Don't Forget to Save!
The Impact of User Experience Design on Workflow of Authoring Video Game Narratives (Transfer Thesis)

Daniel Green

Department of Science and Technology
Bournemouth University

A thesis submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

November 2019

This page is intentionally left blank.

Abstract

Since their inception, video games have been a capable storytelling device. This is only amplified as technology improves. Contemporary video games boast a wide range of interaction and presentational techniques that can enrich the narrative experience. Supporting authors with tools to prototype their stories, or even as direct integration into a game, is vital. This is especially important as the complexity and length of such narratives continues to increase.

Designing these kinds of tools is no easy feat. In order to develop authoring systems for game developers that support prototyping or implementation of their envisioned narratives, we must gain an understanding of the underlying constituents that make up video game narrative and their structural and relational properties. Additionally, when designing the interface of and interactions with such systems, the User Experience (UX) design decisions taken may impact the workflow of the authors to implement their vision. Therefore, we must also gain an understanding of how various UX design paradigms alter the usability of our programs.

This thesis investigates the existing literature on modeling of interactive narrative and video game narrative. Conclusions are drawn upon the ability for existing models to represent the complexities and nuances that are found in video game narrative. Initial work into the experimental identification of UX paradigms and their impact on authoring workflow is also described in detail.

To begin capturing difficult areas of video game narrative, a model of discovered, observed, or experienced narrative, *Discoverable Narrative*, is presented with examples. Additionally, a new theoretical model of interactive narrative, chiefly focusing on video game narrative, *Novella*, is proposed and explored in detail with worked examples. The model is designed on the bases of extension and integration with runtime systems and targets particular difficulties of video game narrative that other existing models struggled to capture. The described models pave the way for further development and research, particularly in a full implementation of *Novella* into a prototype authoring system that will be used in further experiments.

This page is intentionally left blank.

Table of contents

List of figures

This page is intentionally left blank.

List of tables

This page is intentionally left blank.

List of Games

Below is a listing of games mentioned in this thesis sorted alphabetically by series.

BioShock. Irrational Games. 2007.
BioShock 2. 2K Marin. 2010.
Call of Duty: Modern Warfare 2. Infinity Ward, Inc. 2009.
Call of Duty: Black Ops. Treyarch. 2010.
Dragon's Lair. Advanced Microcomputer Systems. 1983.
Half-Life. Valve Corporation. 1998.
Halo: Combat Evolved. Bungie, Inc. 2001.
Halo 2. Bungie, Inc. 2004.
Halo Wars. Ensemble Studios. 2009.
Heavy Rain. Quantic Dream SA. 2010.
Just Cause 3. Avalanche Studios. 2015.
Left 4 Dead. Valve Corporation. 2008.
Life is Strange. Don't Nod Entertainment SA. 2015.
Mass Effect. BioWare. 2007.
Mass Effect 2. BioWare. 2010.
Minecraft. Mojang AB. 2011.
Pac Man. Namco Ltd. 1980.
Pokémon Go. Niantic, Inc. 2016.
Portal. Valve Corporation. 2007.
Return of the Obra Dinn. Lucas Pope. 2018.
Shenmue. Sega-AM2 Co., Ltd. 1999.
Silent Hills: P.T. Kojima Productions. 2014.
Spec Ops: The Line. 2K Games. 2012.
The Elder Scrolls V: Skyrim. Bethesda Game Studios. 2011.
The Stanley Parable. Galactic Cafe. 2013.
The Talos Principle. Croteam. 2014.
Tomb Raider (series). Various developers. 1996–present.
Uncharted (series). Naughty Dog, LLC. 2007–present.
Wing Commander III: Heart of the Tiger. Origin Systems, Inc. 1994.
Zork i. Infocom. 1980.

This page is intentionally left blank.

Declaration

I declare that this thesis has been composed solely by myself and that it has not been submitted, in whole or in part, in any previous application for a degree. Except where otherwise stated by reference or acknowledgment, the work presented is entirely my own.

Daniel Green
November 2019

This page is intentionally left blank.

Nomenclature

Glossary

Authoring Workflow 4
The method by which authors write in terms of frequency of actions taken, focus on particular actions, and the order in which the actions occur.

Command-Based Interactive Fiction 9
Interfacing with interactive fiction by entering commands into a terminal.

Locative Hypertext 9
Hypertext systems that are location dependent.

Acronyms / Abbreviations

DSL 29
Domain Specific Language.

ELO 25
Electronic Literature Organization.

HCI 4
Human-Computer Interaction.

HTML 24
Hypertext Markup Language.

HTTP 24
Hypertext Transfer Protocol.

IDN 3
Interactive Digital Narrative.

NPC 10
Non-Playable Character.

UX 4
User Experience.

This page is intentionally left blank.

“I like the cover,” he said. “Don’t Panic. It’s the first helpful or intelligible thing anybody’s said to me all day.”

— Douglas Adams, The Hitchhiker’s Guide to the Galaxy

This page is intentionally left blank.

Publications

Green, D., Hargood, C., and Charles, F. (2018). Define "Authoring Tool": A Survey of Interactive Narrative Authoring Tools. In *Authoring for Interactive Storytelling 2018*. Springer.

Green, D., Hargood, C., and Charles, F. (2018). Contemporary Issues in Interactive Storytelling Authoring Systems. In *Proceedings of the 11th International Conference on Interactive Digital Storytelling*, ICIDS 2018.

Green, D. (2018) Novella: An Authoring Tool for Interactive Storytelling in Games. In *Proceedings of the 11th International Conference on Interactive Digital Storytelling*, ICIDS 2018.

Green, D., Hargood, C., Charles, F., and Jones, A. (2018). Novella: A Proposition for Game-Based Storytelling. In *Narrative and Hypertext 2018*. ACM.

Jones, A., Gyori, B., Hargood, C., Charles, F., and Green, D. (2018). Shelley's Heart: Experiences in Designing a Multi-Reader Locative Narrative. In *Narrative and Hypertext 2018*. ACM.

CHAPTER 1

Introduction

Narrative can be found in many aspects of our daily life and has evolved alongside us as a species. Whether it be in the form of discovered cave paintings depicting instinctual survival, ancient hieroglyphics forming some of the first known texts, the contents of a good book, or even browsing the news over breakfast, all of these forms have one thing in common: they describe a narrative; they tell a story. These examples, along with countless others, can be verified by the works of Bruner who identified ten features of narrative [18]. Within these elements, Bruner highlights the diachronic nature of narrative as a sequence of unfolding events (although they are not necessarily arranged temporally), and in particular notes that nonverbal media such as cartoon strips or cathedral windows also follow this diachronicity. Among his other points, it is argued that a recounting of a sequence of events does not necessarily constitute a narrative, but that they must consist of purpose, motivations, and may be interpreted in many ways.

The novel has existed in its current form for quite some time. Traditionally, this is presented as a non-interactive, linear story from one end to the other without any intervention from the reader. One of the notable innovations in paper-based novels was the evolution of gamebooks such as the *Choose Your Own Adventure* [61] series, which incorporated reader choice into an otherwise static form.

The computerized equivalent to this, Interactive Digital Narrative (IDN), is able to leverage the interfacing abilities and flexibility of computer systems for storytelling. This means that IDN can provide a rich multimedia experience unachievable through other mediums. The early text adventure game *Zork*[1], for instance, combined narrative often found in novels with the interactivity of computer terminals. Players took control of a virtual avatar by entering commands into a computer terminal, exploring the environment and solving puzzles along the way, as well as making story-critical choices. The player could revisit previously encountered scenes that may now differ based on items they have collected, quests completed, time elapsed, and other parameters.

[1]Zork i. Infocom. 1980.

Computers are becoming more ubiquitous and the interfaces to such systems are ever increasing. Consequently, the complexity and scale of stories that we are able to produce, and the many methods of interaction with them are likewise growing. With such growth, it becomes more difficult to manage and plan such intricate and involved systems. To counteract this, we must leverage the available technology to develop user-friendly tools to enable authoring, testing, and management of old, contemporary, and emerging methods of storytelling in the interactive realm.

Video game writing in particular is a branch of IDN that is currently undergoing rapid expansion. The complex interaction and presentation methods found in video games make for intriguing challenges when developing tools to assist writers.

When developing authoring tools for any interactive narrative, we must first break down the story into its constituents. This allows us to form a theoretical model. These models exist at all kinds of levels of detail with differing purposes, but all share one common goal: to abstract the story and represent it with some kind of logical structure. This raises the question of how to create a genre-agnostic model of interactive fiction for video game narrative, and at what level of detail to represent stories at. Additionally, contemporary models struggle to capture elements of video game narrative that can be optionally discovered or experienced rather than those integral to the story (findable books, letters, audio diaries, and so on). We must also consider the implications of integrating such models into authoring tools, as theoretical systems are not always practical.

The models themselves, while important, are not what the end-user interacts with directly. The user interfaces and User Experience (UX) design of such systems is equally of vital importance. While each authoring program is unique, they can share common interface paradigms. Although the concept of what is considered a 'good' interface is subject to opinion, the impact of various interface paradigms and UX techniques on the workflow of authors to create their desired stories is something that we can test for. Understanding which interface paradigms often hinder authors and which aid them can help us to make more intelligent decisions about the interface and UX design of our tools.

1.1 | Research Questions and Scope

The origin of this thesis lies in the desire to represent narrative in games and to develop an understanding of what impact our design choices have on authors. While video game narrative will be a primary focus, this thesis will consider areas of related interest such as narratology and Human-Computer Interaction (HCI). Throughout this thesis, I will explore the following research questions:

1. How do existing models of interactive narrative fail to capture the content of game narrative and how might a new model capture this?

2. What are the HCI requirements of a new authoring tool supporting a new narrative model of video games?

3. How do user interface paradigms as part of an authoring tool impact the UX of the creative process of authoring game narrative?

The first of these questions is dedicated to understanding how existing approaches to narrative modeling capture video game narrative, and to what capacity they fail to capture the nuances and details of game narrative. It also looks at how we may build upon these approaches to create a new model that does capture them. The second question seeks to discover requirements for the development of new authoring systems implementing such models as described above. The third question focuses on identifying the relationship between user interface paradigms and their impact on UX, particularly surrounding the alteration of the workflow of the author's creative process in such authoring tools.

