to make their own decisions, the Virtual Story teller uses a 'Virtual Director.' The Virtual Director manages the characters in the story, and no character action can be performed without Director approval. This screening process prevents a character from performing an irreversible action that could ultimately ruin the story. Because the Virtual Director only approves actions, rather than choosing them directly, the characters remain believable. The characters can still choose their own actions and respond emotionally to the situations they are in. In this way, the Virtual Storyteller attempts to avoid a significant downfall of a purely agent-controlled story: the inability to guarantee an interesting story.

The Virtual Director must steer the story in a particular direction. To do this, it must judge how well a character's intentions fit with the plot structure using information about the story world as well as criteria for making a good story [11]. Unlike many other story generating systems, the Virtual Storyteller does not contain pre-specified plots or plot fragments, but only the general knowledge of what constitutes a good story. It is with this knowledge that the system can decide whether a character's actions will be interesting. The Director employs several different methods to influence a character's actions: (1) environmental, by introducing new characters of objects into the story world, (2) motivational, by adding to a character's goals, or (3) proscriptive, by disallowing a specific character action.

At the start of the story generation process, the Director establishes the basis of each episode by selecting four 'episodic scripts' from a database. These four scripts are based on Propp's story functions [37]: (1) initial state of equilibrium, (2) disruption of the state of equilibrium, (3) mission of the hero, and (4) return to the state of equilibrium. A story world is created according to the specified settings within the selected scripts. The episodic scripts are pre-authored with very basic outlines and goals for the characters. However, the episodes are emergent since the character must choose the actions necessary to achieve their goals. The character will pursue a goal until it is attained. If the character cannot achieve their goal, the Director must find a new way for the goal to be reached.

In order to enhance the creativity of the stories produced by the Virtual Storyteller, a "Creative Problem Solver" system was developed [38]. Similar to Minstrel's TRAMS, the Creative Problem Solver employs a case-based problem solving approach to create plotlines based on those that have been previously used. When a character must make a decision, it is posed as a problem and passed to the Creative Problem Solver. To handle the new problem, the Problem Solver first searches for a solution given the problems requirements and constraints. If no solution is available in the system, the problem is transformed into a similar problem, but with some requirement modifications. This process repeats until a good solution is found, at which point it is adapted to satisfy the original problem. By transforming the problem, the Creative Problem Solver can find a

reasonable solution that did not necessarily exist beforehand. This case-based reasoning system, which agents and the Director are able to consult with, allow the Virtual Storyteller system to modify and extend existing story plotlines to produce unique stories.

Work has also been done to allow modifications to the story world in order to fit a specific situation. Late commitment in emergent narratives is the practice of changing story world details, as long as these changes can be accepted as having always been true by all agents [39]. If the Director wishes to alter some aspect of the story world, it must be communicated with all agents. The agents must decide whether the change will affect their current goals or conflicts with any previous actions. For example, if a player needs a map to reach a destination, the Director may decide that the player has the map. If previous knowledge of the map would drastically affect a previous plot, then the late commitment change cannot take place. If, however, the knowledge of the map would be irrelevant to any previous story plots, then there is no harm in assuming the character had the map all along. Together, the use of late commitment and the Creative Problem Solver work together to manipulate and create a variety of interesting story plots.

Because the Virtual Storyteller is purely agent-driven, control over how the story unfolds relies primarily on character goals. While this allows for truly emergent stories, predetermined structure or goals are not guaranteed. The author has no assurance that a certain character will make their desired choice, which could possibly ruin a dramatic moment in the story. Manipulation of the author's character model and testing would be required to increase the chances of producing the desired storyline, but even this would not guarantee a particular storyline. While this may be the goal of the Virtual Storyteller, this quality of the system makes it more of a story simulator than a story generation system. Although the Director can structure the story appropriately, it is unable to make actors perform actions outside of their regular behaviours. This limits the likelihood of any out of character plot twists occurring.

Note that the Virtual Storyteller does not currently accommodate any form of user interaction. It does, however, have many aspects that could be translated into an interactive storytelling system. For this reason, it has been included in this review.

2.4.2.4 Interactive Drama Architecture

The Interactive Drama Architecture (IDA) is an interactive drama system that seeks to. to allow a human author to design a plot, while giving a user as much freedom in their choices as possible [9] [10]. In this way, the user can make decisions that may work with or against the plot. The system builds a predictive model of user behavior in order to deal with the random actions that an interactive user may take. By using a predictive rather than reactive model, the story system can proactively try to avoid conflicts between the user and the plot by steering the characters away from those situations.

IDA uses a central Director agent to monitor both the user's behaviours and the plot. The Director labels the user as being in an error state or not, depending on whether or not they are following the intended storyline. Because the user has the freedom to make decisions, the Director must decide which state they are in and act accordingly. The Director must also lead the intelligent actors in the story world. These agents are predefined with characteristics that describe their physical and mental states. The Director agent will communicate any plot points pertaining to the agents, but only give them the information as they need it. Lastly, the Director is responsible for initiating parts of the plot as the user interacts with the world.

The IDA uses a state-based model to represent the plot. Each scene of the story is represented by a graph of the desired states to be reached within that scene. The scene can have a single, final goal that it should reach, or multiple states goals to add variability. Each connection between different states has s depth limit, which provides timing information. Timing can refer to duration (*e.g.* in seconds or minutes) between desired states, or the number of actions taken by the user. If the depth limit is reached before the necessary state change occurs, the Director can interfere and try to encourage specific user behaviours.

User error can be detected in three different ways. The first is to perform a search from the goal state of a scene to ensure the next goal state is attainable. The second method is to determine the likelihood a user will reach the next state. The predictive model is used to determine the user's likelihood of reaching the goal state or performing an action that

will prevent it. When the user veers from the predicted path or the likelihood of reaching the next state diminishes, they are considered to be in an error state. The third method is to monitor the timing of the state transition. If the user reaches the depth limit, they enter the error state. It is the responsibility of the Director to handle all failure situations; otherwise the story may come to a dead end. Being able to handle any error state requires a large number of pre-authored responses to situations, and requires many hours of testing. While complete user freedom sounds appealing, handling a rogue user can be a difficult task, and IDA does not have a solution for handling all situations. Even if the predictive model can consistently predict a user's actions, an unexpected event could derail the entire story if not handled appropriately.

Although the state-based model used in IDA allows the story to be run according to the author's intentions, it does not inherently give the user complete decision-making freedom. The user must progress through the specified states of a scene. This reduces the architecture to more of a branching tree with a limited number of paths, rather than an interactive emergent system. Furthermore, this model can potential have coherency problems [40]. The author is responsible for forming the plot graphs. In authoring the pot points, an author can potentially write dialogue that refers to past event that they may have anticipated would happen, but did not yet. The likelihood of this occurring increases as more content is included in the system, and it the responsibly of the author to ensure coherent plot points.

2.4.2.5 Generator of Adaptive Dilemma-Based Interactive Narratives

Generator of Adaptive Dilemma-Based Interactive Narratives (GADIN) is a system based on the premise of presenting characters with dilemmas for which they have to make choices [41][42]. These dilemmas are usually cliché plots found in everyday life or common stories, such as 'getting a girlfriend.' The dilemmas chosen to be presented are based on a player model developed for the individual characters.

The GADIN architecture attempts to solve two problems in current interactive narrative systems. The first is maintaining the dramatic interest of a narrative over a long period.

The second is being domain independent and allowing any type of story to be used in the system. Making tough decisions with weighted outcomes is thought to be a dramatic aspect of any storytelling form. Thus, the use of dilemmas is a critical aspect of the system's structure for maintaining the dramatic interest of the audience. The second goal is achieved through the architecture of the system. The main program searches through its database of domain dilemmas and storylines to create the story presented to the audience. This database can be modified to become domain specific for any type of story.

The system deliberately attempts to choose dilemmas that will have a strong effect on the story world and the characters in it. Once a list of potential dilemmas is chosen, a search for the most appropriate storyline begins. Planning is use to achieve the preconditions of the desired dilemmas. The planner decides on a sequence of actions that can be performed in the story world in order to achieve the preconditions. Based on the current state of the story world, each possible sequence plan is complied, and one is selected based on some criteria such as previous character decisions, recently used storylines, player models, *etc*. The plan is presented as a sequence of events to the user, after which the dilemma is presented and the user's choice is recorded. During the sequence of events, the user can input commands for the actions they want to perform. The user is able to stray from the intended plot with these actions, but the system will generate a new place to achieve the intended dilemma, if possible. If it is not possible, a new dilemma is chosen. The cycle repeats with a next dilemma, which is selected based on what is most appropriate for the user.

Dilemmas can be presented to all characters in the story as well as the user. The characters (controlled by the system) will make choices for any dilemmas based on their player model. The dilemmas presented to the user is often in the form of a 'yes' or 'no' question. Any choices made affect the story and modify the player model of every character involved. This design allows the system to run, even without user interaction, since choices can be made by each character using their current player model.

The dilemmas presented to the characters are the pivot points of the story. The information presented in between is essentially a simulation of the story world where the

characters in the story perform actions based on their internal character models. The user is able to interact with the story world outside the conflicts presented by inputting commands. Once a conflict is presented, the user must choose how to respond and affect the story world. Through the choice requires a simple binary response, it is intended to be difficult, with both options having positive and negative consequences on the story world. Once a choice is made, the stimulation of the story worlds resumes and works toward the next dilemma. The freedom presented to the user in the story world is limited. They must make decisions based solely on the dilemmas presented, and can only perform a finite list of actions that will affect the story world.

GADIN is able to support both continuous stories, such as soap operas that run for a very long time, and finite stories with a set goal. The finite method simply picks a goal and creates a plan to go from the current story world state to the goal state. If the user makes decisions that stray from the current plan, replanning takes place. Generations of story can also go on indefinitely by repeatedly choosing new dilemmas to present to the characters and planning storylines to achieve the preconditions of those dilemmas. Although the dilemma space is not an infinite pool, dilemmas can be used differently each time since new characters can play different roles. In this way, the system can generate lengthy stories while still maintaining the interest of the audience, similar to the structure of a soap opera.

Because GADIN is designed to be independent of any domain, any creator using the system can input their own specific domain information and story world settings and the system will create an interactive story. This is approach of authoring content for an interactive narrative is significantly different from most other interactive narrative systems. Rather than depending on pre-authored content (such as plot points) to create a structured story, GADIN only requires domain dependant information such as such as common actions and clichés, as well as common dilemmas presented in that domain. However, the author then has very little control over the story that is generated, and in turn, little opportunity to exercise creativity.

2.5 Evaluation Metrics

The following sections evaluate different design aspects of narrative generation systems that must be considered when using a system to create a story.

2.5.1 Temporal and Spatial Consistency

Consistency within a system is the guarantee that the narrative will be comprehendible and continuity is maintained throughout. Consistency can be evaluated temporally and spatially. Temporal consistency refers to the degree that the system guarantees the plot occurs in the correct order. Spatial consistency refers to the degree that the system controls whether the location of characters or objects are accounted for in the story world. For example, if a burglar robs an empty house, the story cannot later mention that the owners were actually home at the time. This can accidently happen in order to fill plot requirements and should be avoided.

The Oz system uses an arrangement of plot points favourable to the author by placing a higher evaluation on permutations that more closely follow the author's intended narrative. This is done by trying to capture the author's belief of what makes a story good. However, as the user is able to deviate from the intended arrangement of plot points the drama manager is responsible for choosing plot points to present to the user. The design provides no guarantee that the drama manager will evaluate a consistent plot ordering. This could lead to an inconsistent ordering evaluating highest and being presented.

Alternatively, consistency within Façade and IDA relies on the author. Façade maintains consistency in its hierarchical arrangement of story beats. Beats are only played once they appropriate preconditions have been fulfilled, and consistency will be maintained so long as the hierarchy of beats is constructed appropriately. In IDA, the order of plot points established by the author partially ensures consistency. It is possible, however, for the author to write content that refers to events that have yet to occur. It is the author's responsibility to ensure this does not happen, which becomes increasingly difficult as more content is added.

The Virtual Storyteller system maintains consistency through its use of agents. The agents follow their own goals and only perform actions consistent with their agent model. The Director can restrict agent's action, but only if they align with their models. Even if story facts are altered using late commitments, the changes must still be approved by each agent.

The GADIN system utilizes semiautonomous agents to control their own actions within the story. Decisions and actions of the agents are based on their internal character model to ensure their actions always align with their personalities, and therefore maintain consistency. The occurrence of story events and dilemmas require preconditions to be fulfilled, ensuring consistency in the story world.

2.5.2 Granularity of Story Control

Granularity of story control describes the control the system has when creating a story. This includes the extent to which the system controls the story world elements and the degree to which the story relies on the actions of the user. A highly granular system controls a large portion of the narrative, and allows the smaller pieces within that section to be manipulated by the user. The overall direction of the story is controlled (by way of major plot points) but how those plot points are achieved is not controlled. On the other hand, systems with low granularity exhibit high control over the story world elements and the plot points. Each aspect of the story is micromanaged in order to guide the user in the correct direction. Games with linear storylines are considered to be low in granularity because they have controlled environments and the user plays the story exactly as the author intended.

Because only major plot points are decided by the system, the Oz system and the Virtual Storyteller have a large grain. In the Oz system, major plot points are arranged by the drama manager, and control within the plot points is given to the user. This gives the user the opportunity to manipulate the story world, yet maintains the general storyline. Stories formed by the Virtual Storyteller are manipulated by the Virtual Director agent. The Director has a general guideline of four pre-authored plot scripts, and can allow or disallow actions of the story world agents in order to guide the story in the intended

direction. Players can perform any action they would like, but the Director has the power to permit only actions that accommodate the plot script. This gives the Virtual Storyteller a large grain.

On the other hand, Façade uses a medium to small grain. Organization of the story beats is highly controlled by the system. Although the user is able to interrupt the beats, the characters rigidly follow the story beats chosen and the user has little influence in changing or avoided conversational subjects. IDA also has a small to medium grain size. The author has full control over the order of the story scenes and the events within those scenes. Within the scenes, the Director monitors and guides the user through the plot and even has control over characters in the story world. Similarly, the storyline and most actions are determined by the system in the GADIN system. Its grain size is small since the dilemmas and a step-by-step plan to achieve them is chosen by the system, and the story is simply presented to the audience.

2.5.3 Freedom to the User

Freedom to the user is not necessarily the amount that the user is able to move around and manipulate the story world, but to what degree they are able to impact the story through their own manipulations. With a high degree of freedom, the user's actions easily affect the course of the story, whereas a small amount of freedom means the user has little or no effect with their choices. A system with low freedom may appear to give the user choices, but will always attempt to guide them to a specific goal. As the level of freedom increases in a system, it must be able to handle an increasing number of situations, making the story more difficult to create and test.

A high degree of freedom is provided to the user in stories authored by the Oz system, the Virtual Storyteller, and the IDA system. In the Oz system, the user can change the order of plot points and can manipulate the story world. The drama manager offers little or no interference, and must adapt or try to persuade the user to follow the intended path. A user controlling an agent in the Virtual Storyteller has the freedom to perform any action since the system is in a simulation format. The user is given a high degree of freedom to interact with parts of the story world and change parts of the narrative, without restriction

from the Director. Similarly, the user is able to move freely and interact with the story world in IDA. However, in IDA, the Director will attempt to guide the user and prevent them from straying significantly from the intended storyline. Nevertheless, the user is still given a high degree of freedom since they are able to make their own decisions, even if they conflict with the story envisioned.

The Façade architecture appears to provide a high degree of freedom in the story world. Through the textual interaction and responsive nature of the characters, the user can perform actions within the story. They can takes sides with the arguing couple, and uncover problems in their relationship. However, the user has little overall effect on the conversational topics discussed during the gameplay. The story beats are highly controlled, and the user has little influence in steering the conversation in a particular direction. This limits the actual level of freedom of the user.

Users of the GADIN system also have limited influence on the overall story. Dilemmas presented allow the user to change story world values, such as affecting relationships. In this way, they are able to modify the story world. However, the user cannot directly choose the storyline in which they partake. The dilemmas are determined by system, as is the plan to achieve the dilemmas. The user can stray from this intended path, but the system will attempt to bring them back.

2.5.4 Ease of Authoring the Story/Creative Control

Ease of authoring the story or creative control refers to the extent that a system allows the desired story direction to be input, and can output a story matching that desire. A system with low creative control will not give an author the means to control the story. A system with high creative control provides a way for the author to control the story world and how the story unfolds. The degree of creative control does not necessarily dictate the degree of user freedom.

The Oz system, Façade architecture and IDA provide the author with a high degree of creative control. To describe major plot ideas, the Oz system uses pre-authored plot points. The author determines the most appropriate order of these points, and conveys

this information through an evaluation function. The author is given a high level of creative control, yet the user is able to uniquely influence the story. In the Façade beat system, the author uses Façade-specific language to provide detailed information for each beat, such as preconditions, weights, priorities, and the effect use of a particular beat will have on the story. The progression of the story is determined by the author's structured tension arc. Authors using IDA can create story scenes and desired plot states within those scenes. The Director presents the scenes to the user in the author's proposed order, and attempts to guide the user between the plot states to create the author's intended story. The author is given a high degree of creative control, potentially at the price of story variability.

Limited creative control is given to the author in the Virtual Storyteller and GADIN system. A basic story outline is in the Virtual Storyteller through the use of major plot scripts. Of the scripts, the Director selects 4 to create world agents with goals and models. These scripts are written by the author, and the author has control over how the story unfolds at a high level. However, the author does not dictate what happens within the plot segments since the agents have control over their own actions. The least of amount of creative control is given to authors using the GADIN system. Although the GADIN system supports the creation of dynamic and emergent stories, it does not allow the author to create a particular story. The system does not use plot points, but rather the stories are composed of clichés and story patterns. The creative control is limited to the story pattern the author inputs to the system, and the characters and settings created beforehand.