1.2 | Thesis Structure

In the second chapter, I will describe what kind of games this thesis tackles as well as explore the nuances of video game narrative and what makes it a difficult medium to represent. In the third chapter, I will discuss the rise of interactive narrative from its origins in hypertext and narratology and provide an examination of numerous authoring systems. In the fourth chapter, I will discuss approaches to modeling video game narrative, as well as present my first major contribution, *Discoverable Narrative*. In the fifth chapter, I will describe my second major contribution, which is a theoretical model of interactive narrative chiefly targeting video games. The sixth chapter will conclude the thesis and detail upcoming and in-progress work.

This page is intentionally left blank.

CHAPTER 2

Video Game Narrative

In order to correctly analyze and build narratological models of video games, we must first broadly understand what games are, if and how they are considered narrative experiences, how they differ from other forms of storytelling, and note any potential difficulties that may arise when modeling them. We must also pay attention to the relationship between the non-narratological components of games and how they support the narrative experience of players. This chapter will look at these topics to give us an understanding of video game narrative such that we can then begin to model it in a way that captures its nuances.

2.1 | What is a Game?

The concept of a 'game' is not something that can have a single, agreed definition. Different cultures will interpret such a phrase differently, as will different generations. If we take the relevant base definition from the Oxford Dictionary[1], it is stated that the noun *game* means "An activity that one engages in for amusement". This does not get us far. Playing Cricket is a game, we can all agree on that. Playing the latest video game is also a game. However, by this definition, playing a musical instrument - an activity that is often done for amusement - is also a game, but most would likely not consider this an actual game.

This ambiguity is echoed by Avedon in his in-depth discussion of what games are structurally [5]. Avedon goes on to identify and list ten factors of which games are composed. First is the purpose of the game; what's the aim, goal, or intent? Second is the procedure for action, i.e., what methods are involved in play. Third are rules governing action or fixed principles that determine conduct and behavior. Fourth is the number of required participants, both minimum and maximum, for actions to take place. Fifth is identifying the roles of each involved participant in terms of their functions and status. Sixth is the results or payoff assigned to any defined action. Seventh lists the abilities and skills required to complete an action in the cognitive (cognition, memory, etc.), sensory (motor skills, coordination, etc.),

[1]Definition of 'game' taken from the Oxford Dictionary as of writing.

and affective domains (semiotics that stimulate emotions). Eighth defines interaction patterns, which are the methods of interaction between players and the game divided into eight subcategories. Ninth explains the physical and environmental settings and requirements for the game to take place, although Avedon notes that this element is not always present. Finally, the required equipment employed to complete actions is listed, and as with the ninth, may not always be present. This definition does aid us in refining our definition of what games are. We now have a formal set of rules that we can apply to determine what is and is not a game, and what each game contains. If we then apply these rules to our musical instrument example, some factors become difficult to answer unless stretched, which is what we want. If we instead apply them to activities traditionally thought of as games - board games, tabletop games, playground games, video games - then we can satisfy each factor without trouble.

However, this definition, while useful, is still perhaps a little too broad. We need clarification on exactly which kind of "game" this thesis will tackle. We are particularly interested in games that are of a digital nature. Games that are played electronically and require some kind of input device. A digital video game. Therefore, for the purposes of this thesis, I define *video game* as an extension of game as:

A game that is played electronically and is manipulated with interactions using some form of input device that generates visual or otherwise notable feedback.

This definition, by design, excludes physical gaming such as board games, as they are not the focus of this thesis. What is captured is any form of interactive entertainment with input and feedback. This covers both interactive fiction as well as contemporary digital games such as those found in the PC or console gaming markets.

2.2 | The Ludology vs. Narratology Debate

Now that we have defined the kind of games this thesis will tackle, the next challenge is this question: Can these kinds of games be considered narrative experiences, and can we apply narratological methods to them? The answer is to both of these questions is broadly yes; we can look at these games, or at least components of them, from a narratological point of view. However, it has not always been this simple.

For quite some time, there was a heated debate surrounding the most appropriate methodology for the study of games [2, 12, 27, 30, 39, 40, 68]. This debate came to be referred to as the "Ludology vs. Narratology" debate [39]. Ludologists believed that games should be examined purely by their components and systems that they create, caring little about the game's story seeing it as a secondary to the gameplay, mechanics, and so on. On the other hand, narratologists looked at games as another form of narrative expression, inverting the position of ludologists by seeing the story as the primary component and the gameplay, mechanics, etc., as a way of interacting with it, and as such, that games can be analyzed as a derivative of narrative using narratological theories. There are numerous instances of back-and-forth academic squabbling surrounding the topic, of which further fueled the discourse. Instead, let us look at the outcomes of such debates.

It is now accepted that not only was the debate misguided, but it has actually hurt the field by creating a division of ‘us vs. them’ [2, 12, 68]. As pointed out by Aarseth [2], games are complex software that can emulate other mediums, including those that are considered narratological such as film, graphic novels, and text novels. While some points raised during the debate will remain undecided, the consensus is now that while not every game may tell a defined story, they are most definitely a medium capable of telling intricate and involved narratives (and as we will see, introduce unique storytelling methods not found in other fields). An important point to conclude the debate is that while academics were occupied debating whether or not games can be considered a derivative form of narrative, the games industry and consumers were busy discussing how great the contained narratives were [68].

2.3 | Difficulties of Video Game Narrative

Before we tackle the challenge of modeling video game narrative, we must first gain an understanding of the broad challenges that this form of narrative expression has. By understanding these, we can better develop models by taking into account the observed challenges.

As mentioned earlier, video game narrative is able to emulate many other forms of narrative expression. For instance, locative hypertext can be found in *Pokémon Go*[2], and command-based interactive fiction can be found in the *Zork* Easter egg of *Call of Duty: Black Ops*[3]. This means that when modeling video game narrative, we must take into account that it can exhibit numerous factors often prominent in other fields. This magnification and combination of difficulties found in other fields is a common occurrence within video games. Sometimes video games incorporate unique challenges in addition to borrowing from other fields, particularly around methods of presentation and user interaction.

The complete analysis and breakdown of video game narrative difficulties for modeling is unto itself a contribution. Instead, the following discussion intends to highlight a limited number of key challenges that make video game narrative difficult to model rather than create a taxonomy listing all factors. With this list considered, we can then begin to investigate the modeling process with these challenges in mind.

2.3.1 | Narrative Methods

There are many ways that narrative can find its way into games. Two of these methods are Embedded and Emergent narrative.

[2]Pokémon Go. Niantic, Inc. 2016.
[3]Call of Duty: Black Ops. Treyarch. 2010.

Both Embedded and Emergent narrative were first introduced by Marc Leblanc in his 2000 Game Developer's Conference talk[4], and have since been refined, notably by Jenkins [39]. Embedded narrative refers to the authored portion of the narrative within a game. That is, the systems created by the developers working together as intended to build a narrative experience. At its extreme, the player has no control over this form of narrative. Cutscenes, for example, restrict player control to ensure that certain narrative content is presented. Emergent narrative, on the other hand, refers to the narrative that players create themselves through the act of play and imagination that is not explicitly authored by the developers. At its extreme, players take the game out of its intended narrative using the systems and tools provided by the game to make their own narrative experience. The fan-made Red vs. Blue[5] series is a prime example of this, which originally took systems of *Halo: Combat Evolved*[6] and used them in ways not intended by the developers to create their own complex narrative loosely based on the game's lore. I will now refer to this form as *Direct Emergence* due to the player's key role in the creation of the narrative experience.

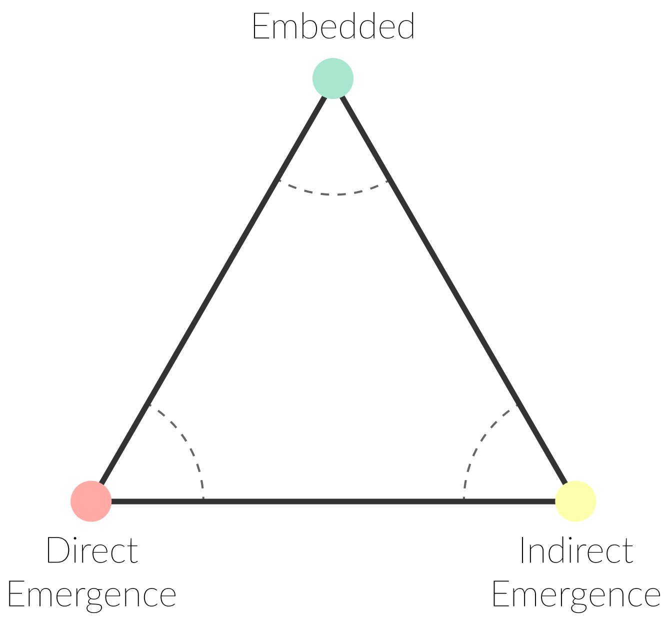


Fig. 2.1 A narrative form triangle showing the three primary types.

Another important form of emergent narrative is when designed systems of a game act in seemingly autonomous and not explicitly authored ways, creating vignettes of narrative. For example, in *The Elder Scrolls V: Skyrim*[7], it is not uncommon to witness a bandit being chased by city guards, or hunters attacking animals, and other similar non-playable character (NPC) interactions. These subsystems of the game all have defined behavior, often dynamic, but are not explicitly author narrative by the designers. As such, they are encountered and interpreted narratives that may or may

[4]'Formal Design Tools: Emergent Complexity, Emergent Narrative' at GDC 2000.
[5]Red vs. Blue is a highly successful machinima based on *Halo*.
[6]Halo: Combat Evolved. Bungie, Inc. 2001.
[7]The Elder Scrolls V: Skyrim. Bethesda Game Studios. 2011.

not be experienced by players. I will now refer to this form as *Indirect Emergence* due to the lack of direct player influence on the instigation of the narrative happenings.

While both direct and indirect emergence can be considered similar, I think that distinguishing between them is important as the methods of instigation are different. Additionally, it is possible that Indirect Emergence is never experienced by the player, whereas Direct Emergence is naturally experienced due to the player's role in creating it.

Figure 2.1 demonstrates a narrative triangle. Each of the vertices represents one of the three forms of narrative expression in games. As a single narrative occurrence is rarely a single extremity of one of the categories, we can use barycentric coordinates of this triangle to broadly represent the form of a given narrative occurrence by weighting it within the triangle.

2.3.2 | Structure

Games often involve a form of travel around a virtual environment. Sometimes this can be abstract such as in interactive fiction adventure games, where the player must specify the direction the avatar moves in the virtual world (often between rooms or scenes) to which the location is described, allowing the player to build up a mental map of the environment and its connectedness. More explicitly players can directly control avatars through virtual worlds, such as in first-person shooter games.

In his book *Fundamentals of Game Design* [3], Adams highlights a number of common patterns that have emerged in the structure of level design. These factors are important, as the environment in which the player experiences their narrative must be considered when modeling, especially when the structure is able to take effect on the narrative. If the structure allows for players to revisit previously accessed locations but with new information, the narrative experienced may differ from the first time. In some structures, this may not be possible, therefore it is of importance that we consider these common structures.

Open Layouts

In open layouts, players are almost unconstrained from movement and are able to wander in any direction at almost any time. These kinds of layouts are typically found in open-world games such as role-playing games and to some extent are found in multiplayer shooting games. Predefined narrative is difficult to craft around such layouts as the player has high levels of freedom and may not follow defined paths. As a consequence, this form of layout is an ideal source of emergent and procedural narrative as the player interacts with the world. An example of this layout can be found in *Just Cause 3*[8], which sees the player control their avatar through a vast open landscape with the freedom to roam almost anywhere at any time. This sandbox environment lets players create their own narrative through their interactions with

[8]Just Cause 3. Avalanche Studios. 2015.

the world, and observe procedural narrative as the game's systems interact with one another, such as spontaneous battles between rival factions unexpectedly breaking out, of which the player can optionally join in and assist their own side.

Linear Layouts

Linear spaces are connected constrained segments in a fixed sequence without any side corridors or branches. In this form, players are only able to traverse forward and backward between predefined spaces (in some, even backward movement is denied). Games of this form are often said to be *on rails* due to the lack of divergence from a given path, akin to trains. This form of layout restricts player agency by constraining their choices to not be able to alter the narrative pathway, but does allow for fine-tuned authored narrative, particularly that which is cinematic. If discussing the overall story arc, a number of games conform to a linear structure. *Half-Life*[9], for example, when observed at a narrative arc level, progresses the player from space to space with the same beginning, middle, and end each time.

Parallel Layouts

Parallel layouts are similar to Linear layouts, but they introduce the concept of multiple tracks for the player to choose from. The beginning and end are consistent, but the pathway that each player may take throughout is not fixed. Additionally, if the game supports bidirectional flow, then it enables players to explore by visiting alternative routes after already picking one. This form retains most authorship of the designers while injecting increased agency and freedom of the player. A prime example of this form of layout can be found in *The Stanley Parable*[10], where taking the beginning and one of many endings is fixed, but the pathway to it is not.

Cyclic/Ring Layouts

Ring layouts are a misnomer as they suggest that they are restricted to circle-like shapes, but in fact are not restricted by geometry. Instead, I find that 'Cyclic Layouts' is more of a fitting term. Regardless of name, these layouts feature paths that return to their starting point with optional shortcuts between portions of the cycle. As they allow for repeated experiencing of a narrative space, it is possible that the experience may change on each viewing as more information about the narrative is obtained. These layouts are often found in lap-based racing games. In *Silent Hills: P.T.*[11], the player traverses through a house's corridor which continuously loops and redecorates itself each time, and progressively horror unfolds. This sequence is eventually broken but demonstrates a looping narrative in which each cycle is different from the last.

[9]Half-Life. Valve Corporation. 1998.
[10]The Stanley Parable. Galactic Cafe. 2013
[11]Silent Hills was a Playable Teaser (P.T) as part of the Silent Hill franchise Developed by Kojima Productions in 2014, to be published by Konami, but was unfortunately canceled.

Network Layouts

Discrete spaces that are interconnected in a variety of ways are referred to as Network layouts. Networks give the player freedom for which path to take and encourage exploration due to the interlinked segments. Control of the player's flow can be increased by tweaking if, how, and when spaces connect. This layout is often found in multiplayer first-person shooter levels where each distinct area has multiple connections to other areas to allow for flow control of players and increased chance for tactics to be devised by players.

Hub-and-Spoke Layouts

In this form, there is a central zone that acts as a hub, which connects to a number of other spaces acting as the spokes. In order to explore spokes, the player must move from the hub, and if they want to explore another spoke, must first return to the hub. Not all spokes have to be unlocked at once and can be controlled by designers to gradually feed the player new opportunities to access from the hub. If returning to the hub at the end of a spoke is done by backward traversal, there is the opportunity for the space to be experienced in a different way to before with the player having gained new narrative information at the end of a spoke. By providing multiple spokes, players are given choice, although some authorial control over the player is still retained. An example of this layout can be found in *The Talos Principle*[12] which situates the player within three nested hub-and-spoke layouts, where each spoke eventually resolves to a single puzzle that must be solved to unlock further puzzles.

Combinations

Many games use more than one layout, and commonly combine them by nesting them within one another. Great complexity can come from nesting otherwise simple structures. All of these available options (and more) mean that games have a greatly varying selection of structural patterns to their design and narrative. Player agency over the narrative differs based on the structure that is being employed, as does authorial control over the narrative.

2.3.3 | Presentational Methods

If we say that instead of games being narratives, they contain various narrative forms as a result of their many systems and the way in which they interact, then it comes with no surprise that games provide a vast range of techniques for presenting narrative to players. While some of these techniques are shared with other fields, a number of them are (currently) unique to the way in which games portray their narrative experiences. These are critical to understand, as the methods available to portray narrative to the player means that we must consider these factors in our

[12]The Talos Principle. Croteam. 2014.

modeling process. The terminology of these various presentational methods have no single concrete definition, and as such, are my own personal interpretations based on knowledge of video games.

Cinematics

Games can deliver narrative through pre-rendered sequences. These are often streamed full-motion videos (FMVs). During playback of such sequences, player control is almost entirely restricted, save for essential video interactions (play/pause, skip). Cinematics are often used to highlight specific moments of narrative significance, such as an introduction to set the context for the happenings within the game's proper. Sometimes, particularly in older titles, real-life video footage was incorporated into the game as FMV sequences to contribute to the game's narrative. For instance, in *Wing Commander III: Heart of the Tiger*[13], the space combat gameplay was intercut with live-action footage. In the past, cinematics were drastically more frequent perhaps due to real-time technical limitations. For example, *Halo Wars*[14] had high quality pre-rendered FMV sequences between levels to inform the player of the narrative progression, most likely due to technical restrictions (i.e., being unable to reach the same quality in real-time using the game's rendering engine). In contemporary games, while cinematics are still used, they are gradually being replaced with in-game cutscenes as the gap closes between the quality of real-time rendering and pre-rendered sequences. Additionally, the cost of modifying a pre-rendered sequence is high as the entire sequence needs to be composited again, whereas real-time has no such limitations. Additionally, the flexibility gained by real-time rendering allows for better visual integration with the game's regular graphical style and allows for dynamic content such as themed characters based on player choice.

Cutscenes

Cutscenes refer to in-game sequences that temporarily restrict player control or the active play area for the purpose of showing something of narrative significance. They differ from Cinematics in that they are not pre-rendered and do not have to entirely restrict control of the player. Since this definition is rather broad, we can further scope it with some examples.

Sometimes cutscenes directly cut away from the game's proper in favor of showing the cutscene. In this situation, they are comparable to Cinematics with the difference being that they are self-contained within the game. For example, in the original *Pac Man*[15] gameplay was periodically suspended to show an in-game animation of the Pac Man avatar being chased by ghosts only to return larger chasing them. This is also considered one of the first uses of modern cutscenes in games.

[13]Wing Commander III: Heart of the Tiger. Origin Systems, Inc. 1994.
[14]Halo Wars. Ensemble Studios. 2009.
[15]Pac Man. Namco Ltd. 1980.

Other kinds of cutscenes suspend the player's controls and/or area of accessibility either through restriction of movement or restriction of the available play area to deliver in-game sequences that are of narrative significance. This method is popular in games that are considered to be highly cinematic[16]. In *Act I - Team Player* of *Call of Duty: Modern Warfare 2*[17], the player is faced with an out of service destroyed bridge making it impassible. When the player enters a vehicle on the bridge, they must then wait for an alternative path over the bridge to be laid. Once complete, the player's squad waits for a friendly airstrike to complete on the city ahead before moving on. During this sequence, there is constant chatter and casual behavior (such as marines filming the airstrike on their phones) adding to the believability and narrative backstory of why the player is about to attack a city.

A third variant of cutscenes is those that are interruptible. These sequences are regular cutscenes but with additional control given to the player to, at key points, make decisions that alter the way the cutscene may play out. The effects of such decisions may or may not have longstanding significance, but do alter the perceived narrative of the player by potentially unlocking different pathways or additional segments of otherwise unseen narrative. For example, in *Mass Effect 2: Lair of the Shadow Broker*[18], as the player and Liara are walking back towards their ship in a dialogue-heavy cutscene, the player is able to optionally take a series of interrupts, each of which sees a unique segment of dialogue and interaction between the two characters take place. If the player and Liara are considered romantically interested from the previous game, then an additional interrupt is available where the player can discuss the status of their relationship given the events that have transpired. These sequences are similar to quick-time events but differ in that they exist specifically within a cutscene. It is perhaps more fitting to say that these narrative sequences contain quick-time interactions embedded within them.