2.5.5 Pacing Control

Pacing a story properly is key in telling a good story. The pacing can determine whether an event happens too quickly or too slowly, and helps to control many dramatic aspects such as suspense or urgency. This metric describes if a system has a mechanism in place to control the pacing of a story. This can be done either by a build in method, or by allowing the author to provide timing information about certain story elements.

Pacing control is employed by Façade through a time limit placed on the beats. Beats are changed every minute. While a player can interject and interact with the story, the story continues from where it was interrupted to continue the story's pace. The short beat duration allows the current beat to induce the correct amount of tension required by the story arc, as the beats are changed often. The Virtual Director in the Virtual Storyteller implements pacing control by allowing or disallowing character actions. If a character is achieving their goals too quickly, the Director can slow them down. On the other hand, if the characters is not highly motivated, the Director can encourage them and append their goals to help them progress faster. IDA controls story pacing through use of its depth metric associated with each story state. The depth of a plot point dictates how long the Director must wait before the point is presented and how long the user has to achieve to achieve the state.

Neither the Oz architecture nor GADIN explicitly make use of pacing control. In the Oz system, the drama manger only intervenes during a major plot point transition. Otherwise, it remains inactive while the user is free to explore the story space, allowing the user to take as much time as desired to complete a task. Similarly, the progression of the story depends on the user and character actions in GADIN. Although the system limits the duration of time the user has to respond to a dilemma, the speed at which the story is told is not constrained.

2.5.6 Domain Dependency

If a system is domain dependent, then it cannot easily change the type of story that it can tell. Being able to conform to different story types or domains permits greater flexibility for authors creating story worlds to host their stories. Systems that are domain independent do not have an architecture that is limited to only a few story types.

Oz, IDA and GADIN are domain independent. Oz stories are able to be structured to fit any domain, only requiring the author to create an evaluation function to match the specific domain. Stories written with IDA must be written in scenes, which can be used for most story domains. GADIN allows for any domain to be specified by the author. The

author is required to input domain specific information into the system, and GADIN is able to create stories from those patterns or clichés.

The Virtual Storyteller can be considered both domain independent and dependent. This system requires a story to be in a style similar to the four act structure described by Propp [37]. Although the story must be written in this particular format, it is possible to adapt the Virtual Storyteller to a different domain.

Façade is domain dependent. Façade is structured so that it composes a simple one act drama depicting a dinner party. It does not support stories that require certain plot points to maintain a storyline, such as action or thriller stories.

2.5.7 Comparison of Evaluation Metrics

Table 1 contains a summarized comparison of narrative generation system evaluation metrics for each of the systems presented in Section 2.4.2.

Table 1: Comparison of Narrative Generation Systems

	Evaluation Metric					
System	Temporal & Spatial Consistency	Granularity	User Freedom	Creative Control	Pacing Control	Domain Dependency
Oz Project	None described Dependant on the author.	Large grain	High	Medium	No pacing control.	Domain independent.
Façade	Yes Dependant on the author.	Medium to small grain	Limited	High	Has pacing control.	Domain dependant.
Virtual Storyteller	Yes Maintained through agent models and action supervision by Virtual Director.	Large grain	High	Medium	Has pacing control.	Both domain independent & dependant
IDA	Yes Dependant on the author.	Medium to small grain	High	High	Has pacing control.	Domain independent.
GADIN	Yes	Small grain	Limited	Limited	No pacing control.	Domain independent.

Ideally, an interactive narrative generation system should: offer temporal and spatial consistency, flexible granularity, high freedom to the user, a significant level of creative control to the author, appropriate story pacing, and be domain independence. To date, no narrative generation architecture provides this particular combination of the six metrics used to review a story generation system. Each of the five systems will produce a consistent narrative. However, the systems that support high granularity, namely the Oz Project and Virtual Storyteller, only offer medium levels of creative control to the author. Additionally, both these system are not entirely domain independent. Of the systems that are domain independent, IDA offers high user freedom and creative control, whereas GADIN is limited in both user freedom and creative control. However, as mentioned, neither of these systems uses a high grain. Of the five systems, three offer different methods to control the pacing of the story. By combining the favourable attributes of each of these narrative generation systems, perhaps an improved architecture for creating interactive emergent narratives suitable for videogames can be developed. Such an architecture is proposed and introduced in the next chapter.

Chapter 3

3 Architecture and Implementation

This chapter describes the main components of the emergent narrative generation architecture developed in this thesis. A description of the design architecture is followed by a description of its implementation in a prototype game system created for testing.

3.1 Design Architecture

This section details the main architectural systems of the proposed architecture for implementing emergent narratives in videogames, and how these components work together to present a cohesive story that makes use of the Game World (Figure 1). The story presented consists of plot fragments, which the user experiences in a manner similar to that of missions or levels in a videogame. The fragments are pre-written by the author using the proposed architecture and contain preconditions that must be met before being presented to the user. The architecture also contains different systems that monitor and present the plot and Game World to the user, namely the Director, Evaluator and Controller. The Director is responsible for maintaining a number of lists. The Plot Fragment list holds all available plot fragments available to the system, while the Active Plot Fragments list are those which the system is actively pursuing but some preconditions are missing. The Trigger Plot Fragment list are those fragments which have no missing preconditions but have yet to be presented to the player. The Goals list contains goals for the system accomplish, comprising of missing preconditions from Active Plot Fragments.

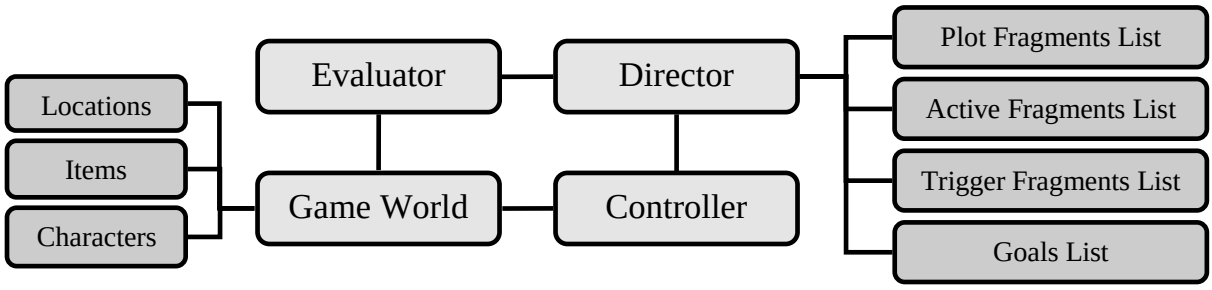


Figure 1: Major systems of the narrative generation architecture and the connectivity between them.

3.1.1 Game World

In the system's architecture, the Game World exists as the interactive area for the user. Its implementation can vary from a simple text description of the setting to a full 3D rendered free-roam world, or anything in between. The user can interact with the different mechanics of the Game World and the world should respond. Even if a game has no story, it can still have a Game World. Although these two parts do not necessarily have to be tied together, many mechanics of the Game World are often used in the narrative.

Common elements of a Game World include Locations, Characters, and Items. Locations describe the distinguishable areas within the game world, and can be occupied by Characters and Items. The Characters live in the Game World, and can be controlled by the game or by any player playing the game. Items are objects which populate the Game World, and can be interacted with by the Characters.

A key objective of the proposed architecture is to be able to integrate a narrative into the world of any game, rather than development requiring the Game World and narrative system to be strongly coupled. The Game World should be a fully functional entity on its own, consisting of its own game mechanics and characters, much like the real world. The narrative generation system should present plot fragments to the existing world, to be played out by the game's players.

3.1.2 Plot Fragments

The plot fragments presented to the user tell different parts of the story (Figure 2). They are the main components of the narrative, and each plot fragment consists of an intended goal for the player to achieve. However, the player can use the game mechanics to deviate from this proposed goal.

Each plot fragment contains preconditions that must be fulfilled prior to being presented to the player, and end conditions that modify some story world value. These story world values are end goal values for that particular plot fragment, and are used to plan the storylines presented to the user. Although these end value changes are intended goals, their achievement is not guaranteed. Additionally, the player is not necessarily aware of these goals and it is up to the plot fragment to guide the player, without forcing any actions. Plot fragments can guide a player to an intended goal through the use of in game elements, such as other characters giving directions, or in game events occurring.

Upon meeting its preconditions, a plot fragment will not be executed until its trigger condition is met. The trigger conditions are unknown to the player, and so they will not necessarily encounter the trigger. However, each fragment can contain Motivation for the game's Controller to present to the user as hints (Figure 2). With these, the Controller attempts to manipulate the Game World and encourage the user to initiate the fragment. These instructions can include, for example, a waypoint marker on a map pointing to a location of interest. Once the trigger condition occurs, the fragment is presented to the player. If a situation occurs where multiple plot fragments have their trigger conditions activated at the same time, the Director will select one of the plot fragments to present to the player. Only one plot fragment can be the Current Plot Fragment at once.

Plot fragments also contain instruction to be used later by the Controller when the fragment is presented to the player. These instructions stage actions and scenes in the Game World and setup the plot fragments in which the user plays.

A Description of what takes place within the plot fragment is included to help describe the events of the fragment. This can be used by the author to organize plot fragments

during story creation. In the prototype system described later in Chapter 4, the description is output to the user when a plot fragment begins to help instruct what Game World changes must be made before the user completes the plot fragment.

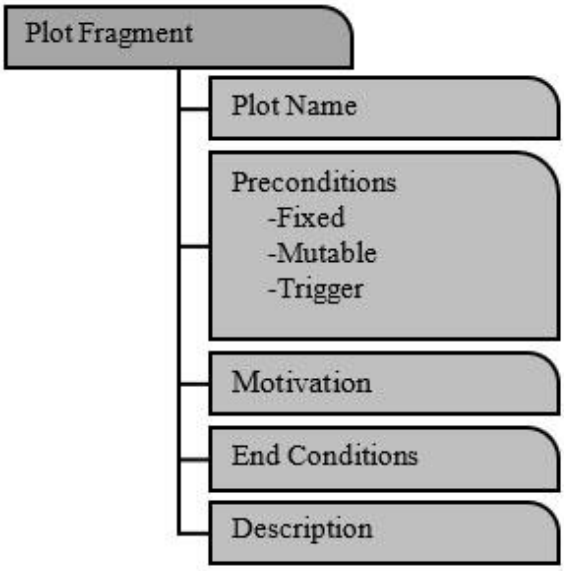


Figure 2: Plot fragment outline.

3.1.3 Conditions

A condition is an evaluation of a specific story world value, and is either set to true or false, depending on whether the requirements of the condition have been fulfilled. For example, the condition ‘inLocation Paul PoliceStation true’ is the condition that Paul must be in the police station. On the other hand, ‘inLocation Nick PoliceStation False’ requires that Nick is not in the police station. If this condition is checked while Nick is in the police station, then the evaluation of the condition will return false. Each plot fragment has a set of preconditions that is a list of Boolean conditions. In order for the plot fragment to be presented to the user, each of these conditions must first be evaluated as true.

Two different preconditions can be used in a plot fragment, namely fixed and mutable preconditions. Fixed preconditions are those that must be true in order for the story fragment to be considered. Once fixed preconditions are satisfied, mutable preconditions can be considered. Mutable preconditions can be unmet conditions that are added to the

Goal list. In other words, if all fixed conditions are true, the mutable preconditions are evaluated, and any mutable condition that evaluates as false becomes an active goal for the Director. The unmet preconditions are stored in the Goal List, and additional plot fragments may be necessary in order to meet these goals. To keep the story moving forward, it is the job of the Director to present additional plot fragments to the user to achieve these goals.

Because some conditions may be difficult to meet within the restrictions of the Game World, unmet fixed preconditions are never added to the Goals list. For example, in Figure 3 the fixed precondition is 'Simon is a Knight.' This plot fragment may be used in a game which allows the player to choose which type of role they would like to play, Knight being one of these choices. If another option in the game is to take the role of a Thief, then the character would not be able to play the 'Slay the Dragon' plot fragment, as the role of the character is unchangeable within the game.

However, the preconditions 'Simon has a Sword' and 'Simon has a Shield' can become goals for the system to achieve, so therefore are mutable preconditions.

As previously mentioned, each plot fragment also contains possible end conditions, which are Game World value changes that can occur once the player plays the fragment. For example, the plot fragment in Figure 3 has the two different end conditions of "The Dragon is dead' and 'Simon has Dragon Scales.' Any end conditions is a possibility, but not a guarantee, and the plot fragment is flexible in how it is played by the user. To this end, each end condition contains a *possibility* value that reflects the likelihood that a story world value will change. This value is used to plan a storyline, since some end conditions may be desirable but not likely. If the chances of Simon getting Dragon Scales from the dragon are low, then the system should know that this plot fragment should be a last resort in trying to satisfy the goal of getting dragon scales versus other plot fragments in the system.

Additionally, each plot fragment has a trigger condition that is evaluated once all preconditions are met. This special precondition is the last requirement that must be met before the plot fragment is presented to the user. For example, in Figure 3 the trigger

condition is for Simon to be at the Dragon Lair. Once all other preconditions are true, and Simon enters the Lair, the plot fragment will begin. A plot fragment can have multiple triggers, and only requires one to be true in order to exhibit the plot fragment to the Game World. The reason for this is to allow the game some flexibility when presenting the player with plot fragments. If the narrative can present a specific plot fragment in a number of different locations, such as one of the different Dragon Lairs, then this fragment should trigger once any of these trigger conditions are met.

On the other hand, a fragment can also exist without any triggers, which would begin as soon as all preconditions are met. An example of such a fragment would be a natural disaster that affects part of the Game World.

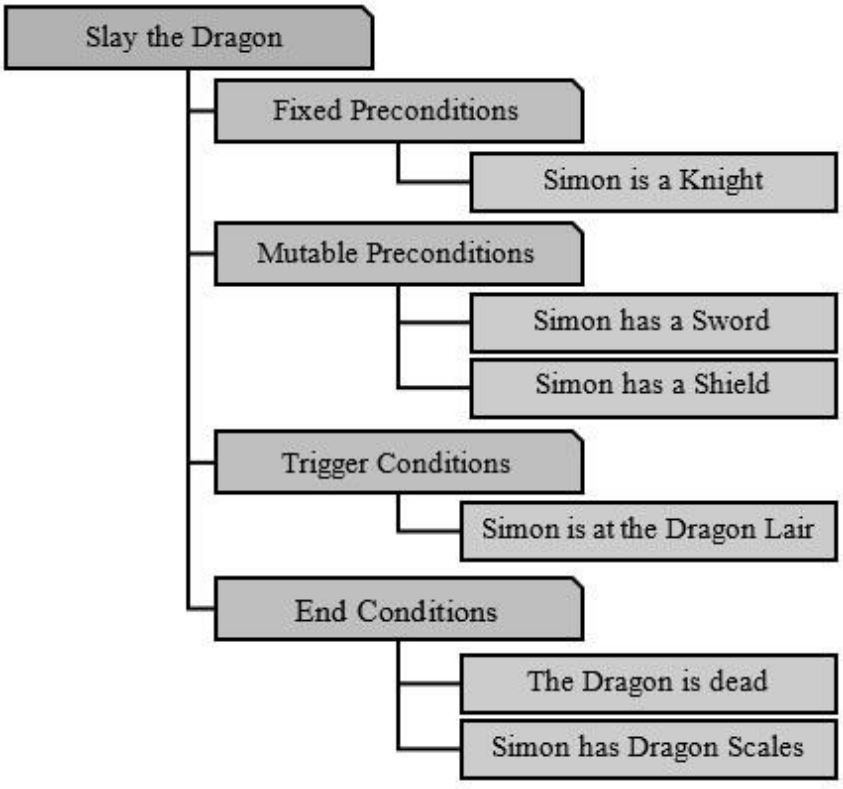


Figure 3: A sample plot fragment with different types of conditions.

Plot fragments are categorized by the number of preconditions that they have. With very few preconditions, a fragment is easier to achieve and considered a 'low-level fragment.' 'High-level fragments' are those with more preconditions, and require more low-level

plot fragments to achieve all preconditions. ‘Author-level fragments’ usually contain many preconditions, with the quantity depending on how much creative control the author wishes to have over the story.

3.1.3.1 Variable Parameters

A condition can also contain a variable parameter. It is left unknown intentionally because the plot fragment does not require a specific value. The value of the variable can be determined multiple ways, and can even be relevant to multiple conditions in the plot fragment. Characters, locations and items can be used as variable parameters within a fragment, and the system has to find suitable matches for these parameters in the story. For example, Figure 4 shows a sample plot fragment with 3 preconditions.

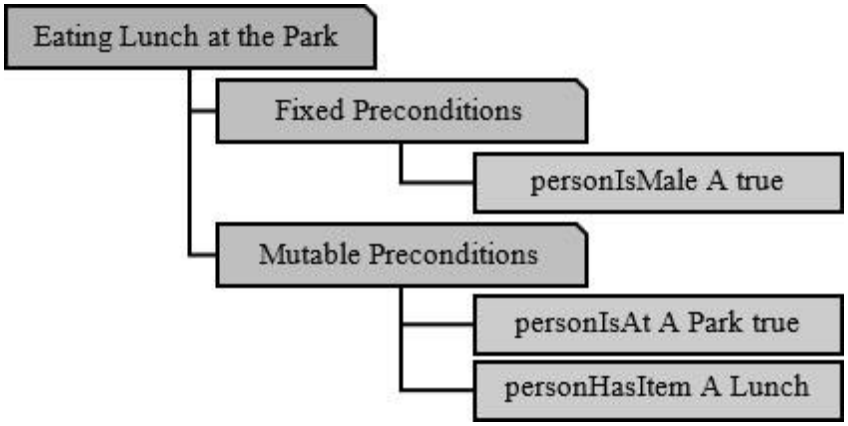


Figure 4: Sample plot fragment with variable parameters.

‘A’ is a variable and is used in a fixed condition. The Evaluator will return a list of characters which ‘A’ could take the value of to make each fixed condition true, and then one of these characters is chosen for the plot fragment. The mutable preconditions which evaluate false with this chosen ‘A’ character are then added as goals to the Goals list. Preconditions can also contain multiple variables. For example, ‘areFriends A B’ where both A and B are variables. The Evaluator then has to return a list of all possible combinations for A and B that fulfill the precondition.

3.1.4 Evaluator

The Evaluator of the system analyzes the current state of the story world. A condition is passed to the Evaluator, which returns the value of the condition in the current state of the Game World. All preconditions of a plot fragment must be checked by the Evaluator to determine if the plot fragment can be executed. The Evaluator does not continuously monitor changes in the story world, but rather evaluates the preconditions as needed. This provides an interface between the Game World that the player is in, and the architecture of the narrative generation system. A list of evaluation functions used in the prototype described in Chapter 4 is available in Appendix C.

The type of information evaluated by the Evaluator is limited to dual-option story values, since the system can only evaluate conditions as being true or false. However, the status of quantitative values, such as health or money levels, can be evaluated using the Evaluator through the use of conditions. For example, if a plot fragment requires the character Nick to have a level of health between 50 and 70, two preconditions can be used to evaluate whether this is true: 'hasHealthGreaterThan Nick 49' and 'hasHealthLessThan Nick 71'. Both preconditions will be fulfilled once Nick's level of health is in the 50-70 range.

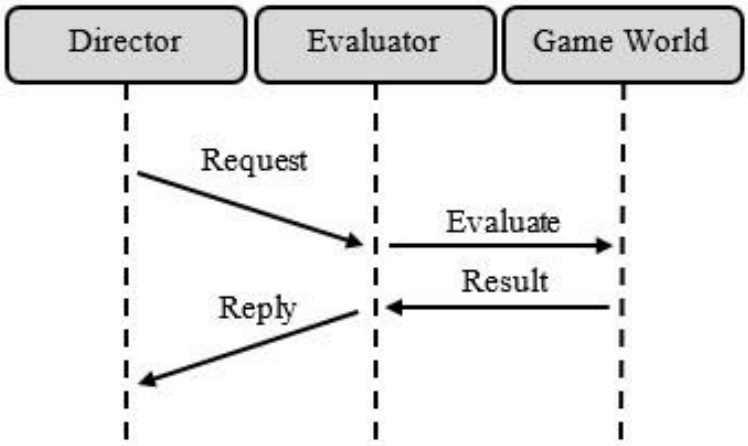


Figure 5: Synchronous Evaluator Operations

The Evaluator can be implemented in two different ways depending on the application. Using a synchronous implementation, as is used in the prototype described later in

Chapter 4, control is passed back and forth between the Director and Evaluator as conditions are checked in sequence. This is shown in Figure 5. The Game World values needed for the Active and Trigger Plot Fragment lists are evaluated continuously as they could be changed at any time. This method is less efficient but makes debugging narratives easier as the process of events happen in sequence.

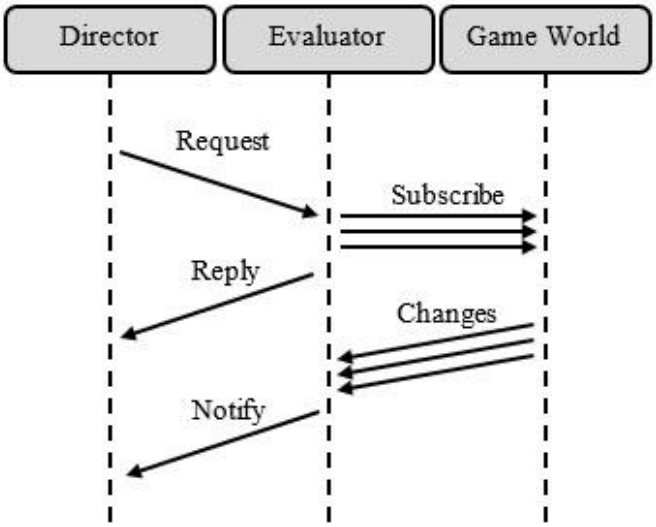


Figure 6: Asynchronous Evaluator Operations

In an asynchronous implementation, as shown in Figure 6, the Director requests the Evaluator keep track of the needed preconditions for a plot fragment. The Evaluator subscribes to these events in the Game World, and is notified when any of these values changes. Once all values for a plot fragment are changed as needed the Evaluator notifies the Director that the plot fragment is ready to be triggered, and the Director moves the plot fragment into the Trigger Plot Fragment list. This mode is more efficient because rather than constantly having the Director ask the Evaluator if every precondition for every plot is true, the Evaluator can wait until the Game World notifies of any desired changes.

3.1.5 Director

The Director is the core component of the architecture, and is responsible for managing plot fragments and goals. The Director has control over which plot fragments are presented to the user by giving plot instructions to the Controller. As well, the Director

must determine which goals need to be accomplished by consulting the Evaluator system. It is the job of the Director to make sure unmet goals are accomplished before a particular plotline can be presented. The Director begins by loading the author-level plot fragments into the system. These fragments are the major plot points and are used to guide the story in the author's intended creative direction. To maintain order of the fragments, each subsequent author-level plot fragment uses the completion of the previous fragment as a precondition. The life cycle of a plot fragment is shown in Figure 7.

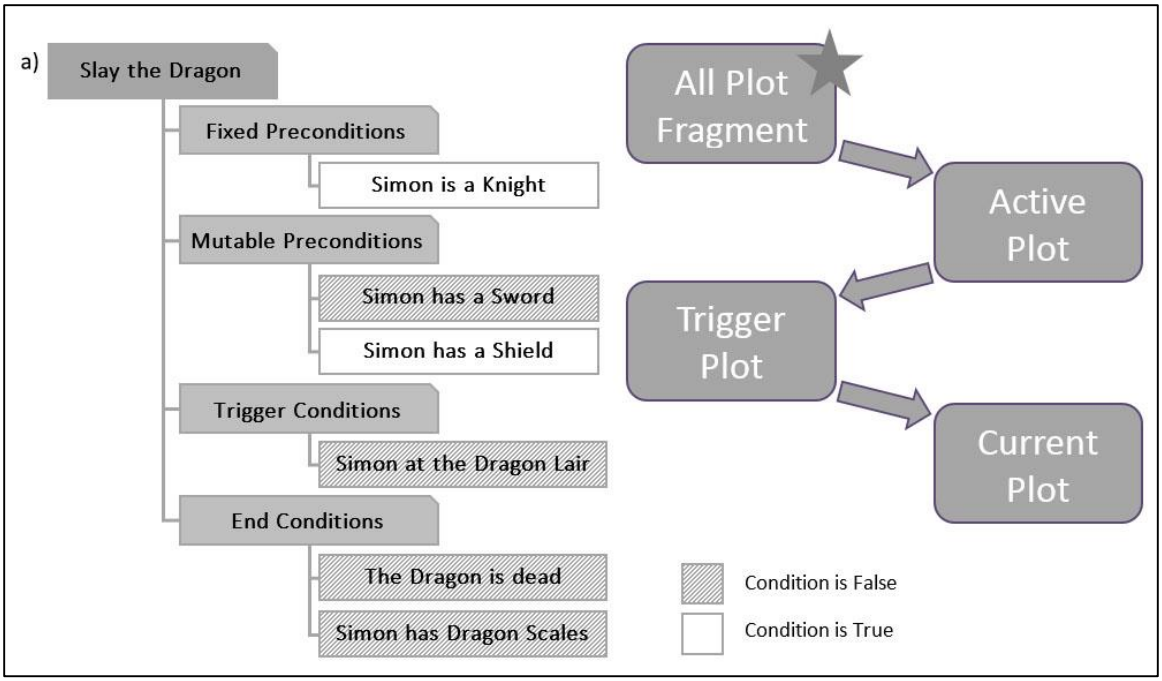
Plot fragments become 'active' by the Director when they are needed to pursue a goal of the system (Figure 7(b)). They remain in the Active Plot Fragment list as long as they have unmet preconditions that the system is actively trying to fulfill. Once all preconditions are met, the fragment becomes a 'trigger plot fragment' and the system waits for the plot fragment's trigger condition to be met. Once the trigger occurs, the player is presented with the plot fragment.

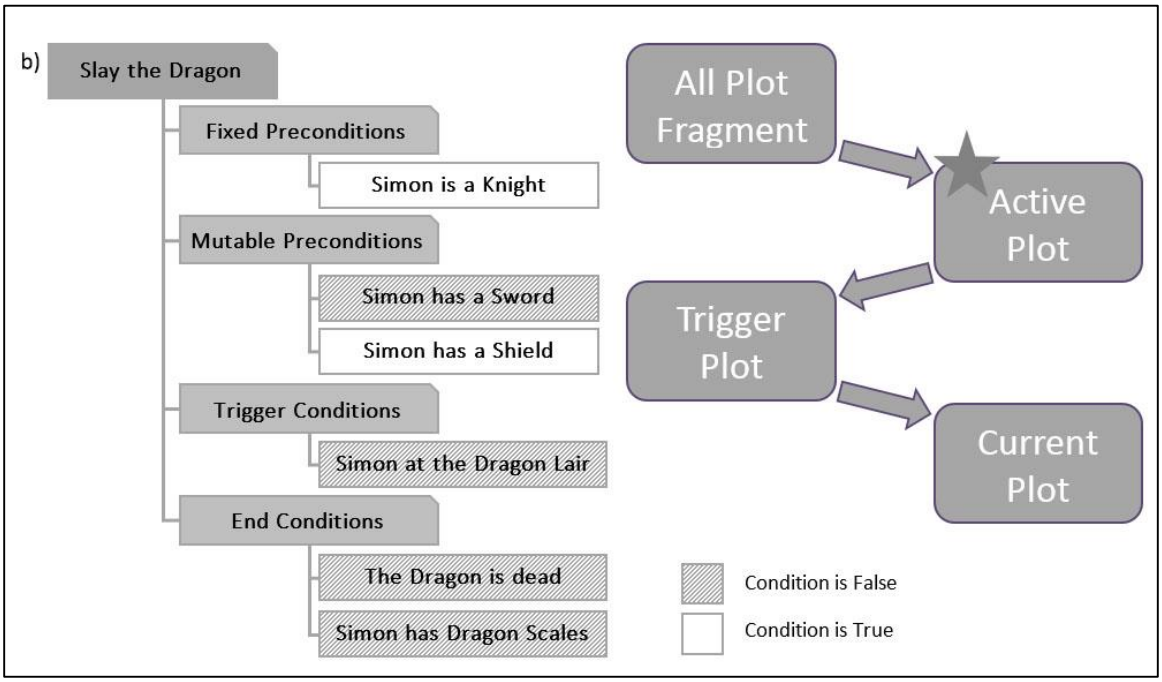
The Director has two main jobs concerning plot fragments and story goals. First, the Director is responsible for creating goals from missing preconditions. Second, the Director evaluates all current goals to determine if they have been fulfilled or require active plans to fulfill them. To do this, the Director first checks all plot fragments in the Active and Trigger plot fragment lists to determine if (a) the plot fragment is needed anymore and (b) there are any missing preconditions. A plot fragment may become unnecessary if a change in the Game World occurs induced by a player's actions. In this situation, an Active Plot Fragment intended to solve a specific problem is no longer needed and should not be presented to the user. Therefore, it should be eliminated from either the Active or Trigger Plot Fragment list. The Director determines the relevance of a plot fragment by comparing its end conditions with the Goals list. If any end conditions match a goal on this list, then the particular plot fragment is still relevant. Otherwise, the fragment is no longer necessary. For example, if a character Nick needs a gun to start a bank robbery, then the goal of 'hasItem Nick gun true' is created and added to the Goals list. Next, a new plot fragment that gets Nick a gun is added to the Active Plot Fragment list in order to fulfill the goal. However, if Nick acquires a gun before the new fragment

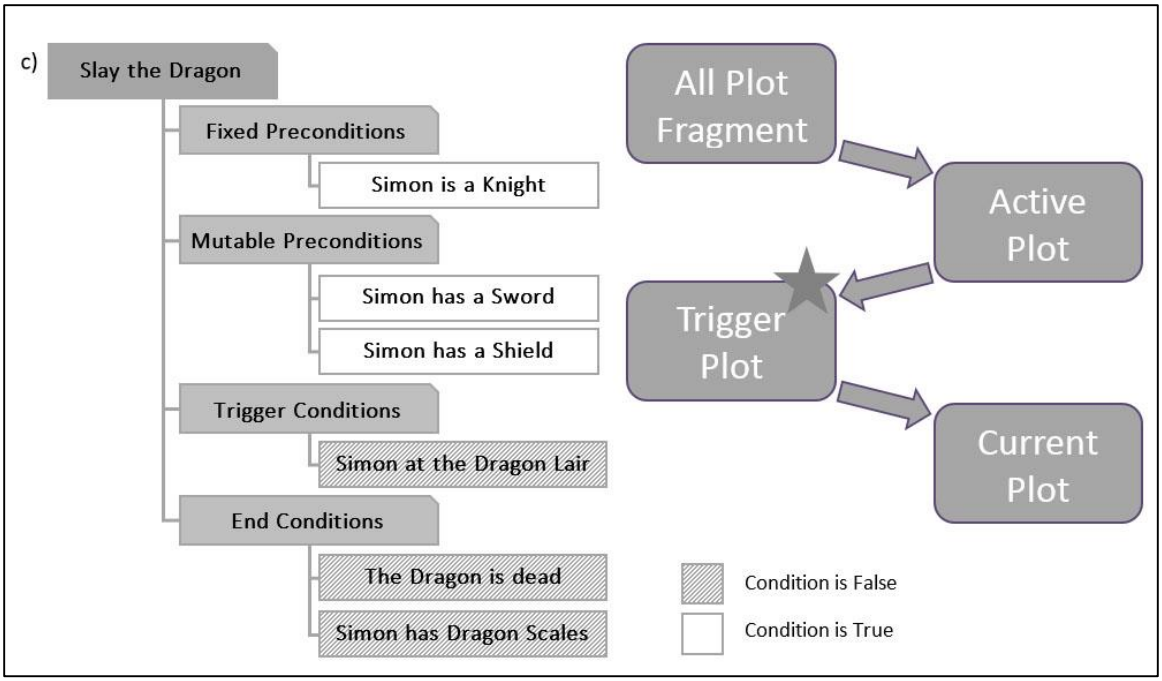
is presented, then this fragment is no longer needed and should be removed from the Active Plot Fragment list.

To determine if the any missing preconditions must be added to the Goal List, the Director must check through both the Active and Trigger Plot Fragment lists. Although all of a plot fragment's preconditions must be fulfilled before becoming a trigger plot fragment, it is possible that a player can manipulate the Game World so that the plot's precondition change before the plot fragment is presented to the user. Once every plot in the Trigger list is checked, the plot fragments in the Active list are checked for missing preconditions, and these missing conditions are added to the Goals list.

The Director's second job relates to solving goals created by missing preconditions. Once a goal is fulfilled, it is removed from the Goals list. However, for goals that remain unfulfilled, the Director must check if any Active or Trigger fragments are pursing these goals. This is done by checking the end conditions of the plot fragments in these lists. If no plot fragment's end condition matches the goals, then a new one must be chosen from the database of plot fragments. A goal may have no plot fragment pursuing it if, for example, a previously played plot fragment failed to achieve this goal or if this goal was only recently added to the Goals list.


a)
Slay the Dragon
Fixed Preconditions
Simon is a Knight
Mutable Preconditions
Simon has a Sword
Simon has a Shield
Trigger Conditions
Simon at the Dragon Lair
End Conditions
The Dragon is dead
Simon has Dragon Scales
All Plot Fragment
Active Plot
Trigger Plot
Current Plot
Condition is False
Condition is True


b)
Slay the Dragon
Fixed Preconditions
Simon is a Knight
Mutable Preconditions
Simon has a Sword
Simon has a Shield
Trigger Conditions
Simon at the Dragon Lair
End Conditions
The Dragon is dead
Simon has Dragon Scales
All Plot Fragment
Active Plot
Trigger Plot
Current Plot
Condition is False
Condition is True



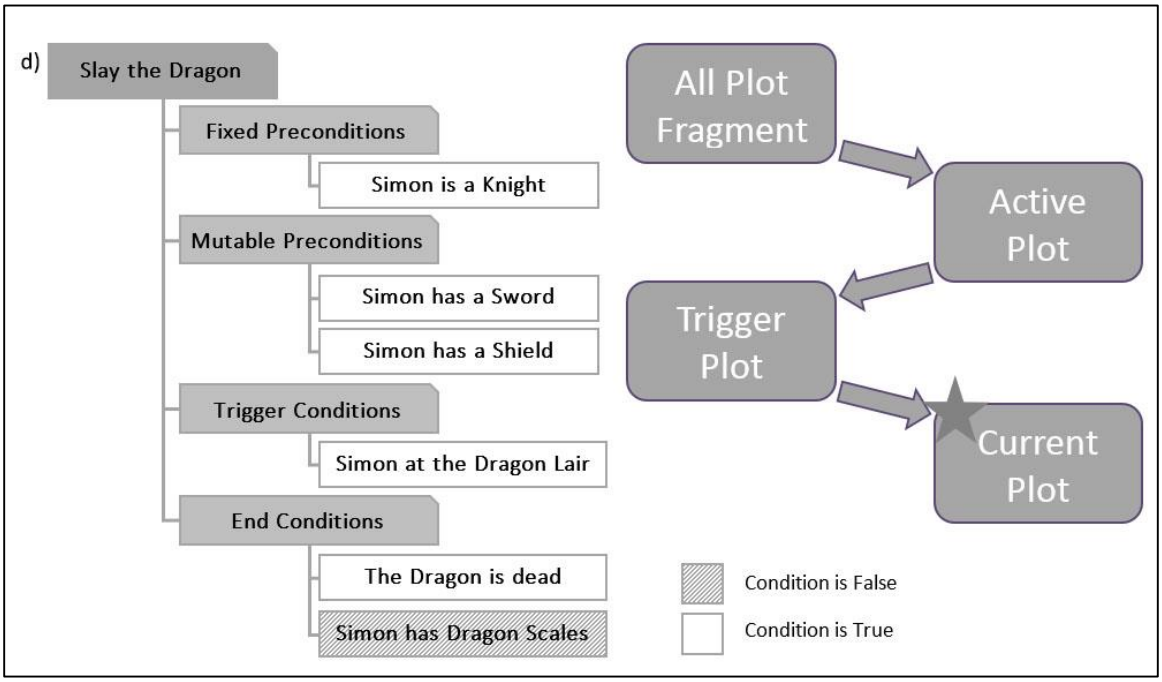


Figure 7: Life Cycle of a Plot Fragment (a) The Plot Fragment 'Slay the Dragon' begins in the All Plot Fragment list. The director checks that the Fixed Preconditions are met and activates the Fragment, moving it to the Active Plot list. (b) While in the Active Plot list the director will attempt to achieve the Mutable Precondition 'Simon has a Sword.' (c) Once all Mutable Preconditions are met, the Director moves the Fragment to the Trigger list. (d) Once the Trigger Condition becomes true, the Director will present the Fragment to the Player. In this case the Player killed the Dragon but did not retrieve the Dragon Scales.