In-Game Events

Narrative can also be delivered via player interactions with the world and as a result of the simulation that the game is running. This encapsulates anything within the game of narrative significance that is a result of direct interaction or that can be observed passively. Every action that the player takes can contribute to their interpretation of the narrative, and result in their own emergent narrative (particularly when involving multiple players). Additionally, players can interact with a whole subset of systems within games to experience narrative, such as branching dialogue trees, where players actively participate in dialogue by altering the path taken by their virtual avatar. Authored scenarios that are not of primary importance to the story can also contribute to the narrative experience a player receives. This is often content added in to make the world feel active and alive; authored situations to make the

[16]Cinematic in the sense of cinematography rather than my presentational definition above.
[17]Call of Duty: Modern Warfare 2. Infinity Ward, Inc. 2009.
[18]Lair of the Shadow Broker downloadable content for Mass Effect 2. BioWare. 2010.

world more believable. For instance, on the planet Illium in *Mass Effect 2*, the player is able to observe various NPCs bartering with other NPCs at stores. The player is able to listen in and find out backstory about the bartering, the economy of the city, the environment, the inhabitants, and in some cases even use the information gained within missions. Elements of the earlier discussed Procedural Narrative can also contribute as an in-game event, where the simulation of a game results in unplanned narrative segments.

Loading Screens

A surprising form of narrative is that included within loading screens. This form is being increasingly used, especially since games have been able to efficiently allocate and retain loaded resources to memory in order to display interesting and sometimes interactive loading screens, whereas in the past loading screens were static text at best. Often this form is complimentary by simply enhancing the background of the main narrative. *BioShock*[19], for instance, has a themed loading screen according to the game's setting and has various rotating quotes from characters involved in the narrative that provides an insight into the game world. There are situations where the loading screen is incorporated into a level introduction cinematic, such as in the mission briefing loading screens of *Call of Duty: Modern Warfare 2*. Some games hide a number of loading screens with restricted gameplay in which narrative can be communicated during the invisible loading sequence. In *Mass Effect*[20], radios present in the loading elevator rides deliver background information to the player about the state of the universe for the duration of the ride, and in some cases even grant the player quests. However, such forms can perhaps be seen as a hybrid or even as a type of cutscene.

On-Screen Information

Another oddity for narrative presentation is that of on-screen information shown to the user. Generally, on-screen information may not be of narrative significance (gameplay instructions, directions, and such). However, there are cases where such information provides a narrative experience too. In *The Talos Principle*, the beginning of the game is accompanied by on-screen text as the player traverses the tutorial/introduction. The on-screen text reads a number of sequences such as 'Initiating child program logic check. . . ', which helps to reinforce the narrative that the player is an AI robot.

Discoverable Items

Since games take part in a virtual world that the player can traverse around, there arises the opportunity to hide narrative for the player to discover. This optional

[19]BioShock. Irrational Games. 2007.
[20]Mass Effect. BioWare. 2007.

narrative may or may not be found and experienced by players, and as such, we must factor in the possibility into our models, as well as how players are able to find them. These items may be used for providing background to players that isn't of core importance to the narrative but enriches it as a reward for those willing to explore. This kind of narrative delivery is popular in open world games, such as the hiding of books, letters, and other similar items, each of which provides some kind of insight into the game world. A unique example of their use can be found in *The Talos Principle*, which scatters painted QR codes around the game world, which when observed will show a message to the player relating to the game world. Interestingly, the QR codes in the game actually represent the text's QR code if scanned with a camera.

Environmental Storytelling

The concept of environmental storytelling, popularized by Carson [21], borrows from theme park design. Carson explains that *"one of the secrets [of designing] entertaining themed environments is that the story element is infused into the physical space"*. He documents a number of tips and tricks for effective environmental storytelling, many of which rely on manipulation of the audience's expectations. For instance, Carson describes *Cause and Effect* as a way of generating interpretative narrative based on mise en scène. By carefully arranging and staging scenarios, the environment leads players to come to their own conclusions about the past events that have transpired. *BioShock 2*[21], known for its immersive environments, demonstrates this in Figure 2.2. This game is set in a dystopian, rundown underwater city after civil war had broken out. All sorts of inhumane things have happened and the city is on the brink of collapse. The environment is designed to reflect this by carefully curating the placement of environmental factors such as rubble, fallen pillars, broken glass, and so on, to hint to the player at the horrors that have happened and the potential volatility of their surroundings and future endeavors.

This concept was later built upon by Jenkins [39]. In relation to environmental storytelling, Jenkins further highlights the reliance player preconceptions in his *evocative spaces*. This is when the environment does not necessarily tell the entire story but relies on previous knowledge of players to invoke familiarity. He notes that carefully curated environments can be a source for emergent narratives that players themselves create through interpretation.

2.3.4 | Interaction Methods

In addition to having a variety of presentational methods, video games expose some interesting methods of interaction to players. A number of these give the player agency over controlling the narrative flow. Therefore, we must consider them in our modeling process.

[21]BioShock 2. 2K Marin. 2010.

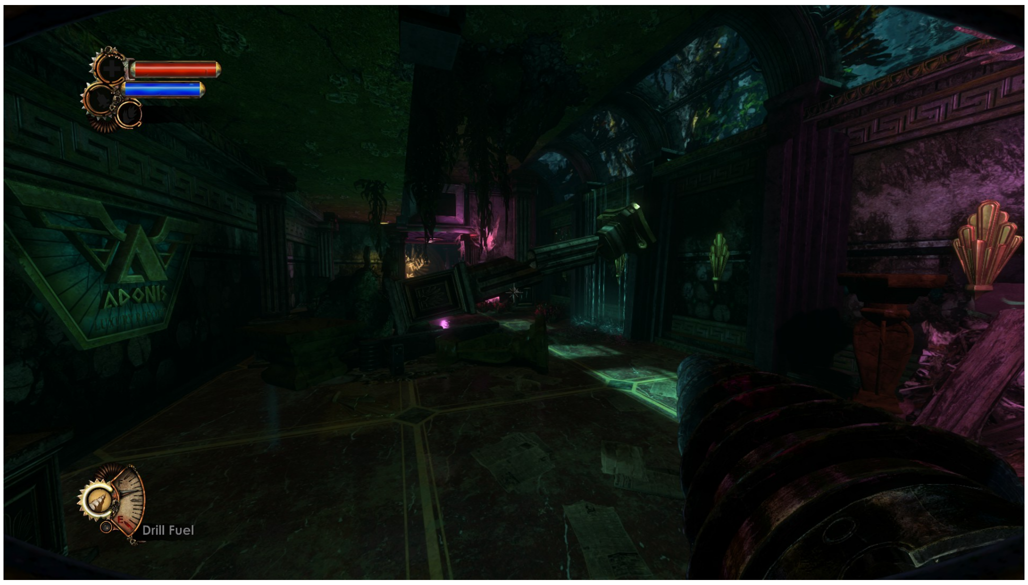


Fig. 2.2 Environmental storytelling in *BioShock 2*.

In Adams' *Fundamentals of Game Design* [3, p. 170], he describes the concept of Immediate, Deferred, and Cumulative influences the player can have on the narrative as in-game events take place (usually as a result of player choice or interaction). Immediate choices have direct and instant alternation to the narrative; a usually irrevocable decision of which road to take through the narrative. Deferred choices have an impact on the story not at the time of decision but instead at some point in the future. Cumulative choices are a series of decisions that build up throughout the game, such as player actions and choices, that result in some branch of the story. They are in a way like groups of deferred choices working together to alter the narrative as a collective, but may not have a significant impact on their own. We can then extend Adams' definition to include not only in-game events, but also any form of user interaction as to how it alters the narrative.

Quick-Time Events

Quick-time events are short-lived sequences presented to the player in which they must respond to some given prompt requesting input. The result is binary. The player either achieves the requested prompt or does not. Based on the player's success, the narrative is able to branch, and therefore by participating (or not) in these events, the player is able to alter the direction of the narrative. Some prompts are not timed but instead pause the game indefinitely while the player freely chooses an option. In *Life is Strange*[22], major decisions that will alter the course of the narrative in significant ways completely halts gameplay while the choice is made. This is not considered a quick-time event and is instead a form of Explicit Player Choice (see below).

[22]Life is Strange. Don't Nod Entertainment SA. 2015.

One of the earliest games to feature quick-time events was *Dragon's Lair*[23]. This game, or more accurately interactive movie, consisted of almost entirely quick-time events that the player had to achieve, else they would often meet their demise. These quick-time events were immediate in effect.

The prompts presented to the player may not always be in the form of a simple button press. An example of this can be found in *Heavy Rain*[24], whose gameplay largely revolved around sequences of rapid succession of immediate quick-time events. Instead of the quick-time events being satisfied by a simple button press, they involved more advanced conditions such as shaking the controller within a given time frame or rapidly pushing the controller itself down. While the output is still binary, these additional methods do provide the option for quick-time events to no longer be binary. In the case of the controller shaking, multiple paths could be taken based on how much the player shook the controller versus the desired amount.

In some cases, quick-time events can be used against the player's instinct for narrative purposes. In *Shenmue*[25], the sequence where the main character, Ryo, is in a barber shop having a philosophical lesson reverses the expected goal of quick-time events. The barber instructs the player not to move and brings a blade to Ryo's throat, which is followed by an extended quick-time prompt. If a button is pressed, which is the instinct of many players, the sequence is failed. Only if the player goes against the quick-time event and purposely ignores it does the game continue. This mechanic is used as a metaphor in the game for tranquility and remaining calm under pressure as is the topic of the lesson being taught to Ryo.

Temporal Events

Video games often have living worlds. Worlds in which time actually passes. This property can be used for enhancing the believability of the game world as well as providing additional gameplay benefits. *Shenmue* demonstrates this admirably. The game uses time as a way of gating the player's access to certain quests and increases the realism of the world by making citizens appropriately act based on the time. Certain shops the player must access for quests will have actual opening times that are honored based on the in-game clock. Similarly, during the day, the player can expect to see citizens going about their daily life, but as night comes, the people remaining outside become more dangerous and different shops open. Temporal settings can also affect emergent gameplay. In *Minecraft*[26] during the day cycle, players generally gather resources and build structures, but the night cycle is hostile and the player faces great enemies and therefore must change their play style accordingly.

[23]Dragon's Lair. Advanced Microcomputer Systems. 1983.
[24]Heavy Rain. Quantic Dream SA. 2010.
[25]Shenmue. Sega-AM2 Co., Ltd. 1999.
[26]Minecraft. Mojang AB. 2011.