3.1.5.1 Choosing a Plot Fragment to Pursue

In order to select a new plot fragment to fulfill a goal, the Director must consider a number of criteria. First, the number of potential goals a plot fragment can fulfill must be considered. If multiple goals can be met with a single plot fragment, different storyline can be intermixed, which is a favourable outcome. Second, the number of attempts for a particular goal must be taken into account. This metric can be used to adjust the game difficulty. If a goal is being attempted for the first time, a more difficult plot fragment may be presented than if the goal has been previously attempted. As the number of attempts increases, the difficulty of plot fragments will decrease in order to increase the player's likelihood of fulfilling the goal. In the worst case, a plot fragment that guarantees the goal is met is necessary when it takes too long for the goal to be met. Third, the number of preconditions must be considered when selecting a plot fragment. If the Goals list is populated with several unmet goals, then a plot fragment with only a few unmet preconditions is preferable. This prevents too many story lines from occurring at once and having none conclude. If the Goals list only contains a limited number of unmet goals, selecting a plot with a larger number of unmet conditions is beneficial in order to populate the Goals list. This initiates more stories, thus creating more activities to complete in the Game World.

Using the author-level plot fragments as seed fragments, the Director compiles a list of goals and plot fragments to present to the user. Although the author-level plot fragments are major story plot points used to guide the story in a particular direction, the Director can use any combination of fragments in the database to achieve all preconditions of the fragments.

The database of fragments should include a variety of fragment types to ensure a minimum number of fragments are always in the Trigger Plot Fragment list. A plot fragment is a high-level fragment if it contains many preconditions, while a low-level fragment has a small number of preconditions. When initially selecting fragments to fulfill the author-level goals, the number of goals on the Goals list will be sparse. The Director begins by choosing higher-level plot fragments to populate the list. In selecting fragments of this variety, more goals are created for which the Director can work. As

more goals are added and the Goals list fills, the Director then begins to choose lower-level plot fragments to avoid adding too many goals to the system. The lower-level plot fragments will be easier to achieve because they will likely have less preconditions missing. This hierarchy allows for players to play lower-level plot fragments to achieve the preconditions of the higher-level plot fragments, and allows the story action to naturally build toward more involved plotlines.

3.1.5.2 Side Quests/Side Missions

The Director may come across plot fragments which do not directly relate to the main narrative, and can be presented to the user at any time. For example, a side mission may be to retrieve a special weapon from an abandoned factory. This type of plot fragment would have no preconditions, and only a trigger condition of being in the abandoned factory. The Director places these plot fragments directly in the Trigger Plot Fragment list, and the player simply needs to trigger the side quest. Side quests can also be linked together chronologically. This can be accomplished by using the completion of plot fragments as preconditions for each other. For example, the completion of the 1st plot fragment chronologically would be a precondition for the 2nd plot fragment, and so on. This would allow the main narrative to continue normally in between these individual side question fragments, and allow the player to completely ignore the rest of the side quest if they wish.

3.1.6 Controller

The Controller is an active system that interacts with and manipulates the Game World according to the plans of the plot fragments. Using the Evaluator system, the Director evaluates the trigger conditions for the plot fragments in the Trigger Fragment list. If any of the trigger conditions are fulfilled, the Director signals the Controller to present the plot fragment associated with that trigger. The Controller uses the instructions provided with the plot fragment to manipulate the Game World and present the fragments to the player in the form of levels or missions.

The Controller represents the predetermined Game World staging sequences that are common in missions or levels in the game. For example, if a player can walk onto a

marker in the open Game World to trigger a mission, the Controller is responsible for manipulating the Game World in order to begin the mission. This may include initiating a cut scene and setting up the situation to indicate that a mission is about to begin to the player. The Controller is also responsible for actions and changes that occur during the mission, such as new characters appearing and shooting at the player, characters yelling at each other, *etc*. The Controller directs the aspects of the Game World that allow the player to feel as if they have an active role in the narrative.

3.1.7 Pacing

Pacing control can be implemented into the architecture to some degree with the use of preconditions and time variables. By adding a fixed precondition to a plot fragment which requires a minimum or maximum amount of time before the fragment can be presented, the author can control a time frame when the plot is capable of being presented to the player. This can control the time between plot fragments and allow the narrative the chance to slow down when moving too quickly and speed up when too much time has gone by. Although this method does not give the author complete control over the pacing of the story, it does allow for some control in situations that would otherwise be entirely up to the actions of the user or randomness of the narrative selection process.

By using a precondition which requires a minimum amount of time from the previous plot fragment the author can prevent an ambitious player from completing the plot fragments too quickly and making the narrative seem unrealistic. For example, if a sequence of plot fragments involves the same character, slowing down the player may be necessary in order to allow the illusion that the non-player character was able to run some errands or move location in between plot fragments.

This can also work with players who are moving too slowly or exploring too much. With a precondition that monitors for a player taking too long, a plot fragment can be triggered which can be used to motivate a player to move along faster. For example, if a player is exploring an area of the Game World for a long time and they are needed at a certain location to trigger a plot, a plot fragment can be used which causes the player to get chased by enemy players in order to move them along.

3.1.8 Branching Plot Fragments

An author can control the main direction of a video-games narrative through the selection of author-level plot fragments. The Director will work to achieve them and keep the narrative on track. The author can also make the choice to include branching author level plot fragments which can add more variety to the narrative, and allow the player to have some influence with their actions. A branching plot fragment will present an opportunity for the player to make a choice, changing some Game World value. Based on this change, the Director can pursue a number of different author-level fragments, changing the course of the narrative.

For example, in Figure 8 there are two example branching author-level plot fragments. These can be used in a situation where the character Neo is given the choice to choose between selecting the *Blue Pill* or the *Red Pill*. Since Neo is only allowed to choose one of these items only one of these plots will ever be considered by the Director because they have conflicting fixed preconditions. If Neo chooses the *Red Pill* then the 'Chose Blue Pill' fragment will never pass the fixed precondition stage and will never be used.

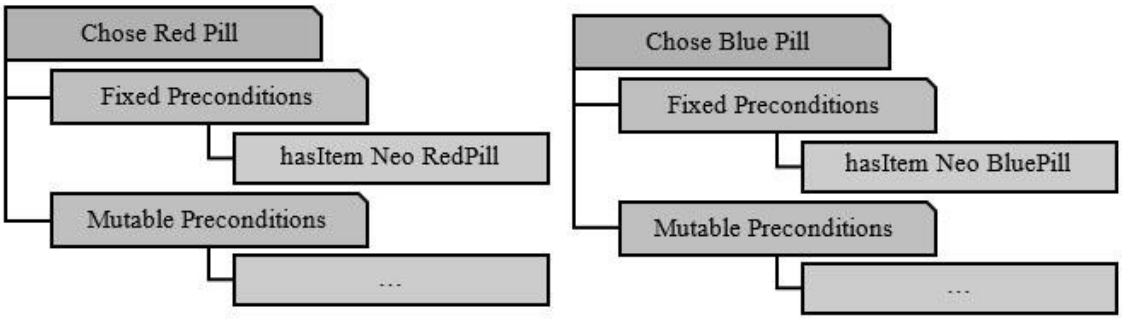


Figure 8: Branching Author-Level Plots

Chapter 4

4 Prototype Implementation

A prototype implementation of the narrative generation architecture was developed to demonstrate its efficacy. The prototype game system was developed using the C++ [43] programming language, and the graphical user interface was developed with the Qt [44] framework. Some Boost [45] C++ libraries were used for text processing and manipulation.

The prototype game system uses plain text files to load all Game World values and plot fragments. Each text file requires a specific format in order to load all information into the system. This method of implementing information was used in the system as it straightforward and suitable for a preliminary system. A more sophisticated method that minimizes loading time and restricts player manipulation of game files should be employed in future generations of the system.

The system begins by loading and storing all games world values, which includes characters involved in the story, items in the world, world locations, and the relationships among these story world aspects. This is followed by loading the plot fragments into the system, at which point the user selects which fragments would be considered author-level fragments. These are the main fragments the system will work to achieve using the selection of other plot fragments available.

The implementation is a representation of the underlying framework discussed in the architecture of Chapter 3. The graphical user interface allows the user to simulate a video game by manipulating the Game World and setting up situations to test plot fragments to be used in the videogame. The prototype can also be used as a useful tool to allow authors to test the narrative of a story without the need to integrate with the rest of the game. This can be helpful when debugging storylines and deciding which goals may need more plot fragments to fulfill them. Once a set of plot fragments has been tested in the development environment it can be fully integrated into the game.

4.1.1 Characters

The characters in the system employ a simple player model capable of having a number of minor attributes. The attributes currently supported are those that are common among videogames, and are therefore characteristics a designer would likely include in their game. A total of 11 different character attributes are supported by the system, and include:

- Name
- Gender
- Mood (value from 0-10)
- Money
- Health (value from 0-100)
- List of items held
- List of Relationships with other players
- List of actions that the character could do
- List of actions that could happen to the character
- Story Role (value representing what this players role in society is)
- Boolean describing if they are the main character (only one can exist)

Using text files, the attributes for each character are loaded in the system upon initiation. A character is loaded from the file beginning with a line indicating that lines following describe character attributes. This line has the following format:

<Person> Name Gender(**0** male, **1** female)

When loading a character, all items and relationships the character has are listed after them in the load file, as well as all the other character values to be applied to them.

The character's name is its identifier within the construct of the system, and must be unique. With the character's name, the system's Evaluator can check the status of any of a character's attributes and evaluate any conditions pertaining to plot fragments or goals. For example, to check if Nick is holding a gun, the Evaluator evaluates the conditions 'true isHolding Nick gun'.

4.1.2 Items

Characters can pick up, hold, and lose items available in the story world. Each item has the following three attributes:

- Name
- Action this item can do
- Actions this item can have happen to it

When loading an item using the load file, an item is identified using the following format:

> <Item> Name

The Evaluator can analyze conditions involving items, such as whether or not an item can perform or receive an action. For example, to evaluate if a wooden plank can be used to attack, the Evaluator checks the condition ‘true canDoAction woodPlank attack’.

4.1.3 Actions

Actions include those that a character or item can perform, or those that can be performed to a character or item. In the load file and action is listed below a person or item to indicate who or what has the action ability. The format for loading an action is:

> <Action> name type(**do** if this action can be performed, **get** if this action happens to this person/item)

Additionally, the actions a character must perform may require a particular item, and so the character must be matched with a specific item that meets certain conditions. In this case, the item can be left unknown by using a selecting variable in the condition statements of a plot fragment. This gives the system more freedom in selecting items and plot fragments. For example, the system might want to use a plot fragment where a character can juggle as in Figure 9.

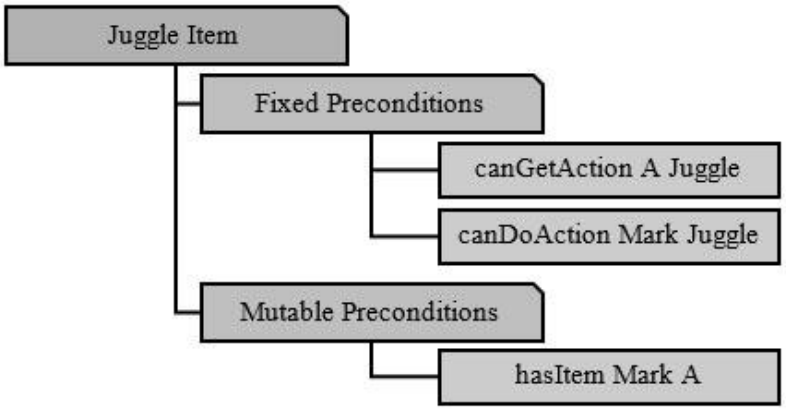


Figure 9: Action Plot Fragment Example

In this plot fragment A is a variable, and must be an item in this case. These preconditions are passed to the Evaluator, and any items that fulfill these conditions (*i.e.* items that Mark has and can be juggled) are returned to the Director, and can replace A in this plot fragment. Any items that meet these conditions, whether they are expected items such as tennis balls, or unexpected items such as bananas, can be returned, so long as the item has the correct actions set. This enhances the dynamic nature of the emergent gameplay, and allows the Director to create a variety of narratives based on the state of the Game World.

Preconditions that evaluate the actions of an item are typically fixed preconditions since these properties are unchangeable. If a character is able to learn a new skill, this can be added to the character's action list. However, if no plot fragment exists to teach this new skill, the Director will never find a way to fulfill this goal.

4.1.4 Relationships

A relationship between two characters is modeled using the relationship type and a value representing the strength of the relationship. The relationship type is selected from a list of common relationship, and for this system, include:

- Parent
- Child
- Sibling
- Married
- Family

- Significant Other
- Strangers
- Friends
- Enemies
- Teacher
- Student
- Working
- Community Member

For the current system, the value of a relationship can take one of three possible values, namely bad, average, or good. For example, in the prototype Nick and Paul are police officer partners, therefore Nick's relationship with Paul is 'Nick working Paul good.' Although a more complex system should ideally support a large spectrum of relationship values, only three discrete values of relationship strength are supported in this preliminary system to demonstrate this character attribute.

The Evaluator evaluates the type and strength of the relationship value between two characters. Each character maintains a relationship with every other character, but the relationships are not necessarily mutual. For example, an unreciprocated relationship would be the case when deception is involved. If character A is only pretending to be a friend to character B, the relationship according to A would be 'enemy' while to B, it would 'friendship.' Additionally, the relationship type listed for each character may not be identical for other relationships, such as the parent-child relationship and the student-teacher relationship, since the distinction of who plays which role must be made.

A relationship is listed below a character declaration in the load file. The relationship declared is then added to the character it is listed under only. The format for a relationship is as follows:

<Relationship> NameOfPartner Type Strength

4.1.5 Story Role

A character is given a specific Story Role, which describes their role in the narrative. This value is a label that describes to the Director which role the character has in society,

and is used in plot fragments that require character to play specific roles. For this system, 8 unique roles were used including:

- Shopper
- Shop Owner
- Bartender
- Police
- Mobster
- Banker
- Gang Member
- Dock Worker

These roles were selected based on the narrative theme being used, and can be expanded to include any real world society role. When loading a character, the Story Role is listed below the character declaration in the load file. The format for a story role is as follows:

<StoryRole> Role

4.1.6 Locations

Characters in the open story world can roam around from one location to the next. Each location in the story world has a list of other locations that are connected to it, which creates an interconnected graph that can be traversed. For simplicity, the current system only supports major areas of a city and the major areas not modeled in finer detail. For example, streets are major hubs, and are interconnected to all stores and buildings on that street. The buildings on the street are modeled in the system. However, details inside of the building, such as individual rooms or hallways, were not modeled in the current system.

Locations must store which characters or items are in that particular location. This information is used by the Evaluator when evaluating plot fragment conditions. Locations are also a component of many trigger conditions in the system, which allows dynamic stories to occur as the player moves from one location to the next.

Locations are loaded into the system by first declaring the location using this format:

<Location>name

Following this declaration separate locations can be added which are connected to this location using the following format:

<GoTo>name

The system will first check if this location exists. If the location used in this line does not exist it will be added as a new location to the system. If it does exists, then these two locations will be connected.

When loading characters or items in the load file, the use of a <Location> tag will set the current location of that character or item.

4.2 Using the System

The system is initialized by loading all the Game World parts, including characters and items, and all locations in the Game World. The graphical user interface contains four main tabs (Game World, Plot Fragments, Director, and Output) which the user navigates through. These tabs provide information about the state of the narrative. An overview of the Game World and the interaction between its components it provided to the user through the Game World tab, shown in Figure 10. The user manipulates the Game World through this interface. The user acts as the Controller, and is able to manipulate the story world values pertaining to the characters, items, and locations.

When the system is initialized, characters can begin with items and relationships, as well as default character values. The user of the system can manipulate the characters to give them items and change any values. Items can also be manipulated through the interface by changing which character has each item and the actions that the item is able to perform or have performed to it. This provides the user with the ability to create any situation in the Game World that they desire.

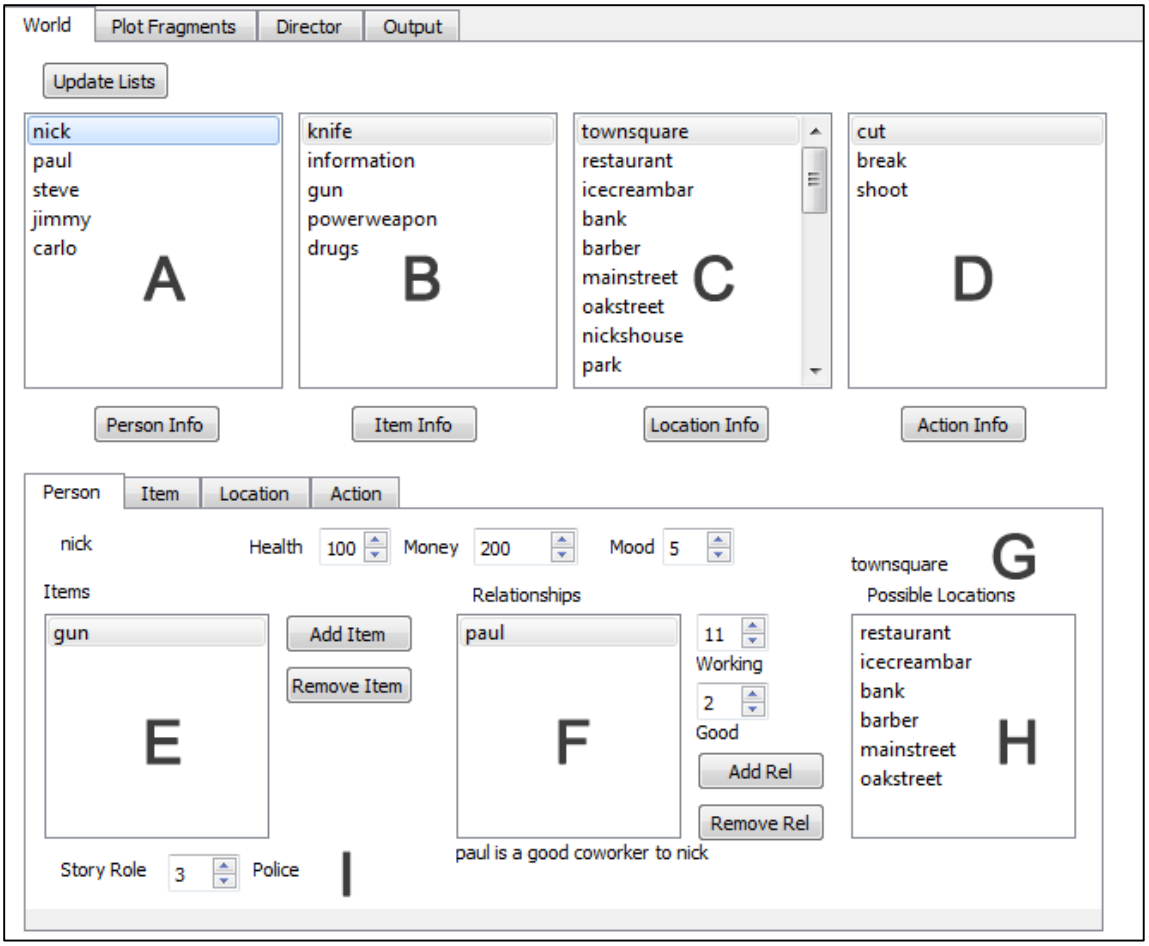


Figure 10: Game World Tab view Within the Game World, the user is presented with (A) a list of characters, (B) a list of world items, (C)a list of locations, (D) a list of possible actions, (E) a list of items held by selected character, (F) a list of relationships of selected character, (G) the current location of selected character, (H) a list of possible locations selected character can travel directly to, and (I) the story role of the currently selected character.

This simulation format allows the user a great degree of Game World manipulation and a focus on the narrative generation of the system. Although it is not an ideal representation of a game level or how a player would actually play a game, it accurately portrays the results of a game level and the changes that occur in the Game World as a consequence of a plot fragment. For example, in a fully developed videogame, the player can move their character from one street to another. In this system, moving a character is indicated by changing the character's location attribute. These types of change can be performed for any character, and simulates a player's actions as they play the game as well as those of non-player characters.

Figure 11 depicts a general user interaction with the prototype system, and the operations the system performs in reaction to the user's actions.

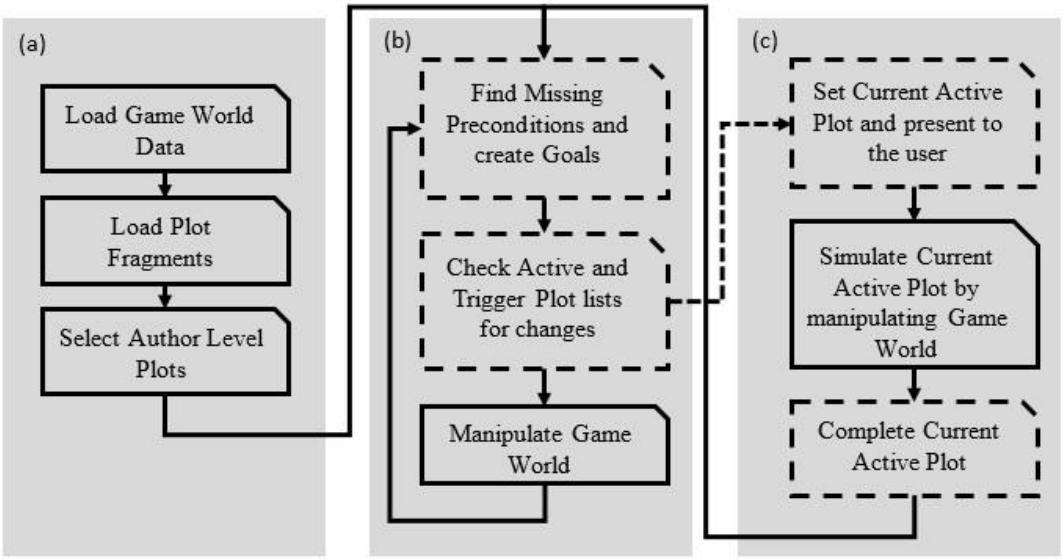


Figure 11: Workflow of the prototype system. Dashes boxes represent system operations. Solid boxes represent user interaction. Section (a) represents the preliminary loading steps performed by the user. Section (b) represents the main active loop, which takes place when the user simulates a player interacting with the Game World and the system reacts to these changes. Section (c) represents the presentation of a plot fragment to the user, and a return to the main loop once the plot fragment is complete.

After initializing the characters, items and locations, the plot fragments are loaded into the system using the Plot Fragments tab Figure 12. Within this tab, the user must select which fragments are to be used as author-level fragments (Figure 11 (a)). From the selected fragments, the Director determines which preconditions are missing, creates necessary goals, and chooses which plot fragments are needed to achieve these goals (Figure 11(b)). This is done automatically, and the list of goals is made visible to the user as shown in Figure 13(b). The Active and Trigger Plot Fragment lists are also available for the user to view. The Active Fragment list shows which plots the Director is actively pursuing. The Trigger Plot Fragment list shows the fragments that have met all preconditions and are waiting for their trigger conditions to be met. These lists cannot be modified by the user, and are only maintained and modified automatically by the Director. Once a fragment is triggered by a user's action, it becomes the current plot and is presented to the player (transition from (b) to (c) in Figure 11).

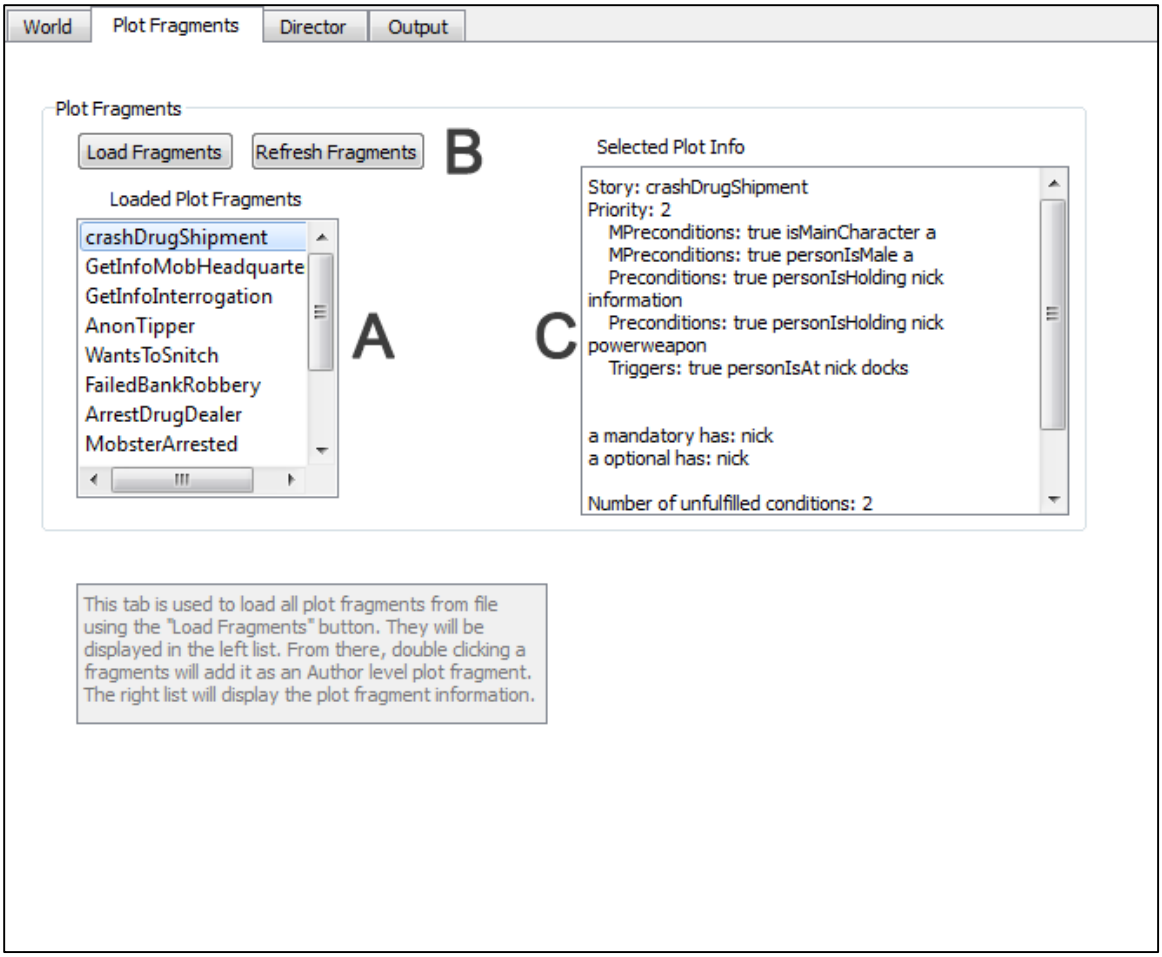


Figure 12: Plot Fragments Tab View (A) is a list of all loaded plot fragments. Double clicking a plot fragment will add it as an author-level fragment. (B) Buttons allowing the user to load plot fragments and refresh the currently loaded fragments. (C) A display of the currently selected fragment.

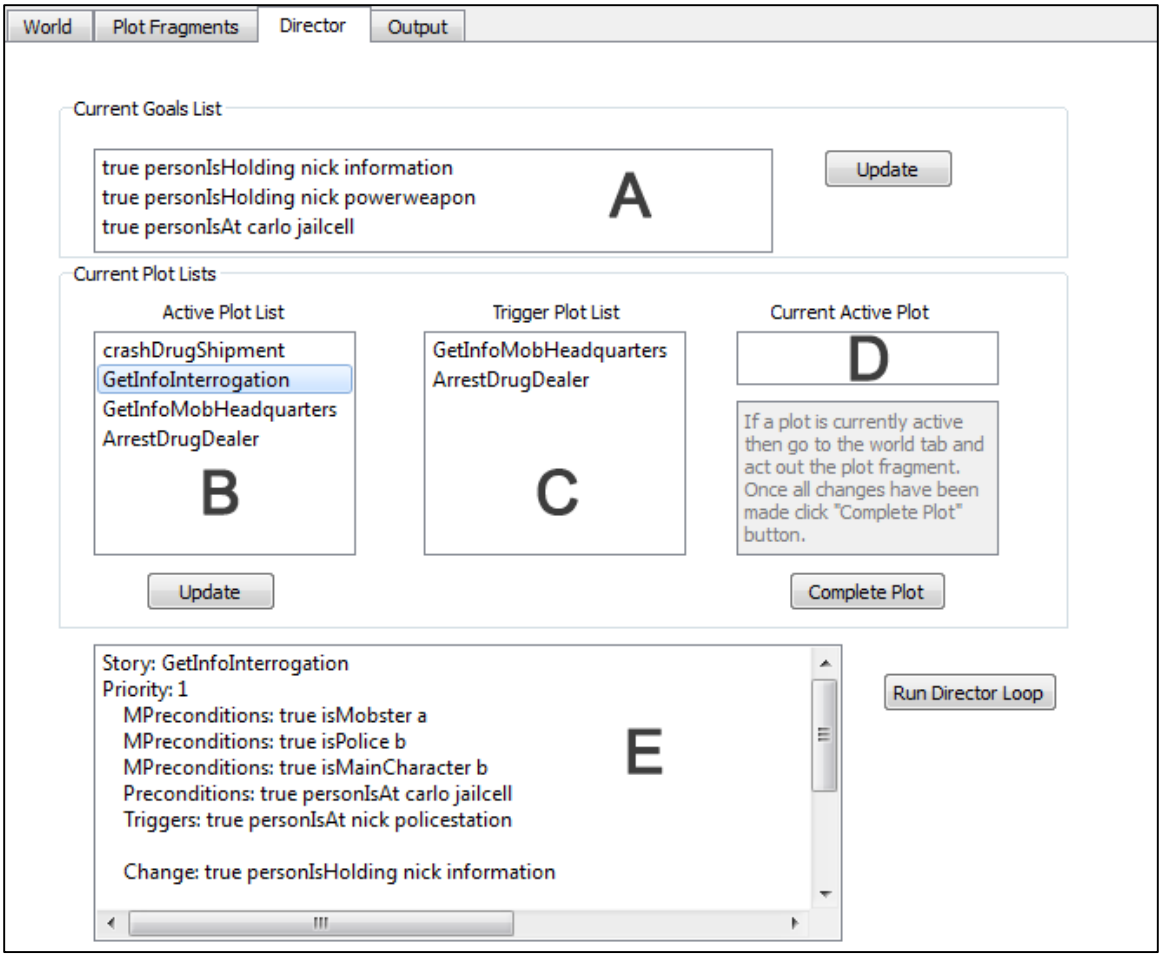


Figure 13: Director Tab View. The Director must maintain (A) a Goals list, (B) an Active Plot Fragment list, (C) a Trigger Plot Fragment list, and (D) the current active plot view. Information pertaining to the currently selected plot fragment from any of the lists is also displayed in the Director view (E).

When a plot fragment becomes the current active plot, the user simulates the execution of the fragment. This current plot fragment represents the beginning of a level or mission in a real videogame. A short description of the plot is provided with each fragment, and the user must then interact with the graphical interface to manipulate any characters or items involved in the fragment. For example, if the character needs a particular item, the user must give the character the item, or if the health of another player must decrease, the user must do this. Once all the changes have been made successfully, the user presses the 'Complete Plot' button to indicate they are finished with any Game World changes. The current plot fragment is complete and the Director takes over again (transition from (c) to (b) in Figure 11). The Director must evaluate whether any system goals have been achieved, and remove them from the list. For unmet goals, the Director must find another

plot fragment to put into the Active Plot Fragments list in another attempt to achieve these goals, and the cycle begins again.

4.3 Ensuring the Narrative Continues

The author of the narrative is responsible for creating a number of plot fragments with varying degrees of difficulty to ensure the game can adapt to any player actions and provide a continuous narrative. The Director begins by choosing a higher priority plot fragment to achieve a particular goal. The end conditions of plot fragments are rated by their likelihood of being achieved, and a plot fragment has a higher priority if it is likely to achieve the goal the system is attempting to accomplish. If the user fails to achieve the goal with the current plot fragment, the Director can choose to select a lower priority fragment to present to the user next time. Due to the limit on available plot fragments, the system is forced to select lower priority plot fragments each time the user fails to achieve the intended goal until eventually, the lowest priority fragment is selected. The lowest-priority fragment guarantees the player can fulfill the intended goals, and in turn, that the story will continue. Otherwise, if the Director is forced to keep picking new plot fragments for the player, the plot fragments would either run out or need to be repeated eventually. If the plot fragments run out, it would not be possible for the narrative to always continue, and the system would fail. If the plot fragments were reused, the game would become repetitive for the player. It would also diminish the level of control the player felt they had over their actions in the story, since the player would be forced to repeat fragments as they were played incorrectly the first time through. By prioritizing the fragments, it becomes impossible for the user to fail and the game to end due to an incomplete mission.

The author is also responsible for making sure every mutable and trigger precondition is attainable by the system. This ensures the system is able to change any non-fixed precondition into a goal and present some way for the player to achieve this goal. If this is not possible for a particular precondition, then it must be added as a fixed precondition by the author of the fragment. This guarantees that no unreachable goals enter the Goals list, thus ensuring the narrative is always able to continue.