Triggered Events

Since video games are an interactive form, they are able to take advantage of dynamic events based on the state of the game world that otherwise may not occur. These could happen when the player moves into a specific space, or when a certain set of other conditions are met. The player is usually not aware of how these events are triggered, and they are often authored. Their main purpose is to provide some kind of narrative information about something that is happening in the game world without necessarily having to stop gameplay to deliver it. The player has no real agency over these events other than to avoid, if at all possible, the conditions that activate the event. Often the results of activating such a trigger are irreversible by the player such as buildings collapsing or bridges falling. In series such as *Tomb Raider*[27] and *Uncharted*[28], it is common for structures that appear to be on the brink of collapse stay perfectly stable indefinitely until the player reaches a certain spatial position. Once they reach that location, however, the structure will often irreversibly be destroyed.

Dialogue Trees

Dialogue is almost inevitable in games that involve characters. Conversations between characters increase the believability of the game world and make the player feel more involved. They are also a primary means of narrative delivery, particularly in role-playing games where interaction with other characters is essential. They are also an ideal wagon for providing the player either staged or actual agency of the narrative of the game. When the player's avatar is to respond or instigate a conversation, games are able to present the player with a number of choices, with each branching the story (or at least the current situation). Due to the dynamic nature of games, the available choices can be not static, such as being based on the player's morality or previous activities. Staged agency can be achieved here by providing temporary branches based on player response that quickly reconcile, giving the player the illusion of agency while still keeping them on an authored pathway. In other situations, the player's choice can be impactful and long-lasting. In *Mass Effect*, a heated debate with the character named Wrex - a member of the player's squad - can result in the character being convinced but remain dissatisfied with the player throughout the remainder of the game, or in a worst case, can die, meaning he is no longer part of the narrative which has dramatic impacts on the story both in the moment and later on in the game.

Explicit Player Choice

Players are sometimes given the freedom within a game to make explicit choices. Mason introduced the terms *diegetic* and *extra-diegetic* in the context of interactive

[27]Tomb Raider (series). Various developers. 1996–present.
[28]Uncharted (series). Naughty Dog, LLC. 2007–present.

narratives to represent these kinds of choices [54]. Diegetic choices represent those that players can make as a character or presence within a story world that affect the story, whereas extra-diegetic choices are those that are made as a removed observer from the story world. In video games, generally speaking, diegetic choices are those made without pause of the gameplay, and extra-diegetic choices suspend the gameplay for either a limited or indefinite amount of time while the choice is made. This is akin to the original description of a presence or observer within the story. The use of diegesis in video games is not a new concept [71, 4] and has been explored in detail, but Mason's model expanded this concept to represent player choice and its separation from gameplay.

A number of diegetic choices can be found in *Spec Ops: The Line*[29]. One scene, for instance, has the player confront a friend-turn-foe trapped under a jackknifed tanker with an explosion imminent. The trapped character begs the player for forgiveness and asks to be put out of his misery. However, the player is able to, during gameplay, optionally shoot him and fulfill his wish, or walk away and leave him be. The choice to comply with the man's requests is completely at the choice of the player, and the choice is made as an active participant within the game world.

A common use of extra-diegetic choice in games is through dialogue trees. *Mass Effect 2* is an ideal example of this, as it is heavily dialogue-centric. In this game, dialogue options pull away from the gameplay by halting the game's progression while a choice is presented to the player. If the player does not make a choice, the game world will wait indefinitely; if a battle is raging on in the background and timing seems urgent, waiting will not cause any negative effects.

Player Traits

While not necessarily a method of direct interaction, players are able to alter narrative through traits associated with their avatars. These traits alter the narrative in a variety of ways, often personalizing the player's personalized narrative experience by tailoring it to their own play style, which can increase immersion. We must consider this when modeling the narrative, as it must be able to respond upon player traits.

For example, in *The Elder Scrolls V: Skyrim*, guards react differently based on the player's race, form (human, vampire, werewolf), bounty, or other states (such as approaching with weapons drawn). Players of the Nord race may experience a guard say something warming like "How can I help a brother Nord?", whereas a Khajiit would experience discrimination along the lines of "What do you want... cat?". This can deepen the player's immersion into the game world and into their avatar.

A similar effect can be found in *Mass Effect 2*. The player's morality, based on their actions in the world, directly alter which conversation options are available during conversation. The meaner a player acts, the more mean options become available (and conversely, the less good options are available). A more explicit

[29]Spec Ops: The Line. 2K Games. 2012.

example of this is in *BioShock*, where the player's morality exclusively alters which ending the player receives upon completing the game.

2.4 | Conclusion

We have identified the working definition of *video game* that this thesis will tackle. We have also declared that while video games themselves may not be a direct form of narrative expression, they are certainly capable of it and can be analyzed from both a ludologists' point of view as well as have narratological theory applied in harmony. Finally, we explored a number of features that video games have that demonstrate the complexity that must be considered when designing models to capture video game narrative. The levels of agency and dynamic nature of games means that we must be considerate to not only directly authored narrative, but also that whose execution is not controlled directly by the author, but instead is a result of the game's various systems interacting with one another.

CHAPTER 3

Hypertext & Interactive Narrative

We now know that video game narrative can be approached from a narratological point of view. It would then make sense to familiarize ourselves with narratological theory and structuralism. This chapter will begin with a brief overview of both of these areas, followed by an introduction to hypertext, a summary of its modeling methods, and an exploration of its use in authoring systems for interactive narrative.

3.1 | Structuralism & Narratology

When discussing the term 'narratology', I am referring to literary theory used for systematic study of narrative. Narratology acts as an umbrella term and is not necessarily a single object of study, but instead an amalgam of methods coming together from related but not identical fields. It is in some ways a natural progression of the fields it is made of. One of the strongest influences on narratology is structuralism, which largely dominated in its earlier forms. Structuralism - the study and classification of elements of narrative and their relationships - is of particular interest to this thesis. This is due to its popularity in the modeling of hypertextual systems, and increasingly as a method of study in video games.

We can trace back structural approaches to literary theory to Aristotle's *Poetics* which laid out six elements of tragedy (plot, character, diction, thought, spectacle, and song) as well as the famous beginning, middle, and end structure. However, one of the first modern remnants of structuralism that we see today is the Aarne-Thompson-Index [1]. This index is a classification system intended to find similarities between folk tales by grouping variants of the same tale structure under the same index. This work has been recently continued by Uther [69], under which similar tales are now classified as ATU (Aarne-Thompson-Uther) categories. Propp then introduced a model that listed 31 functions of narratives and the way in which they were combined as well as eight common character archetypes as a result of analyzing more than 100 Russian folk tales [64]. His work differed from those before as the content itself was abstracted and the resulting functions and rules could be used to create or represent arbitrary stories rather than only classifying existing ones. Around this time, Russian formalists differentiated between what is known as the 'fabula' and the 'sjuzhet'. Put simply, the fabula is the chronological sequencing of the events that occur during the narrative, and the sjuzhet is the presentation of these events through many means, and perhaps not in the same order. In crime stories, for instance, the fabula would be the events that actually happened and in the order that they occurred temporally, whereas the sjuzhet would be the reader's discovered events in whatever order they were experienced or interpreted. These concepts were later adopted by French structuralists such as Barthes who later contributed the structural concept of distinguishing obligatory events for a story to be recognizable (kernels) and optional events that supplement the main plot (satellites) [7].

3.2 | Hypertext

The phrase *hypertext* was coined by Ted Nelson in 1965 [58]. The term 'hyper-' is denoting extension and generality, in this case, of text. Nelson defined this as *'a body of written or pictorial material interconnected in such a complex way that it could not conveniently be presented or represented on paper'*. It is also stated that such a system is able to grow indefinitely, gradually expanding. The term *hypertext* no longer conforms exclusively to this original definition, and now refers to most interconnected data regardless of its complexity. Additionally, the term *hypertext* was previously reserved for text-only systems, with the term *hypermedia* being employed for multimedia contents. Now it is not uncommon to see *hypertext* encompassing all kinds of media other than text. Hypertext would later become the driving principle behind the standard language and transfer protocol of the Internet, Hypertext Markup Language (HTML) [8] and Hypertext Transfer Protocol (HTTP) [28], of which both are still among the dominant methods used today. As hypertext was conceived in the early days of modern computing, it has developed alongside a rapidly-expanding technological landscape that still to this day is innovating. As a result of this, hypertext has and continues to become instrumental in many systems. An advantage of hypertextual systems over conventional forms of media consumption (text or otherwise) is the ability for a fluid structure of viewing that is (usually) decided by the reader. Hypertexts such as webpages can be experienced in countless ways based on the personal choices of the reader, providing them a level of control over the reading experience.

Hypertext Fiction

Hypertext fiction is an early form of electronic literature. Its media content (text, images, and so on) is defined in freestanding nodes which have some form of connectedness between them. As this system is hypertextual and both authored and experienced on a computer, restrictions imposed by printed medium are lifted and new structures can be explored. A reader may not necessarily experience a story the same way twice, as the order that nodes are visited need not be fixed, and can be reliant on reader choice. Additionally, hypertext fiction can choose to provide tension between itself and the reader by not explicitly choosing to follow their choices, whereas in printed medium this tension cannot exist at the same level.

One of the first known works of hypertext fiction is Michael Joyce's 'afternoon, a story'[1]. This was first presented to the public as a demonstration of the authoring system Storyspace[2] [11] at the 1987 Hypertext conference [14]. This seminal work was met with praise, partially due to its novelty. It has subsequently been a target for analysis, such as the narratological analysis by Walker [70].

The importance of hypertext fiction led to the creation of the Electronic Literature Organization[3] (ELO) in 1999. The ELO was founded with the intention to preserve and promote hypertext fiction (among other formats of electronic literature).

3.3 | Modeling Approaches

The term 'narrative modeling' in this context is referring to the systematic process of analyzing and breaking down narrative into representative and usually interlinked components. The level at which this process takes place varies. Some models may take a hierarchical approach by defining tree-like structures that embody low-level elements of narrative and their relationships to one another. Other approaches may be more abstract in determining common trends or patterns observed in the apparent structure that comes about as a result of narrative actions taking place. Regardless of approach, all of the models attempt to in some way abstract features of narrative to create a new understanding of how it is formed.

Structural Meta-Analysis

A common and effective method of modeling narrative is to analyze existing material in order to determine its underlying structure. The results of which can then be used to create a reference taxonomy or design guidelines.