However, these situations may not always be evident when designing a narrative for a videogame. Therefore the author would need some assistance in determining which goals are unattainable and which preconditions may require more plot fragments to achieve. For these reasons, the prototype system can be viewed as a testing and authoring tool to provide feedback to the author during the designing stages of the narrative. Using the implementation as a tool can help the author in designing, simulating, and testing every aspect of the narrative, without the need to implement the narrative fully within a videogame environment.

Chapter 5

5 Case Studies

The following sections provide sample case studies of the prototype game system developed in Chapter 4. A reference of all plot fragments used in the case studies is available in Appendix B. For this case study, the prototype system simulates a cops and mobsters game. The characters of the game can roam freely throughout the city in which they live. A layout of the city is shown in Figure 14.

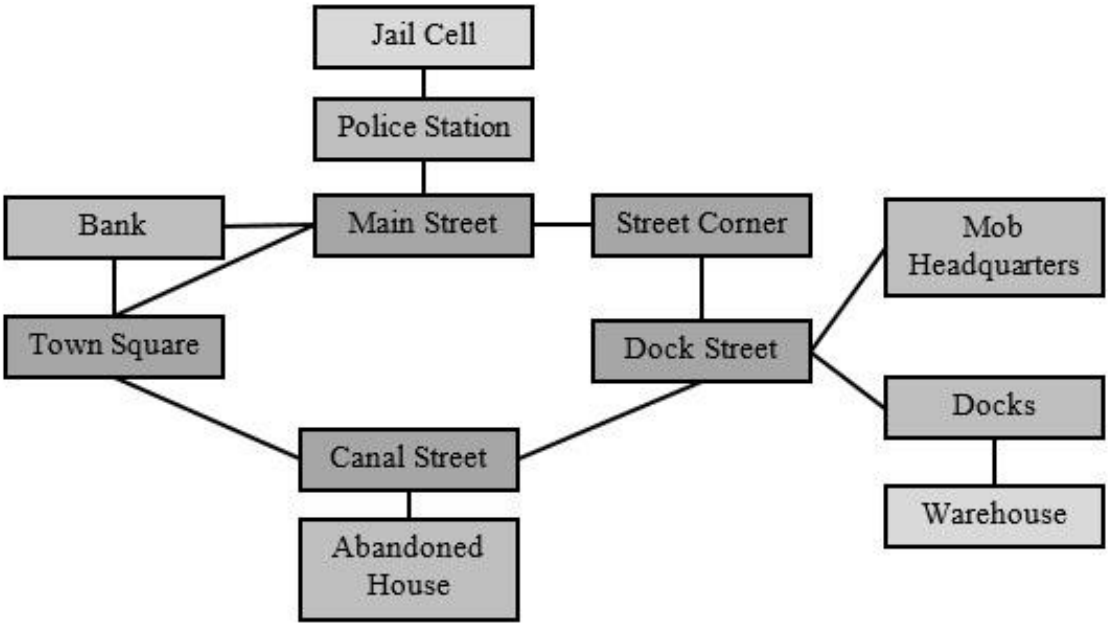


Figure 14: Layout of the city

Upon initialization of the system, the main character is created with the following attributes:

Attribute	**Value**
Name	Nick
Gender	Male
Story Role	Police Officer
Items held	Gun
Relationships to other characters	Good Friends with Paul
Main character	True

The main player character, Nick, is a police officer. Other characters, including other police offices and mobsters, are also created in the system.

Initialization of the Game World state is as follows: the police are aware of an illegal drug shipment being made to the city, but must determine where the drop off will take place. The plot fragment labeled ‘crashDrugShipment’ is the author-level fragment that the Director is attempting to achieve. This plot fragment is shown in Figure 15. When the system is initialized, two preconditions must still be met before the fragment can be presented to the user, and the Director adds these to the Goals list at the start of the run.

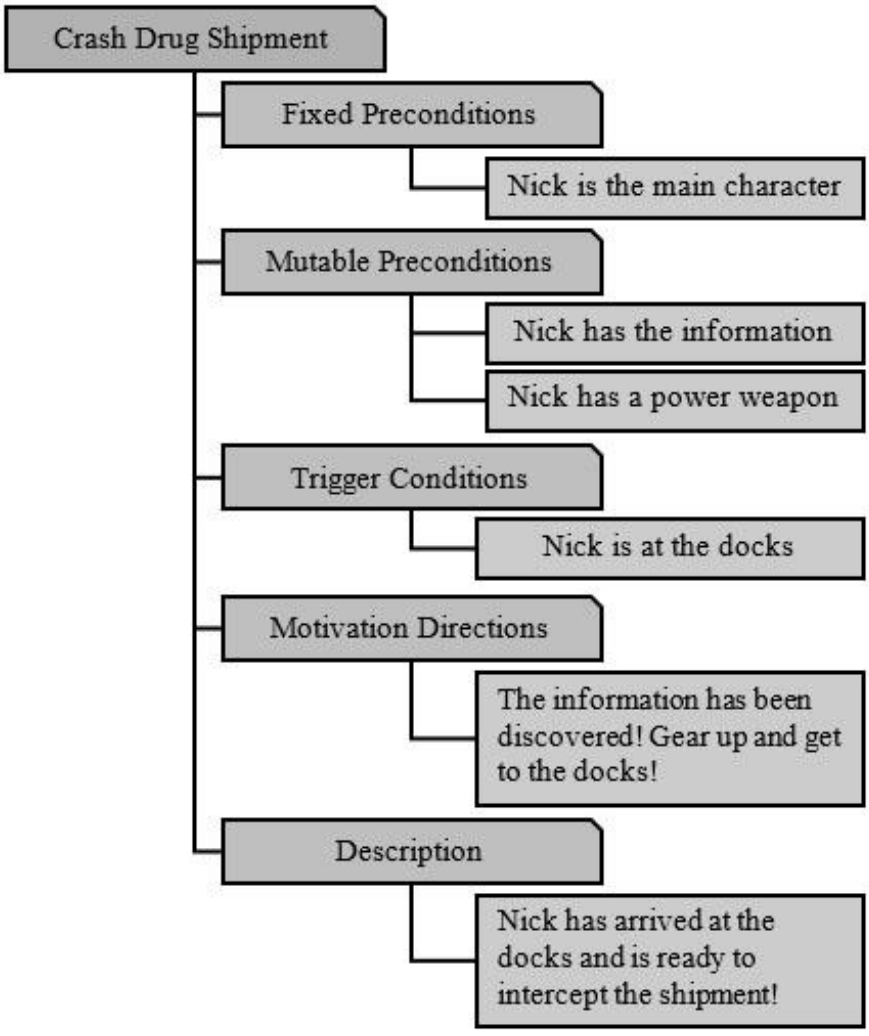


Figure 15: Crash Drug Shipment Plot Fragment

To meet the goals in the Goals list, the Director finds suitable plot fragments and adds them to the Active Plot Fragment list.

Any missing preconditions for these new plot fragments are also added to the Goals list. The process repeats, until eventually every goal on the Goals list has a plot fragment in the Active or Trigger list that will attempt to achieve it. At this point, control of the system is given back to the user.

Trigger conditions are not added to the Goals list by the Director. Instead, the user is encouraged by the system to achieve the trigger conditions using plot fragment Motivation. For this prototype, Motivation is conveyed through text-based instructions presented as 'Hints' to the user. When all of an active plot fragment's preconditions are met and it is moved to the Trigger Plot Fragment list, the hint text is printed on the output tab. This is a simple method of conveying the necessary actions the user should take advantage of in order to advance the narrative. The user can also view the triggers that must still be achieved by examining the plot fragment information (Figure 13 (E)).

In the following sections, outputs for two different scenarios of a user playing the game are shown. In the first, the user attempts to play the game correctly by achieving the necessary goals for each presented plot fragment and reaching the author-level plot fragment 'crashDrugShipment'. The second output demonstrates the other extreme, where the player misses every intended plot fragment goal and eventually forces the Director to select the lowest-level plot fragments to continue the narrative.

Plot fragments chosen to fulfill the system goals are not always chosen the same way each time the system is run. This is evident by analysing the initial setup of Case Study 1 in Section 5.1 and Case Study 2 in Section 5.2. The first Case Study begins with a plot fragment hint describing a suspicious driver somewhere around Main Street and Dock Street, while the second Case Study begins with two hints, one directing you to check out the mob headquarters, and one directing you to go to the bank. While it may appear that these Case Studies begin differently, they actually have the same setup, and are both trying to fulfill the preconditions for the crashDrugShipment fragment described in Figure 15.

In the case study outputs shown in the next sections, text styles are used to represent the different types of output. Hints begin with the word *Hint* and are presented as *italicized text*. A plot fragment begins with a line beginning with ‘Currently playing the plot’ followed by the name of the plot fragment. This is followed by a short description which starts with the word *Description:* and is shown as *italicized text*. This is followed by any Game World changes made by the user, shown in **bold**.

For simplicity, sections during which the user moves a character between multiple locations consecutively were omitted and replaced by ‘**…**’. The omitted lines simply state the locations that the characters visited on their journey through the city between their starting location and destination.

Sections of highlighted text are comments added to clarify what is happening in the narrative. These were added after the case studies were run to help make it clearer what steps were taken in the examples.

5.1 Case Study 1 – Player Plays Properly

The following output of the system reflects an output achieved when a user plays the narrative as intended. The user achieves each intended goal of the presented plot fragments. Note that during the plot ‘ConfiscateFromWarehouse’ the user had the opportunity to arrest Carlo but does not. Because Carlo is not arrested during the chase, the Director is forced to present the user with another opportunity to put Carlo in jail. The Director presented the fragment ‘FailedBankRobbery’ and the user is able to successfully arrest Carlo at that time.

5.1.1 Output 1

Once the system is initialized and the plot fragments and Game World elements are loaded, the user selects the ‘crashDrugShipment’ fragment as the author-level fragment. The Director then takes control and finds plot fragments to fulfill the preconditions of this author-level fragment. In this simulation, the director chooses the ‘confiscateFromWarehouse’ plot fragment to achieve one of the preconditions, and presents a hint to the player to help trigger the plot fragment.

Plot fragment ‘crashDrugShipment’ added as a goal story.
Hint: You get a call that someone is driving erratically. Better check out Main Street and Dock Street.

With the hint output to the screen, the user knows they need to go to either Main Street or Dock Street to trigger a plot fragment. Using the Game World tab (Figure 10) the user can move the Character Nick to Main Street.

Nick is now at the location mainstreet.

This triggers the plot fragment ‘ConfiscateFromWarehouse’ which then displays its description to the user.

Currently playing the plot 'ConfiscateFromWarehouse'.
Description: Nick is patrolling the streets and catches Carlo driving erratically. A car chase begins, ending at the warehouse. A foot chase begins to catch the fleeing mobster!

The user simulates the chase that would take place, ending with both characters at the warehouse. This fragment can end with the Character Carlo in prison as well as Nick finding the Power Weapon. In this case, the user chose to have Nick fail the objective of arresting Carlo, but find the Power Weapon.

…
Carlo is now at the location warehouse.
Nick is now at the location warehouse.
Nick now has the powerweapon.
Completing the plot 'ConfiscateFromWarehouse'.

A new plot fragment is ready to be triggered, so its Hint is displayed to the user.

Hint: You need to do some banking. Better head over to the bank.

The user moves Nick to the Bank.

…
Nick is now at the location bank.

The plot fragment ‘FailedBankRobbery’ is triggered. This fragment involves Carlo again, and because he isn’t required anywhere else the system can move him to the Bank automatically.

Currently playing the plot 'FailedBankRobbery'.
Description: Nick the police officer is witness to a bank robbery, but the criminal, Carlo, doesn't know there was a police officer taken hostage!

The user chooses to have Nick succeed in arresting Carlo, and moves him to the Jail Cell and Nick to the Police Station.

…
Carlo is now at the location jailcell.
Nick is now at the location policestation.
Completing the plot 'FailedBankRobbery'.

Another plot fragment is ready to be triggered so a motivational hint is displayed to the user. This time Nick is already in the trigger location so the fragment begins immediately.

Hint: Carlo is at the police station in custody. You should get there to interrogate for information.

Currently playing the plot 'GetInfoInterrogation'.
Description: Carlo is a mobster and is being held at the Police Station. Nick needs information from Carlo and will get to interrogate him for that info.

Nick was successful in his interrogation and Carlo gave up the needed information.

Nick now has the information.
Completing the plot 'GetInfoInterrogation'.

The plot fragment ‘crashDrugShipment’ requires the preconditions that Nick have the Power Weapon and the Information. Since both are true the author-level fragment is ready to be triggered.

Hint: The information has been discovered! Gear up and get to the docks!

The user moves Nick to the Docks to trigger the author-level fragment. The drug bust was successful so Nick acquired the Drugs

…
Nick is now at the location docks.

Currently playing the plot 'crashDrugShipment'.
Description: Nick has arrived at the docks and is ready to intercept the drug shipment!

Nick now has the drugs.
Completing the plot ‘crashDrugShipment'.

5.2 Case Study 2 – User Fails to Achieve Goals

In this case study, the user is presented with a number of opportunities to achieve the system goals, but fails to achieve them each time. The plot fragments 'NewShipment' and ‘AnonTipper’ are low-level fragments that the Director can use as a last resort to achieve the goals and continue the narrative.

5.2.1 Output 2

Again, the system is initialized and the user selects the ‘crashDrugShipment’ fragment as the author-level plot fragment. However, this time the Director chooses to use the ‘FailedBankRobbery’ and the ‘GetInfoMobHeadquarters’ plot fragments to achieve the system goals.

> Plot Fragment 'crashDrugShipment' added as a goal plot.
> *Hint: Maybe there will be information at the mob headquarters. Better wait until it's empty.*
> *Hint: You need to do some banking. Better head over to the bank.*

The user had a choice this time, and moves Nick to the Bank. The fragment is triggered, however Nick fails to arrest Carlo before the plot fragment ends.

> **Nick is now at the location bank.**
>
> Currently playing the plot 'FailedBankRobbery'.
> *Description: Nick the police officer is witness to a bank robbery, but the criminal, Carlo, doesn't know there was a police officer taken hostage!*
> Completing the plot 'FailedBankRobbery'.

Since Nick failed to arrest Carlo at the Bank robbery, the Director is forced to present a new opportunity to arrest Carlo. Arresting Carlo is not a precondition for the author-level plot fragment, however the Director has chosen a plot fragment to complete one of the author-level plot preconditions which requires Carlo to be in jail. Therefore the Director will try again to have Nick arrest him, this time using the ‘ConfiscateFromWarehouse’ plot fragment.

> *Hint: You get a call that someone is driving erratically. Better check out mainstreet and dockstreet.*
> **Nick is now at the location mainstreet.**

The user chooses to investigate the erratic driver, triggering the plot fragment ‘ConfiscateFromWarehouse.’ Once again, Nick fails to arrest Carlo.

> Currently playing the plot 'ConfiscateFromWarehouse'.

Description: Nick is patrolling the streets and catches Carlo driving erratically. A car chase begins, ending at the warehouse. A foot chase begins to catch the fleeing mobster!
…
Carlo is now at the location warehouse.
Nick is now at the location warehouse.
Completing the plot 'ConfiscateFromWarehouse'.

The user still has the option to pursue the first hint presented when the system was initiated. Therefore Nick goes to the Mob Headquarters and triggers the 'GetInfoMobHeadquarters' plot fragment

…
Nick is now at the location mobheadquartersyard.

Currently playing the plot 'GetInfoMobHeadquarters'.
Description: Nick is at the mob bar headquarters and it is empty at the moment. Nick is going to break in to look for information.
Completing the plot 'GetInfoMobHeadquarters'.

The author-level preconditions are still unsatisfied, so the Director chooses new plot fragments to present to the user.

Hint: You get a tip that some criminals are at the street corner. Maybe go check it out.
Hint: A shipment has come into the station. Head over there to receive some new gear.

Nick is moved to the Police Station by the user.

…
Nick is now at the location policestation.

With a number of failed attempts, the Director chooses a low level fragment to present the user with. 'NewShipment' is a fragment used as a last resort to guarantee Nick receives a Power Weapon, which is required for the author-level plot fragment.

Currently playing the plot 'NewShipment'.
Description: The police station just received a new order of weapons. Nick has received a new powerweapon.
Nick now has the powerweapon.
Completing the plot 'NewShipment'.

The user moves Nick to the Street Corner to investigate another hint. The plot fragment 'ArrestDrugDealer' is played and Nick fails to arrest Carlo again.

…
Nick is now at the location streetcorner.

Currently playing the plot 'ArrestDrugDealer'.
Description: Nick catches Carlo selling drugs on the street corner. Nick chases Carlo to arrest him and take him to the Police Station.
Completing the plot 'ArrestDrugDealer'.

Another hint is presented to the user to get to the Police Station.

Hint: You get a call that there is good news at the police station. Get there right away!
…
Nick is now at the location policestation.

With too many failures, the Director chooses a low level plot fragment to have Carlo arrested.

Currently playing the plot 'MobsterArrested'.
Description: Nick arrives to the police station to find Paul has arrested Carlo.
…
Carlo is now at the location jailcell.
Completing the plot 'MobsterArrested'.

Hint: Carlo is at the police station in custody. You should get there to interrogate for information.

With Carlo in custody, the Director selects the ‘GetInfoInterrogation’ plot fragment to present to the user in an attempt to get Nick the Information needed for the author-level plot fragment. Nick fails to get the information.

Currently playing the plot 'GetInfoInterrogation'.
Description: Carlo is a mobster and is being held at the Police Station. Nick needs information from Carlo and will get to interrogate him for that info.
Completing the plot 'GetInfoInterrogation'.

Again, the Director chooses a low-level plot fragment to present to the user to guarantee that Nick gets the Information.