One of the most prominent works of narrative structural analysis is Vladimir Propp's *Morphology of the Folktale* [64], better known as "Propp's Morphology". This work is one of narratology rather than hypertext analysis, but its techniques are resonant throughout structuralism and therefore is important to discuss. In this seminal work, Propp analyzed over 100 Russian folktales in order to determine common elements of plot structure, what he named *functions* of narrative, as well as

[1]'Afternoon, A Story' by Michael Joyce.
[2]Storyspace was released in 1987 and is now maintained by Eastgate Systems, Inc.
[3]Electronic Literature Organization. www.eliterature.org

identifying common character archetypes. Propp's initial findings resulted in a total of 31 functions and seven broad character archetypes. These functions could be then placed in sequential order using a representative letter assigned by Propp in their definition, and thus the plot of any given story could be expressed as a sequence of symbols. Propp suggested rules surrounding this model, such as functions being ordered, that each story does not have to contain all functions, and that functions can optionally be inverted. Propp's identification of these narrative elements and character archetypes have gone on to be used in narrative generation [31]. They have also been built upon for analysis of video game storytelling [16, 19], which will be discussed in a future chapter.

Mark Bernstein, in his Patterns of Hypertext [9], highlighted that by developing a richer vocabulary of hypertext structure through the analysis of existing hypertexts, we can build a more rigorous framework with which to criticize and inform new works. To achieve this, Bernstein analyzed a large number of websites and hypertext novels and, like Propp, identified repeating structures. In total, he identified 18 structural patterns of the various hypertexts that occurred throughout and highlighted that these patterns can also nest within one-another for increased complexity. Bernstein's analysis contributed a catalog that could be used by authors and scholars alike as a reference as to how certain structures had different effects on the hypertext.

A similar approach of analyzing a corpus to determine structure can be found in the Dexter Hypertext Reference Model [34] (Dexter). Dexter is a framework that not only captures hypertextual information and its connectedness, but also the methods of user interaction. A number of contemporary systems were analyzed to build the model, which was composed of three layers: The Run-Time layer determined user interactions with the hypertext, the Storage layer described a network of nodes and their connectedness, and the Within-Component layer contained the actual content of the hypertext. These three layers worked together to create a framework that could be used to compare systems as well as support the development of an interchange format between generic hypertext systems. However, in comparison to solutions from Propp and Bernstein, this model could not be used as a reference, as the framework produced was more technically-oriented and specified components at a lower-level. Additionally, in Dexter's literature, it is stated that 'hypertext' and 'hypermedia' are different terms, but that Dexter does not differentiate between the two and that 'hypertext' refers to both. It is possible that this was a key point in which the hard-line distinction began to diminish.

Computational Narrative

Computational narrative is an astoundingly large field on its own and is beyond the scope of this thesis. For the interested reader, there exists a fascinating survey of contemporary generation and author-aiding techniques [48]. However, gaining an understanding of some modeling approaches used within this discipline would broaden our view of narrative modeling methods.

Propp's Morphology has a notable history of use as a grammar in story generation. In Morphology of the Folktale, Propp explicitly states *"It is possible to artificially create new plots of an unlimited number. All of these plots will reflect the basic scheme, while they themselves may not resemble one another. In order to create a tale artificially, one may take any A, and then one of the possible B's then a C$\uparrow$, followed by absolutely any D, then an E, then one of the possible F's, then any G, and so on. In doing this, any elements may be dropped (except possibly for an A or α), or repeated three times, or repeated in various forms. If one then distributes functions according to the dramatis personae of the tale's supply or by following one's own taste, these schemes come alive and become tales. Of course, one must also keep motivations, connections, and other auxiliary elements in mind"* [64, pp. 111-112].

The oldest known implementation of complete story generation is a fairy tale generator by Joseph E. Grimes developed in the early 1960s. This project was largely lost and only recently rediscovered [65]. In this early system, Grimes used a grammar-based approach centered on Propp's Morphology which output new stories in natural language in both English and Spanish. By using the rules and constraints laid out by Propp, Grimes was able to print out simple yet complete randomized stories, similar to how Propp had envisioned prior to modern computing.

Propp's morphology was also used in the Proppian Fairy Tale Markup Language (PftML) [53] which proposed an XML schema representing the functions and their rules. The author does conclude a number of features were missing or difficult to capture using XML. While this project was not particularly interested in generation, the representation of Propp's framework in a computer-friendly format is invaluable.

Further substantial work has been done using Propp's Morphology as a grammar for generation [31]. This work differs from the previous in that it implemented the rules and functions in significantly more detail. Functions, for instance, which are defined broadly, were implemented as narrative actions with associated preconditions and post-conditions. Additionally, this approach maintained the context of the generated story by tracking a set of states before and after actions took place. The resulting implementation, prototyped in Java, was able to generate Russian fairy tales, but the authors highlight flexibility in being able to be ported to other areas with modification as well as adapting it to grammars other than Propp's Morphology, such as Campbell's Hero's Journey [20] and Lakoff's Structure of Fairy Tales [49].

Grammar-based generation has also been used to aid the authoring process, such as in Tracery [23]. Instead of relying on an existing grammar as described above, Tracery uses a custom grammar. This grammar is exposed to authors in a Web-based interface where they are able to specify and configure numerous rules, constraints, and keywords. This is then used to generate a narrative by evaluating the input and satisfying the schema of the computational grammar.

A fine example of generative algorithms being used for narrative synthesis in the context of aiding authors is Disney's CANVAS project [42]. The primary goal of this project was to provide an accessible yet expressive interface to author stories facilitating rapid prototyping, iteration, and deployment of concepts, which is critical in the animation industry. Authors of this system are able to specify key plot points in a story, and the system computes and generates the missing details within the narrative (i.e., what happens during the key plot points) and creates a 3D animation for rapid turnaround. In their ACM course *Computational Narrative* [43], they further reflect on the importance of authoring stories using computer-assisted generative algorithms. Previous works by Disney have likewise focused on reducing the authoring burden by integrating narrative synthesis into various authoring systems [41, 44, 63]. All of these authoring-aided systems aim to simplify the authoring experience by having an accessible, high-level definition for authors to work with while maintaining a rigorous and low-level description at the model level.

In summary, there are two clear areas of interest in computational narrative to this thesis. The first is of use of grammars as a modeling approach, often through adaptation of existing high-level models into a computational form. The advantage of these is that they are represented, even at the model level, in human-readable and relatively understandable forms. The second area is computational engines that aid the authoring experience of more complete, complex narrative experiences. These models are often lower level and have to provide abstraction to the authoring process, but as a result are able to handle much more complex and complete narratives.

3.4 | Authoring Structure

When it comes to the authoring of hypertexts, we must consider not only the content itself but also how we go about linking it together. This linkage between individuals forms the basis of structure. Therefore, understanding the methods available for creating such links is advantageous.

Structure is typically created in what is referred to as a *calligraphic* way. Authors take a set of individual, unconnected nodes of data and link them together to form the structure that they desire. This linking may be done in a variety of ways such as visually in an editor or explicitly in code as is often done when authoring hyperlinks between documents in HTML. Regardless of the method used to create the links, it is a manual process starting from a blank state of pure disconnection that gradually becomes more connected as authors see fit.

There is an alternative approach termed *sculptural* that inverts the method of authoring linkage. In this form, all sets of nodes are connected equally to one another, and the author, rather than defining that is linked to, specifies what *isn't* linked to by either removing connections or placing constraints that in turn restrict connectivity. This is referred to as *sculptural* in the sense that sculptors create objects by gradually removing unwanted material. This technique differs from calligraphic hypertexts as the authors gradually reduce connectivity rather than explicitly stating it.

Both of these terms were coined by Bernstein in his works on Card Shark [10]. In Card Shark, authors write the content of individual 'cards' which contain some text, and can optionally declare constraints on the connectivity and availability of the card (e.g., using `BEFORE 25` to state that the card can only be visited early in the reading; if not experienced before this condition, it never will be). A set of rules determines the order cards become available to the reader. Some choice is presented where multiple card constraints are met (and of course taking different cards could result in a different reading experience each time).

3.5 | Interactive Narrative Authoring Systems

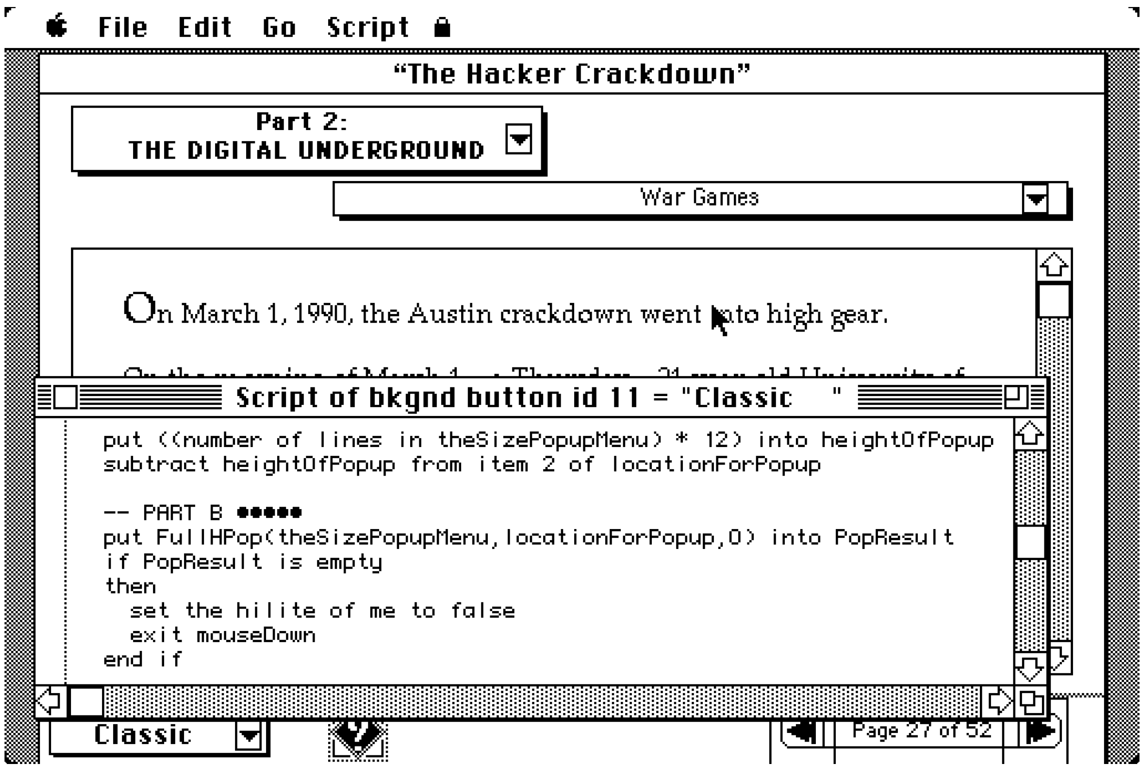
The concept of hypertext has been used extensively within tools for the creation of interactive narrative. This is due to the underlying models or the output generated being of a hypertextual nature. The actual interface and UX of such systems still varies greatly. This section will cover a selection of features found in a number of authoring systems and discuss similarities between their implementation.

Scripting

Support for scripting can be found in a wide array of authoring tools with a varying number of applications. By 'scripting', I am referring to some form of compiled or interpreted syntax integrated into either the authoring system or final output (where applicable). Including the ability to write scripts is important for authors as it empowers them to increase the complexity of their interactive narratives. Narratives are able to, therefore, become more dynamic in that elements can be responsive and link traversal can be manipulated conditionally in runtime. Care must be taken when including scripting, though, as not all users will be comfortable with the concept.

Sometimes Domain Specific Languages (DSL) are purpose-built for use in authoring systems, or alternatively are built on top of existing languages. Apple's *HyperCard*[4] [67] is a good example of this. *HyperCard* was a generic hypertext authoring system for the Macintosh. Authors created 'stacks' which hosted controls and media items. Scripting was implemented into *HyperCard* with their own DSL named HyperTalk. HyperTalk was a high-level scripting language that broadly represented plain English. It would then go on to inspire a number of languages. In *HyperCard*, HyperTalk could be used to add interactivity and complexity to the author's stacks. This could be as simple as making a button link to another stack (à la hyperlinks) or as complex as calculating 3D surfaces for rendering. An example of editing a HyperTalk script can be seen in Figure 3.1. A similar hypertext

[4]HyperCard. Apple Computer, Inc. 1987.



On March 1, 1990, the Austin crackdown went into high gear.

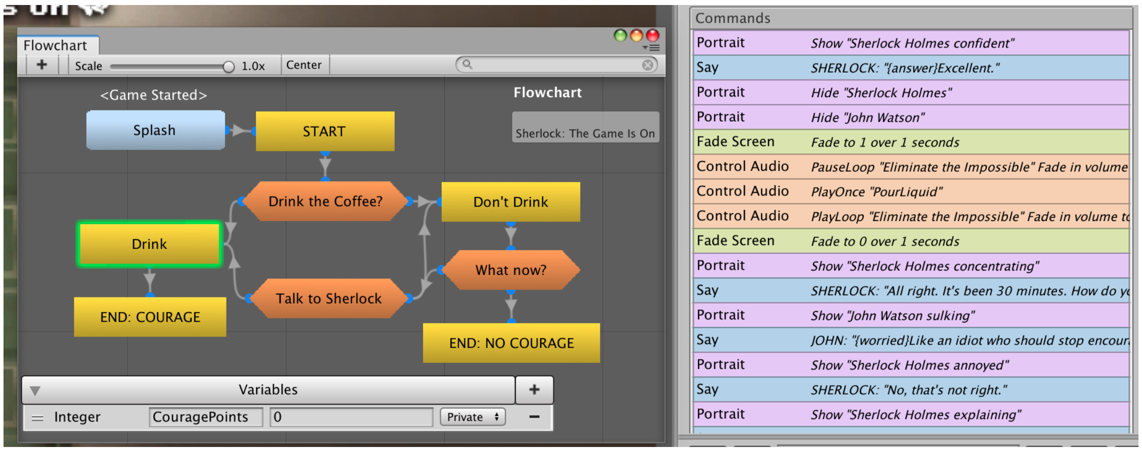
```
put ((number of lines in theSizePopupMenu) * 12) into heightOfPopup
subtract heightOfPopup from item 2 of locationForPopup

-- PART B ●●●●●
put FullHPop(theSizePopupMenu,locationForPopup,0) into PopResult
if PopResult is empty
then
  set the hilite of me to false
  exit mouseDown
end if
```

Fig. 3.1 Editing a HyperTalk script in the *HyperCard* stack "The Hacker Crackdown".

authoring system released a few years later, *SuperCard*[5], mimicked *HyperCard's* behavior and included its own DSL named SuperTalk, which was heavily influenced by HyperTalk. HyperTalk's influence can also be found in *Adobe Director's*[6] DSL named Lingo, which implemented a similar natural language approach to otherwise complex programming.

The *Fungus*[7] plugin for *Unity*[8] implements its own version of LUA[9] named FungusLUA[10]. This extension provides a simple language to authors who are comfortable with programming. However, *Fungus* also lets authors use a visual scripting system implemented as a flowchart combined with a list of predefined 'commands' for those who are not familiar with programming. This can be seen in Figure 3.2 with a visual chart on the left and a list of simple commands on the right. This method of simplification can also be found in the *Quest*[11] narrative authoring system. When adding scripts in *Quest*, the author has the choice of either writing actual code or selecting and combining behaviors and actions from a list, which can then be edited using simple controls (text boxes, drop-down menus, and so on). A unique feature with *Quest* is that it can toggle between the script and simplified interface rather than being a binary choice. *inklewriter*[12] takes this a step further by not allowing any

[5]SuperCard. Bill Appleton. 1989. (Latest version by Solutions Etcetera, 2012.)
[6]Adobe bought Director in 2008. It was originally released in 1985 as 'VideoWorks'.
[7]Fungus. Snozbot. www.fungusgames.com
[8]Unity. Unity Technologies. www.unity3d.com
[9]LUA is a cross-platform scripting language.
[10]FungusLUA is based on MoonSharp, which is an open-source interpreter compatible with LUA.
[11]Quest. Community. www.textadventures.co.uk
[12]inklewriter by Inkle Ltd. Due to close August 2018. writer.inklestudios.com



Commands	
Portrait	*Show "Sherlock Holmes confident"*
Say	*SHERLOCK: "{answer}Excellent."*
Portrait	*Hide "Sherlock Holmes"*
Portrait	*Hide "John Watson"*
Fade Screen	*Fade to 1 over 1 seconds*
Control Audio	*PauseLoop "Eliminate the Impossible" Fade in volume*
Control Audio	*PlayOnce "PourLiquid"*
Control Audio	*PlayLoop "Eliminate the Impossible" Fade in volume t*
Fade Screen	*Fade to 0 over 1 seconds*
Portrait	*Show "Sherlock Holmes concentrating"*
Say	*SHERLOCK: "All right. It's been 30 minutes. How do y*
Portrait	*Show "John Watson sulking"*
Say	*JOHN: "{worried}Like an idiot who should stop encour*
Portrait	*Show "Sherlock Holmes annoyed"*
Say	*SHERLOCK: "No, that's not right."*
Portrait	*Show "Sherlock Holmes explaining"*

Fig. 3.2 A sample visual scripting flowchart and command list in *Fungus*.

code at all. Instead, authors handle conditions using 'markers' which are essentially named Boolean flags. Authors are able to enable or disable links by declaring which markers should or should not have been visited.

Other authoring systems take an approach that relies heavily on knowledge of scripting to effectively use the application. *Twine*[13], a web-based interactive fiction authoring tool, has scripting as an integral part of its design. Content is edited in a node graph, but links are defined by the contents of the nodes. For example, if we have two nodes, `A` and `B`, then `A` can be linked to `B` by setting its content to `[[B]]`. The actual language used is quite complex, and since connectedness and advanced behavior relies on scripting knowledge, it can be difficult for non-programmers to use. *Twine* supports (at least) three 'story formats' which are essentially different syntaxes for writing code within the nodes. Sometimes authoring systems rely almost entirely on code, as with *Inform*[14], but this does not inherently limit accessibility. *Inform's* DSL is written largely as natural English, which reads almost as writing a book, significantly lowering the barrier for entry regardless of its heavy reliance on scripting. For example, *"The House contains a chest. In the chest is a key."* would define that inside the `House` is an object named `chest` that contains a `key` that the player would be able to interact with. DSLs are not always used. The popular *Ren'Py*[15] authoring system uses its own language which derives from Python. This lets those who know the language use the flexibility and power of Python, but it also provides a simpler syntax that can be easily learned for those who do not want to explore too deep.

Content Creation

Content creation is the primary means by which authors craft, manipulate, and edit content within an authoring tool. This can come in a variety of forms, each with different advantages and disadvantages. It is not uncommon for an authoring tool

[13]Twine. Community. www.twinery.org
[14]Inform 7. Community. www.inform7.com
[15]Ren'Py. Community. www.renpy.org

to implement more than one method of creation. Ensuring an authoring tool has sufficient and suitable methods of content management can make or break usability.

A common method for content creation and management is the use of connected node graphs. This paradigm represents individual pieces of content as nodes with connections represented as lines between the nodes. A clear advantage of this method is that the structure of connectivity is displayed directly to the authors. Typically, node editors come in two flavors: static and dynamic. Static node editors are for pure visualization of structure. Dynamic node editors allow for actual creation and manipulation of content from within the graph itself.

articy:draft[16] is a game-centric authoring system with a focus on the creation of non-linear stories and branching dialogue sequences. While it has a large array of editing methods, the primary mode of creating and managing content (particularly dialogue) is done through a rich-featured dynamic node graph. Figure 3.3 shows a sample of this graph. Each of the nodes contains involved characters, event text, and a set of inputs and outputs.