Hint: You get a call that there is good news at the police station. Get there right away!

Currently playing the plot 'AnonTipper'.
Description: Nick gets an anonymous call, supplying needed information.
Nick now has the information.
Completing the plot 'AnonTipper'.

Hint: The information has been discovered! Gear up and get to the docks!
…
Nick is now at the location docks.

Currently playing the plot 'crashDrugShipment'.
Description: Nick has arrived at the docks and is ready to intercept the drug shipment!
Nick now has the drugs.
Completing the plot 'crashDrugShipment’.

Chapter 6

6 Discussion of the Architecture

6.1 Architecture Development

In this work, an architecture for creating an interactive emergent narrative was developed. The architecture was designed specifically to remain separate from the Game World to ensure compatibility with any pre-existing game mechanics. This is achieved by observing the Game World values in order to plan the narrative. In the architecture, the Evaluator observes the necessary Game World values, chooses plot fragments or missions with the Director, and presents plot fragments to the player via the Controller.

The Evaluator provides this interface between the Game World and Director, and allows the Director to check the current status of any story world value. It can be customized to interface with any Game World, and is capable of checking any preconditions specific to the mechanics of a particular game. The Director of the architecture organizes the system's goals and preconditions, and determines which plot fragments must be presented to achieve these goals. The Director is a core component of the architecture and does not need to be modified to accommodate a new Game World. It presents plot fragments to the player through the system's Controller. The Controller, like the Evaluator, interacts directly with the Game World, and can therefore be customized to perform manipulations for each unique Game World. The player is permitted to perform any actions they desire within the Game World mechanics, whether they serve to achieve the system's goals or not, and the Director adapts the story to these actions. If the player is unable to achieve a particular goal, the Director continues to present new plot fragments and provide the player with additional opportunities to meet the goal. Eventually, the Director presents the user with a low-level plot fragment to ensure the goal is achieved and the story always progresses. The architecture successfully generates continuous dynamic and consistent emergent narratives that offer variable story control, a significant level of freedom to the user, a high degree of creative control to the author, and is domain independent.

6.2 Evaluation of Architectural Components

6.2.1 Temporal and Spatial Consistency

The temporal consistency of the proposed architecture is maintained through the composition of the plot fragments. Preconditions guarantee that certain conditions must be true before the fragment is presented to the user. Author-level plot fragments can be ordered through the use of preconditions, guaranteeing the correct presentation of plot points, and in turn, the correct order of plot lines. As shown in Figure 16, plot fragments need to have their preconditions satisfied before the order can be acceptable. For example, a plot fragment where two character go on a mission together must happen after a plot fragment where the two characters meet.

Spatial consistency is guaranteed by the Game World and Controller systems, as well as through the maintenance of character and item locations in the Game World. As agents in the world control themselves, or rather are directed by the Controller, they maintain proper spatial locations within the Game World. Although both temporal and spatial consistencies are possible in the proposed emergent architecture, it is ultimately the responsibility of the author to maintain the consistency of the narrative within the plot fragments written. It is possible that the author may add preconditions to plot fragments that would permit inconsistencies in the story, and it is important for these situations to be avoided. This is how the prototype system can assist in the creation and testing of the narrative. The author is able to simulate different situations and test for inconsistencies in the plot.

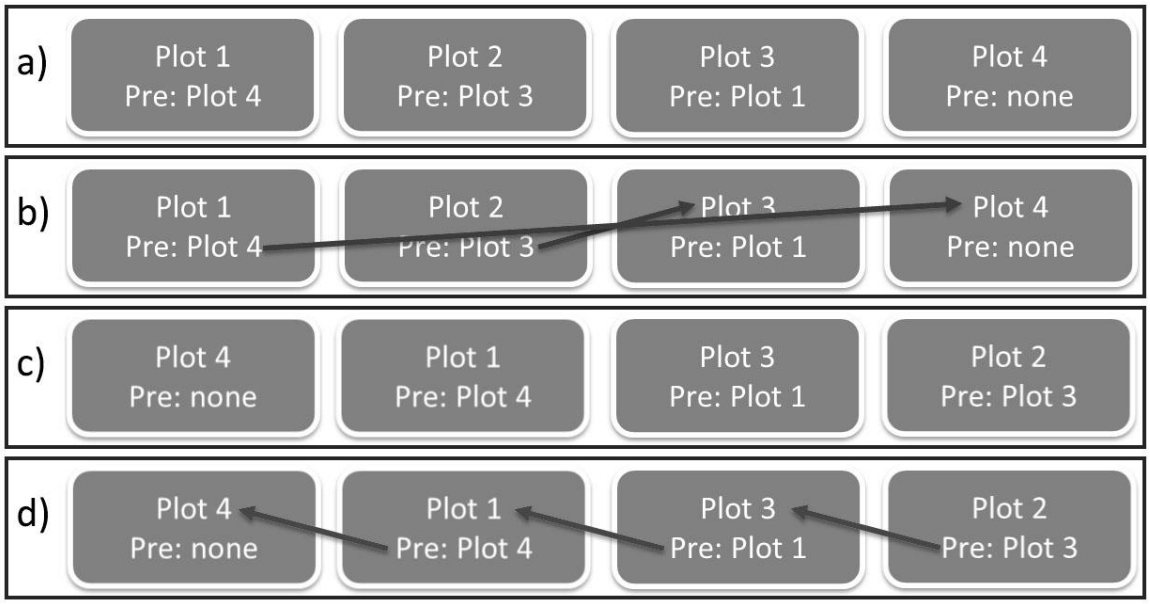


Figure 16: Preconditions in Plot Fragments a) In the initial arrangement of Plot Fragments the preconditions are not satisfied as shown in b). Rearranging the Fragments to the arrangement shown in c) allows for all preconditions to be satisfied, as seen in d).

6.2.2 Granularity of Story Control

The proposed system has medium to high granularity, depending on the author's intentions. The level of the author-level fragments (high to low) determines the level of control the Director has in choosing the plot fragments. If the author creates primarily high-level author plot fragments, representing major plot points, the Director has high-grained control over the fragments presented to the user. If the author-level fragments are largely low-level, which are easily achievable with few preconditions, the Director has limited choice in which fragments to present before the next author-level fragment is reached. With higher-grained control, the Director requires more time to organize and present the plot fragments to the user in between author-level fragments. On the contrary, lower grained control allows author-level fragments to be reached more often, keeping the narrative on a focused path. The narrative system developed supports varying degrees of story control, affording the author the opportunity to determine the level of granularity though the plot fragments they create.

6.2.3 Freedom to the User

While the user is given enough freedom to affect the narrative on a small scale, the overall narrative has a deterministic path. The narrative's major plot points are predetermined by the author (through the author-level fragments), and the system's Director will work to fulfill the preconditions necessary to execute these fragments. However, within any plot fragments, the player can fulfill or fail to achieve any intended goals, and the narrative can indeed recover and continue. This is a result of the system's design to pursue goals persistently. The Director repeatedly presents new plot fragments if the player continuously fails to achieve a goal. Only after the goal has been fulfilled is it removed from the Goals list. Overall, the player is given the freedom to directly affect the plot presented in between major plot points, but the overall narrative is maintained throughout play.

However, as discussed in Section 3.1.8 the author can incorporate some forms of influence on the overall narrative with the use of branching plot fragments. The user can make a choice when reaching a branch, and the narrative can take a number of different directions based on how the Game World values effect the different author-level fragment pre-conditions.

6.2.4 Ease of Authoring the Story/Creative Control

An author using the system has significant control over the creative direction the narrative will take. The author determines the level of the author-level plot fragments (from high to low), and allows the Director to fill the plot in between these major plot points. Narratives created by this system can range from highly controlled linear stories, in which many major plot points are predetermined, to very dynamic stories, in which only a few major plot points are chosen and the Director must decide how to fulfill them. The author can also decide on the level of control between the major plot points by varying the level of the plot fragments (*i.e.* creating predominantly higher-level or lower-level plot fragments). For example, the author may wish to exert greater control at the beginning of the narrative by using many lower-level plot fragments. The author can then

add higher-level fragments as the story progresses to enhance the degree of variability in the latter half of the story.

Authoring stories can become increasingly difficult as the degree of freedom given to the user increases. As more plot fragments and preconditions are added to the system, different plot fragments with different end goals must be added to ensure each goal is attainable. It is also best to include many different fragments to achieve a particular story goal, with varying degrees of difficulty, in order to enhance the flexibility of gameplay. Although this increases the complexity of system development and testing, the same fragment can potentially be used to satisfy different, anticipated goals. For example, a fragment that can be used to acquire a gun may be used in multiple time points in the narrative.

However, the current prototype can be used as an aid to authors to test their narratives and determine where they may need to add more plots fragments of varying types. If an author is testing their narrative and finds the system is unable to achieve a certain goal easily, a clear indication can be given of which goal is difficult to attain and why. This can guide the author by providing feedback for which plot fragments may need to be added.

The current prototype game system contains a limited pool of plot fragments as it is only a preliminary system created for demonstrative purposes. The plot pool could be expanded to enhance the dynamic nature of the game. Although this thesis does not provide a means to expand on the current pool of plot points dynamically, manually adding new plot points will not disrupt the game play as the architecture was designed to allow for the addition of new plot fragments easily.

6.2.5 Pacing Control

Although no explicit pacing controls were implemented in the current architecture, some pacing is implicit through the way the plot fragments are created. By using precondition with minimum and maximum time constraints, the author can control the pace of the narrative to prevent the player from completing plot fragments too quickly or too slowly.

The author can also use shorter and more frequent plot fragments with easily fulfilled preconditions, or longer and less frequent plot fragments, with difficult preconditions. Although this can potentially influence the pace of the narrative, it does not guarantee pacing control. A user can fail to achieve a system goal and prolong the presentation of an author-level fragment.

However, it is difficult to enforce pacing in a gaming system without taking away from a player's freedom. Without monitoring and strictly controlling the duration of time a player spends trying to achieve a goal, the Controller can do little to predict and influence how long a player will spend exploring a level before completing an objective.

6.2.6 Domain Dependency

The narratives generated by the architecture are not restricted in the types of domains they can support. Linear storylines are achievable by setting strict preconditions and maintaining a high level of control over the plot points presented. Highly flexible narratives can be created by allowing variability in the preconditions and leaving many decisions up to the Director.

Additionally, the story fragments are not restricted to any single domain type, so the system can support any narrative type the author wishes to create. The modifications required to change the story domain are minimal, and the underlying architecture would remain unchanged. Changes in the prototype system would require changing the preloaded text files to implement a story with a different theme, as the prototype is data driven and the architecture does not depend on any particular domain.

When implementing the architecture into a videogame the only systems that would require modification to match the game engine would be the Evaluator and Controller systems in order to interact with the new Game World. The Director operations would remain the same, as would the plot fragment selection and narrative generation.

6.3 Comparison to Existing Narrative Generation Systems

Components of the architecture developed draw from aspects of existing narrative generation systems. The Director operations of our narrative generation architecture are deeply rooted in the Universe system. The Universe system uses a Director that chooses narrative goals to be achieved, and selects the best plot fragment for achieving these goals. Plot fragments in our system are selected in a similar manner. However, the two systems differ in goal selection method. In Universe, the goals are predefined narrative goals that represent different plot types, such as the goal to churn the relationship of two lovers. The goals in our architecture are created by the Director using unmet preconditions of active plot fragments. This difference allows the goals in our system to be created dynamically and to be met using a variety of different plot fragments existing in the plot pool. The goals in Universe were meant to be chosen somewhat randomly and to be achieved simply to continue the narrative. The goals in our system are meant to continue the narrative as well, but are chosen based on the larger goal of achieving author-level plot fragments.

The Oz system uses major plot points to guide the overall direction of the narrative. Similarly, our architecture uses author-level plot fragments selected by the author to maintain the narrative's direction. Between the major plot points of the Oz system the player is able to manipulate the Game World and play the game freely. While the player does not actively choose the fragments presented to them in our framework, the Director can create a dynamic narrative based on the user's actions. Furthermore, the plot between the author-level fragments varies each time the system is used.

6.4 Architecture Flexibility

The architecture developed in this work was primarily intended for use in an open world style of videogame. However, it is also capable of accommodating linear narratives and can be used to apply emergent characteristics to a linear story. In this case, the plot fragments would represent different levels presented to the player. The player is free to choose how each level is played, so long as the required objectives are accomplished. For

example, the dynamic game play in the Hitman series of games [46] allows a player to complete each level in whichever way they choose. The player may go into a level silently and never be detected, or go in guns blazing and try to eliminate everyone. The story values simply change based on the way the game is being played. The architecture developed in this thesis permits different levels to be presented depending on how the game is being played and story world values that result. For example, if a player finds a sniper rifle in one level, the next level presented could be sniper-based. The sniper-based level would simply have a precondition that the player be in possession of the rifle.

The architecture developed in this work also supports the addition of new plotlines without disturbing the existing narrative. This is accomplished through the design and selection method of plot fragments. A fragment is only ever presented to a user if all preconditions are met. Because the preconditions are well-defined and rely on Game World values, consistency of the narrative is guaranteed. Thus, if a new plot fragment is added to the system, it may be presented to the player at any time the Director would like, so long as the preconditions are met. Consequently, a game employing the architecture can be augmented post- release, without requiring the player to restart the game in order to access the new plot fragments.

6.5 Other Considerations

The two system outputs detailed in Sections 5.1 and 5.2 demonstrate varying degrees of player cooperation with system goals and intended play through presented plot fragments. These outputs highlight the role of the Director to fill story action (by using different plot fragments) when the user fails to achieve the goals of the fragment. Although the system can handle instances where the user fails to achieve intended goals, there are some limitations in this preliminary system. These include situations of forced gameplay, the degree to which it realistically represents a videogame, and the length of the story presented.

6.5.1 Forced Gameplay

In the developed architecture, the Director continuously presents the user with new plot fragments to meet a goal until that goal is met. If the player cannot meet their objective,

the Director must eventually present the player with a fragment that ensures the goal is met. This is necessary in order to avoid exhausting the fragment pool. However, this can appear as forced gameplay, since the player goal is eventually met without requiring them to actually do anything. For example, in output 2 (Section 5.2), the user fails to arrest Carlo a number of times. Eventually, the system chooses to have another police office arrest him, and allows the narrative to continue. To avoid any forced gameplay, a more complex videogame would have these failures affect the player's statistics, and could even impose repercussions on their standing in the police force. Furthermore, the player is given ample opportunities to reach their goal, and requiring the Director to essentially accomplish it is necessary to continue the narrative. If the player is given the freedom to fail an objective, a checks and balance system must be implemented so that the player cannot fail indefinitely. Otherwise, the game would eventually become unplayable and ultimately fail.

6.5.2 Limited in Representing the Full Dimensions of a Video Game

The prototype system developed for this work was intended to demonstrate the capabilities of the emergent narrative architecture developed in this thesis. Although a fully constructed game was outside of the scope of this project, it is discussed in the future work section of Chapter 7 as a possible next step in developing the architecture. Indeed, the demonstration is much simpler and less dynamic than the typical videogames developed today, and therefore does not simulate a complete gaming experience.

For example, just by looking at the outputs the objectives of each plot fragment are not necessarily clear, while in a fully developed videogame the goals of the level or mission would be made clearer to the user. This could be done through the use of short cut-scenes to introduce the goal, or using an interaction with another character to indicate the intended goal. In the prototype system, the fragments provide the user with a short description of the intended goals of the plot. The user is only aware of the intended goals of each plot fragment by analyzing the plot fragments presented on the interface (as shown in Figure 13(e)). However, the overall aim of the developed system was to present an emergent narrative that maintains consistency during play. The system is sufficient in

demonstrating how a consistent and continuous plot can be creatively maintained, yet still influenced by a user's actions. Regardless of how much information is provided to the user with respect to the plot's intended direction, the system is capable of handling any player action, even if it does not match the intended plot direction. As demonstrated by Figure 10, and the example outputs in Sections 5.1 and 5.2, the user has control over the world values necessary to make changes in the Game World, and the Director and Evaluator can successfully use the outputs of each plot fragment to move the narrative forward.

6.5.3 Limited in Length

To illustrate the capabilities of the developed system, example outputs from a single author-level plot fragment are presented. Yet, the ultimate goal of the proposed narrative generation system is to handle a much longer narrative, with several author-level plot fragments. This can be accomplished with the current architecture by using the completion of one plot fragment as the precondition for other plot fragments. In this way, the narrative can be expanded beyond just a single author-level fragment to include several major plot points (whose order is decided upon by the author). The outputs of each individual author-level plot fragment can simply be linked together in a chain of author-level fragments, each affecting each other and the necessary goals for the next section of the narrative.

Chapter 7

7 Conclusion and Future Work

7.1 Contributions

This thesis explored the development of an emergent narrative generation system for videogame development. Major contributions of this work are as follows:

1. Designed an architecture for interactive emergent narratives that:
    i) Offers variable granularity, user freedom to affect the story, creative control to the author, temporal and spatial consistency, and is domain independent.
    ii) Does not limit certain outcomes during a plot fragment, allowing the user to complete the plot points as they wish. The Director is able to recover from mistakes and will present more options to complete desired goals.
    iii) Works separate from the Game World, allowing the system to operate without any specific Game World mechanics, and customizable to accommodate any pre-existing game mechanic or statistic.
    iv) Permits the addition of plotlines (either author-level or other plot fragments) without the disruption to the existing narrative.

2. Developed an implementation that can be used as a tool for authors to test and debug plot fragments and narratives. Authors can test the story, for a video game using the architecture, separate from the development team. This allows for testing of story elements without the need for integration with the development team until the story has been fully tested.

7.2 Future Work

The architecture presented in this thesis is a solid foundation for a complete narrative generation system to be used in video games and other interactive narratives forms. Although the current system simulates a simple Game World and can effectively choose

plot fragments to present a continuous, emergent narrative to the user, further work would extend its capabilities. Future work should include:

1. Integrating the architecture into a more sophisticated and complex video game. A complete video game which uses a sophisticated game engine's graphics and gameplay would demonstrate the narrative architecture without requiring changes to the architecture's underlying framework. This would also mean developing many more plot fragments to create a more in depth narrative for the player. This would allow for proper demonstration of the architectures design, as well as allowing the Director to work to its full capacity with many more plot fragments.

2. Having the ability to modify the existing plot points to extend the current pre-authored plot pool. This could be done by modifying small details within the plot points to change the outcomes. This would allow for a greater number of plot fragments to be used by players without requiring the author to create additional points. As with Minstrel and Universe, this could lead to the possibility of farfetched plot fragments. Therefore the modified plots should also include an evaluation to examine their coherency.

3. Redesigning a portion of the architecture accounting for history in the Game World. Currently, the system was designed as a cursory investigation into narrative generation, and uses the Game World 'as is.' Prior knowledge of the world does not exist. Implementing a mechanism to maintain the history of Game World value changes would broaden the capabilities by allowing the addition of new features. Some of these new features could include:

    a. Literary devices such as flashbacks. The current architecture does not support using a plot fragment as a flashback. A flashback would allow a character to remember a past event, which can be used to uncover details or information about the Game World. However, the current framework has no knowledge of previous Game World states, and the narrative's spatial or temporal consistency would not necessarily be maintained with the use of flashbacks. In order to maintain spatial and temporal consistency with the use of flashbacks, checks would need to be in place to ensure inconsistencies don't occur.

b. Late commitments, similar to the Virtual Storyteller system. This would allow for a certain Game World value to be assumed as always having been true without significantly affecting the narrative. Prior knowledge of the Game World is necessary for this since changing a Game World value requires checking with game agents to maintain consistency within the narrative. Late commitments can also be introduced through the use of flashbacks.

c. Narrative pacing. This requires knowledge of the length of time the Game World has been in a certain state. Without knowing previous Game World states, the system is unable to determine which dramatic changes the narrative has recently taken, and is therefore unable to determine when to present tense or relaxing moments to the player.

4. Include user testing to test the effectiveness of the architecture. This would include two different kinds of user testing.

a. The first type of user testing would be to evaluate the narratives that are produced. This would be done with regular users with no prior knowledge of the narrative elements or architecture design, and using a set of criteria they would critique the narrative produced. This would help determine if the narratives make sense, are coherent, and lastly if they are interesting.

b. The second type of user testing would be to evaluate how useful the system is to an author wishing to create a story with the architecture. The author would be instructed on how the system works and how to create plot fragments, and would be given the tools to construct a narrative that they wish. The author using the system would then critique the components of the architecture and the tools, giving feedback on what would make the creation process easier and more effective.

References or Bibliography

[1] J. Lebowitz and C. Klug, Interactive Storytelling for Video Games-A Player-Centered Approach for Creating Memorable Characters and Stories. Focal Press, 2011.

[2] T. Sylvester, Designing Games: A Guide to Engineering Experiences. O'Reilly Media, Incorporated, 2013.

[3] C. M. Bateman, Game Writing: Narrative Skills for Videogames. Charles River Media, 2007.

[4] P. Holleman, "Narrative in Games," The Game Design Forum. .

[5] M. Lebowitz, "Story-Telling as Planning and Learning.," 1985.

[6] S. Turner, The Creative Process: A Computer Model of Storytelling and Creativity. Psychology Press, 1994.

[7] M. Mateas, "An Oz-Centric Review of Interactive Drama and Believable Agents," 1997.

[8] M. Mateas and A. Stern, "Façade: An Experiment in Building a Fully-Realized Interactive Drama," in Game Developers Conference (GDC'03), 2003.

[9] B. Magerko, "A proposal for an interactive drama architecture," Ann Arbor, vol. 1001, p. 48109, 2000.

[10] B. Magerko and J. Laird, "Building an interactive drama architecture," First Int. Conf. Technol. Interact. Digit. Storytell. Entertain., pp. 226–237, 2003.

[11] M. Theune, S. Faas, E. Faas, A. Nijholt, and D. Heylen, "The Virtual Storyteller: Story Creation by Intelligent Agents," in Proceedings of the Technologies for Interactive Digital Storytelling and Entertainment (TIDSE) Conference, 2003, pp. 204–215.

[12] R. McKee, Story: Style, Structure, Substance, and the Principles of Screenwriting. HarperCollins, 2010.

[13] B. Cotter, "Story, Plot and Narrative (Not the Same Thing)," Booniverse, 15-Sep-2012. .

[14] M. Twain, "Fenimore Cooper's Literary Offenses," Univ. Va., 1895.

[15] R. Jarrell, Randall Jarrell's Book of Stories. NYRB Classics, 2002.

[16] D. Boyle, Slumdog Millionaire. 2008.

[17] “BNL | History:The First Video Game?,” Brookhaven National Laboratory. [Online]. Available: http://www.bnl.gov/about/history/firstvideo.php. [Accessed: 27-Mar-2014].

[18] Red Dead Redemption. Rockstar Games, 2010.

[19] D. Houghton, “Video Game Storytelling: The Real Problems and the Real Solutions,” GamesRadar, 08-Nov-2013. .

[20] W. Spector, “Warren Spector’s Commandments of Game Design,” Games Industry, Sep-2013. .

[21] J. Kuschke, “Aristotle was not a Game Designer,” Gamasutra, Sep-2013. .

[22] The Walking Dead: The Game. Telltale Games, 2012.

[23] D. Freeman, Creating Emotion in Games: The Craft and Art of Emotioneering. New Riders, 2004.

[24] “Sony Entertainment Network Store,” Sony Entertainment Network. [Online]. Available: https://store.sonyentertainmentnetwork.com. [Accessed: 20-May-2014].

[25] “Xbox Games Store,” XBox Games Store. [Online]. Available: http://marketplace.xbox.com. [Accessed: 20-May-2014].

[26] “Steam Greenlight,” Steam Greenlight. [Online]. Available: http://steamcommunity.com/greenlight. [Accessed: 20-May-2014].

[27] U. Spierling and N. Szilas, Interactive Storytelling: First Joint International Conference on Interactive Digital Storytelling, ICIDS 2008 Erfurt, Germany, November 26-29, 2008, Proceedings. Springer, 2008.

[28] Call of Duty. Activision, 2003.

[29] Grand Theft Auto. Rockstar Games, 1997.

[30] Minecraft. Mojang, 2011.

[31] L.A. Noire. Rockstar Games, 2011.

[32] M. Lebowitz, “Creating Characters in a Story Telling Universe.”

[33] N. Wardrip-Fruin, “Michael Lebowitz on Universe,” Grand Text Auto, 19-Nov-2007. .

[34] R. Pérez y Pérez and M. Sharples, “Three computer-based models of storytelling: BRUTUS, MINSTREL and MEXICA,” Knowl.-Based Syst., vol. 17, no. 1, pp. 15–29, Jan. 2004.

[35] N. Wardrip-Fruin, Expressive Processing: Digital Fictions, Computer Games, and Software Studies. Cambridge, Massachusetts London, England: The MIT Press, 2009.

[36] M. Mateas and A. Stern, “Towards Integrating Plot and Character for Interactive Drama,” in In Working notes of the Social Intelligent Agents: The Human in the Loop Symposium. AAAI Fall Symposium Series. Menlo Park, 2000, pp. 113–118.

[37] V. J. Propp, L. Scott, S. Pirkova-Jakobson, L. A. Wagner, and A. Dundes, Morphology of the Folktale. University of Texas Press, 1977.

[38] I. Swartjes, J. Vromen, and N. Bloom, “Narrative inspiration: using case based problem solving to support emergent story generation,” Proc. 4th Int. Jt. Workshop Comput. Creat., pp. 17–19, 2007.

[39] I. Swartjes, E. Kruizinga, and M. Theune, Let’s Pretend I Had a Sword Late Commitment in Emergent Narrative. .

[40] B. Magerko, “Story Representation and Interactive Drama,” in Artificial Intelligence and Interactive Digital Entertainment Conference, Marina Del Rey, California, USA, 2005, pp. 87–92.

[41] H. Barber and D. Kudenko, “Dynamic Generation of Dilemma-based Interactive Narratives.,” AIIDE, vol. 7, pp. 2–7, 2007.

[42] H. Barber and D. Kudenko, “Generation of Adaptive Dilemma-Based Interactive Narratives,” Comput. Intell. AI Games IEEE Trans. On, vol. 1, no. 4, pp. 309–326, Dec. 2009.

[43] B. Stroustrup, The C++ Programming Language, Third Edition, 3rd ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1997.

[44] H. Nord and E. ChambeEng, Qt Project. Digia plc, 1995.

[45] B. Dawes, D. Abrahams, and D. Frey, Boost C++ Libraries. .

[46] Hitman: Codename 47. IO Interactive, 2000.

Appendices

Appendix A: Glossary

Assets: A contributing file used in video games. This includes art asset files such as audio clips, images, textures, etc. This can also include other needed files, such as binary files containing program code, or text files used for various reasons, such as saving game settings.

Cut scene: A scene used in video games which the player has no control and a part of story is told through a pre-rendered video.

Linear Story: A story which follow a predetermined path in video games. Linear stories offer little to no opportunity for a player to change the direction of the story, and tell the same story each time through.

Non Player Character (NPC): Any character in a video game not controlled by the player.

Open World: Open world describes a type of game world in which the player is free to explore the different areas in any order and duration.

Player Agency: Agency is the level of control that a player feels they have in a game world.

Side Quest/ Side Mission: An extra quest or mission available to a player which is not crucial to the overall story. These can usually be skipped with little to no impact on how the main story results.

Waypoint: A waypoint is a reference point in physical space used for purposes of navigation. In video games waypoints are usually used to indicate to a player a point of interest.

Appendix B: List of Plot Fragments used in the Prototype Implementation organized by major plot objective.

Main Author Level Plot
Name of Plot: crashDrugShipment Plot Level: 2 Fixed Precondition: true isMainCharacter a Fixed Precondition: true personIsMale a Mutable Precondition: true personIsHolding a information Mutable Precondition: true personIsHolding a powerweapon Trigger: true personIsAt a docks Motivational Instruction: The information has been discovered! Gear up and get to the docks! Plot Description: [a] has arrived at the docks and is ready to intercept the drug shipment!
Plots to Get Information
Name of Plot: GetInfoMobHeadquarters Plot Level: 1 Mutable Precondition: true locationIsEmpty mobheadquarters Trigger: true personIsAt a mobheadquartersyard Motivational Instruction: Maybe there will be information at the mob headquarters. Better wait until it's empty. Plot Description: [a] is at the mob bar headquarters and it is empty at the moment. [a] is going to break in to look for information. Possible Change: true personIsHolding a information Possible Change: true personIsHolding a powerweapon Name of Plot: GetInfoInterrogation Plot Level: 1 Fixed Precondition: true personIsStoryRole a Mobster Fixed Precondition: true personIsStoryRole b Police Fixed Precondition: true isMainCharacter b Mutable Precondition: true personIsAt a jailcell Trigger: true personIsAt b policestation Motivational Instruction: [a] is at the police station in custody. You should get there to interrogate for information. Plot Description: [a] is a mobster and is being held at the Police Station. [b] needs information from [a] and will get to interrogate him for that info. Possible Change: true personIsHolding b information Name of Plot: AnonTipper Plot Level: 0 Fixed Precondition: true personIsStoryRole a Police Fixed Precondition: true isMainCharacter a Trigger: true personIsAt a policestation Motivational Instruction: You get a call that there is good news at the police station. Get there right away! Plot Description: [a] gets an anonymous call, supplying needed information. Possible Change: true personIsHolding a information

Name of Plot: WantsToSnitch
Plot Level: 0
Fixed Precondition: true isMainCharacter a
Fixed Precondition: true personIsStoryRole b Mobster
Trigger: true personIsAt a policestation
Motivational Instruction: You get a call that there is good news at the police station. Get there right away!
Plot Description: [a] goes to the police station to find that [b] wants to provide information.
Possible Change: true personIsHolding a information

Plots to get someone in policestation

Name of Plot: FailedBankRobbery
Plot Level: 1
Fixed Precondition: true personIsStoryRole a Police
Fixed Precondition: true isMainCharacter a
Fixed Precondition: true personIsStoryRole b Mobster
Mutable Precondition: true personIsHolding a gun
Trigger: true personIsAt a bank
Motivational Instruction: You need to do some banking. Better head over to the bank.
Plot Description: [a] the police officer is witness to a bank robbery, but the criminal, [b], doesn't know they took a cop hostage!
Possible Change: true personIsAt b jailcell

Name of Plot: ArrestDrugDealer
Plot Level: 1
Fixed Precondition: true personIsStoryRole a Police
Fixed Precondition: true isMainCharacter a
Fixed Precondition: true personIsStoryRole b Mobster
Mutable Precondition: true personIsAt b streetcorner
Trigger: true personIsAt a streetcorner
Motivational Instruction: You get a tip that some criminals are at the street corner. Maybe go check it out.
Plot Description: [a] catches [b] selling drugs on the street corner. [a] chases [b] to arrest him and take him to the Police Station.
Possible Change: true personIsAt b jailcell

Name of Plot: MobsterArrested
Plot Level: 0
Fixed Precondition: true isMainCharacter a
Fixed Precondition: true personIsStoryRole b Police
Fixed Precondition: false isMainCharacter b
Fixed Precondition: true personIsStoryRole c Mobster
Trigger: true personIsAt a policestation
Motivational Instruction: You get a call that there is good news at the police station. Get there right away!
Plot Description: [a] arrives to the police station to find [b] has arrested [c].
Possible Change: true personIsAt c jailcell

Plots to get power weapon

Name of Plot: ConfiscateFromWarehouse
Plot Level: 1
Fixed Precondition: true isMainCharacter a
Fixed Precondition: true personIsAt b streetcorner
Fixed Precondition: true personIsStoryRole b Mobster
Trigger: true personIsAt a mainstreet
Trigger: true personIsAt a dockstreet
Motivational Instruction: You get a call that someone is driving erratically. Better check out mainstreet and dockstreet.
Plot Description: [a] is patrolling the streets and catches [b] driving erratically. A car chase begins, ending at the warehouse. A foot chase begins to catch the fleeing mobster!
Possible Change: true personIsHolding a powerweapon
Possible Change: true personIsAt b jailcell
Possible Change: true personIsAt a warehouse

Name of Plot: BurningDrugHouse
Plot Level: 0
Fixed Precondition: true isMainCharacter a
Mutable Precondition: true personIsAt a abandonedhouse
Trigger: true personIsAt a abandonedhouse
Motivational Instruction: An abandoned house caught fire. Inspect the house.
Plot Description: A burned down house has been found to have been a drug manufacturing facility. The scene must be investigated.
Possible Change: true personIsHolding a powerweapon

Name of Plot: GetToAbandondedHouse
Plot Level: 0
Fixed Precondition: true isMainCharacter a
Trigger: true personIsAt a policestation
Motivational Instruction: You get a call that there is a case at the station. Get to the station to get a briefing.
Plot Description: [a] arrives at the station and is told to head over to a house fire.
Possible Change: true personIsAt a abandonedhouse

Name of Plot: NewShipment
Plot Level: 0
Fixed Precondition: true isMainCharacter a
Trigger: true personIsAt a policestation
Motivational Instruction: A shipment has come into the station. Head over there to receive some new gear.
Plot Description: The police station just received a new order of weapons. [a] has received a new powerweapon.
Possible Change: true personIsHolding a powerweapon

Appendix C: List of evaluation functions the Evaluator uses to test condition in the prototype. These are formatted in C++ style.

```
bool personIsAt(string person, string location);
bool personIsAtSameLocationAs(string person1, string person2);
bool personIsHolding(string person, string object);
bool personCanDoAction(string person, string action);
bool personCanGetAction(string person, string action);
bool personHealthGreaterThan(string person, string healthValue);
bool personHealthLessThan(string person, string healthValue);
bool personMoneyGreaterThan(string person, string moneyValue);
bool personMoneyLessThan(string person, string moneyValue);
bool personIsMale(string person);
bool personIsFemale(string person);

bool personHasStoryRole(string person, string storyRole);
bool isMainCharacter(string person);

bool itemCanDoAction(string item, string action);
bool itemCanGetAction(string item, string action);

bool relationshipPositive(string person1, string person2);
bool relationshipNegative(string person1, string person2);
bool relationshipAverage(string person1, string person2);
bool relationshipParent(string person1, string person2);
bool relationshipChild(string person1, string person2);
bool relationshipSibling(string person1, string person2);
bool relationshipMarried(string person1, string person2);
bool relationshipFamily(string person1, string person2);
bool relationshipSO(string person1, string person2);
bool relationshipStrangers(string person1, string person2);
bool relationshipFriends(string person1, string person2);
bool relationshipEnemies(string person1, string person2);
bool relationshipTeacher(string person1, string person2);
bool relationshipStudent(string person1, string person2);
bool relationshipWorking(string person1, string person2);
bool relationshipCommunity(string person1, string person2);

bool locationIsEmpty(string location);
```

Curriculum Vitae

Name:	Nicholas A. Schudlo
Post-secondary Education and Degrees:	University of Western Ontario London, Ontario, Canada 2008-2012 B.Sc.
Honours and Awards:	Dean's Honor List 2010, 2011, 2012 The University of Western Ontario Gold Medal 2012 Ontario Graduate Scholarship – Queen Elizabeth II Scholarship Summer 2013- Winter 2014
Related Work Experience	Teaching Assistant The University of Western Ontario 2012-2014