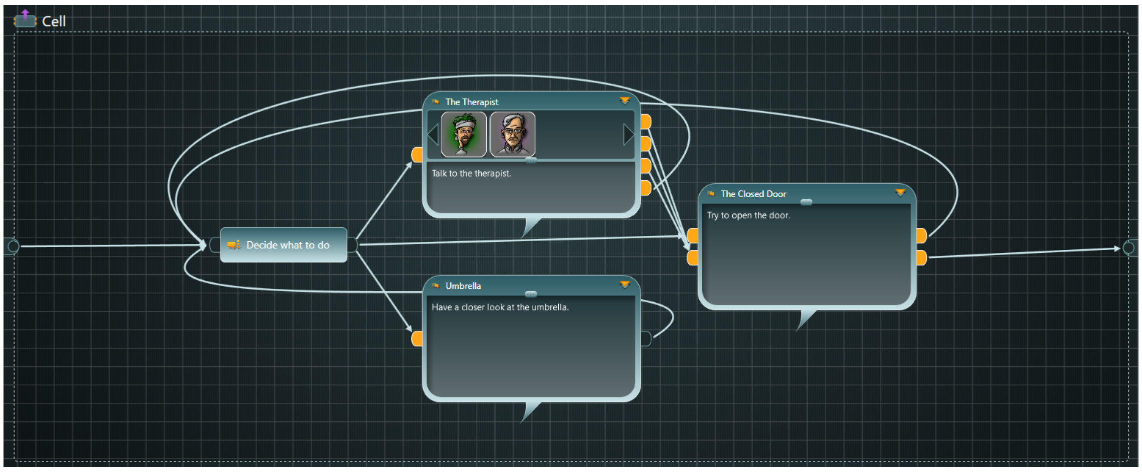


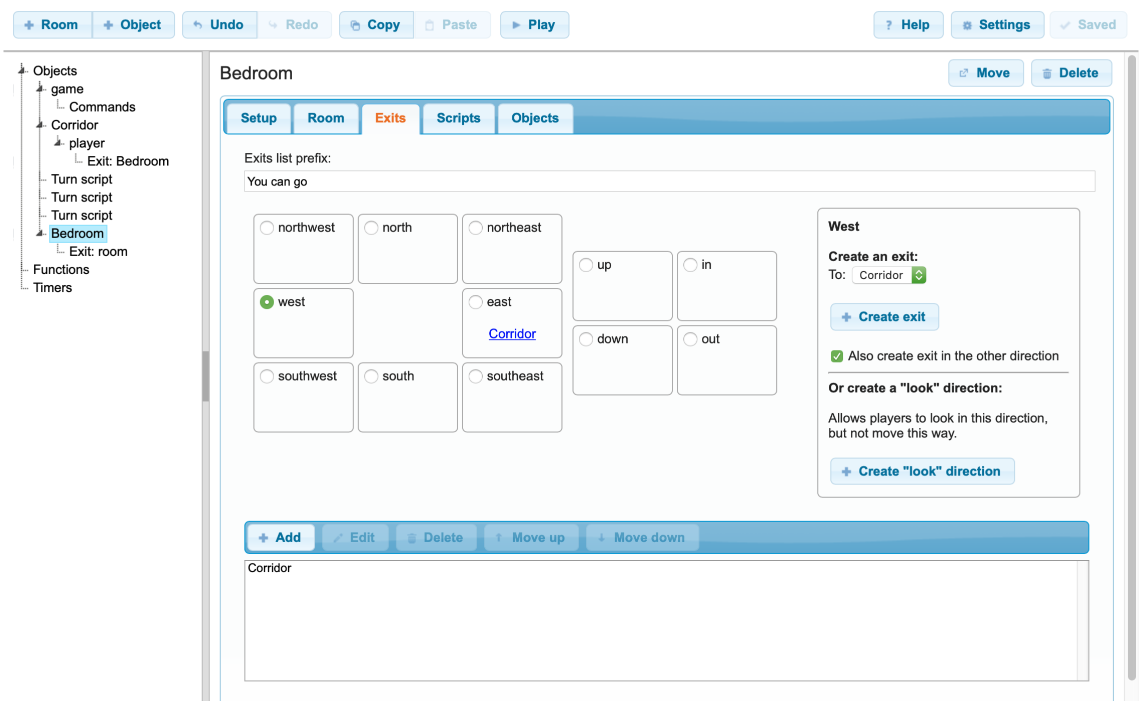
Fig. 3.3 A connected node graph sample in *articy:draft*.

Similar behavior can be found in *Twine*, as mentioned earlier. The primary method of arranging content is through a dynamic node graph. However, unlike in *articy:draft* which edits content inline on the nodes themselves, content is edited by selecting a node and opening a modal dialog that serves as a text-based editor for that particular node's content. An advantage of this is that nodes can be much smaller, but a disadvantage is that modal dialogs are used which can notoriously break workflow. An example of a static node graph can be found in *inklewriter*. The way in which content is edited in *inklewriter* does not clearly show structure as the previous examples do. The 'Map' feature provides a static node view of the scene in a more traditional node-line style. However, these nodes are purely for visualization only and do not serve as a means to edit the content.

An interface paradigm found in almost all authoring applications is multifaceted controls. This term refers to a collection of arranged atomic controls (or derivatives

[16]articy:draft 3. articy Software GmbH & Co. KG. www.nevigo.com

of) such as buttons, text boxes, check boxes, and so on. An example of multifaceted controls is the WinForms[17] library. A practical example of this can be found in the earlier-mentioned *Quest* authoring system. A majority of this program's interface uses multifaceted control design to let the player edit the game world. Figure 3.4 demonstrates editing of room exits using text boxes, push buttons, radio buttons, drop-down boxes, and more. If designed well, these interfaces can be quite simple to understand. However, due to the sheer amount of controls used, editing must often be broken up into multiple sections or pages, as has been done in *Quest*. This means the author must navigate around quite substantially in order to edit their story.



Exits list prefix:
You can go

West
Create an exit:
To: Corridor
✓ Also create exit in the other direction
Or create a "look" direction:
Allows players to look in this direction, but not move this way.

Corridor

Fig. 3.4 Room exit editing with multifaceted controls in *Quest*.

Perhaps the simplest form of content editing is through the use of augmented text editors. These text editors usually have some form of additional features such as autocomplete, syntax highlighting, or other programmer-friendly support.

Inform is an ideal example of this, as its primary method of content creation is through a text editor. In the case of the *Inform* application[18], the main text editor has syntax highlighting for different kinds of keywords. Similar behavior can be found in the text-based editor of *Squiffy*[19]. *TextureWriter*[20] likewise focuses primarily on a text-based editing experience but features less in the way of syntax highlighting. Instead, it highlights interaction phrases by underlining them (or in some circumstances, changes their background color).

[17]WinForms is a graphical user interface library for Microsoft's .NET framework.
[18]I am referring to the editing environment available at www.inform7.com.
[19]Squiffy. Community. www.textadventures.co.uk
[20]TextureWriter. Leinonen, J., Munroe, J. www.texturewriter.com

Integration

Different authoring tools integrate into existing runtimes at different levels. Sometimes the programs are purely standalone and only produce output for viewing or view internally. Other times authoring tools can be integrated directly into a runtime for close coordination between the two systems. By allowing authoring tools to integrate (or export to) existing runtimes, they become more flexible in the kinds of authors they can serve. However, care must be taken when supporting multiple platforms that differ, as those differences may result in some platforms being less supported than others, or in a worst case, the authoring tool compromising because of the limitations placed on it by the platforms.

articy:draft is able to export to a wide array of formats. Most notably is the ability direct support of *Unity*, which means that authors can create their narratives using the standalone software and see it brought to life within the game engine. Additionally, an API for writing custom exports is provided, which delegates the job of data conversion to the team using the product (although for individual non-technical authors, this may not be of much use). The *Inform* application, while more limited in its exporting functionality, supports publishing to a `Blorb` format which can be played by any compliant interpreter. *Inform* also incorporates its own runtime which can be used to preview and test the author's story without exporting to an external runtime. A vast percentage of web-based authoring tools are able to export to formats playable in the browser. This can be seen in *Twine*, *inklewriter*, `Quest`, and *Squiffy*, just to name a few. This method is often referred to as 'publishing' a story which usually makes it public for consumption.

A more direct approach is integrating the actual authoring tool into a runtime itself. By doing this, the tool can facilitate itself directly within the features of the runtime which may not be otherwise possible through exporting alone. Additionally, if integrated directly into the runtime, it is common for the visuals of the authoring tool to be reminiscent of the runtime itself, which can make the tool feel familiar if the author is already used to the runtime. An obvious downside is that the tighter integrated into a software an authoring tool is, the less generic it can become, and therefore supports fewer forms of output. Depending on the use-case, this may be acceptable or even desirable.

Fungus is a great example of this, as it is an authoring tool written entirely into *Unity*. It reuses a number of components from *Unity* itself which means that authors don't have to spend as much time learning as they may already be comfortable with the program. A number of Disney research projects [62, 63, 74] have taken a similar approach by integrating their authoring system directly into *Unity* for similar reasons.

Conclusion

While there are countless features that differentiate authoring tools, this section highlighted and demonstrated a number of broad popular techniques used and showed examples of implementation and how they differ between set programs. A deeper investigation into the common features that authoring tools have and how they differ is an interesting pathway for analysis and will be explored in future works.

This page is intentionally left blank.

CHAPTER 4

Modeling Video Game Narrative

Hypertext has been modeled countless times since its inception, and as it has developed sub-disciplines or engulfed others, they too have largely been modeled. By extension, interactive narrative, being a combination of narratology and hypertext, has also had many systems attempt to capture narrative elements at varying levels of detail. Video game narrative, on the other hand, which can be considered to contain forms of interactive narrative, has not been modeled as much or as well. A number of things could contribute as to why. The field as a whole is somewhat new, and the hypertextual approach to analysis and development of video games has not always been of the most importance. Additionally, video game narrative presents difficult challenges, as discussed in Chapter 2. We have already looked at modeling approaches of hypertext and interactive narrative in Chapter 3. However, we must expand this by looking at if and how these models apply to video games and identify which elements of video game narrative they struggle to capture.

In this chapter, I will explore a number of approaches to creating game-specific models by both adapting existing non-game centric models and through the creation of new ones. This is then followed by an initial exploration into the ability for both game-centric and non-game centric models to represent video game narrative. The chapter then concludes with my first major contribution, *Discoverable Narrative*, developed and informed by the result of the analysis, accompanied by the first version of my theoretical narrative model.

4.1 | Video Game Narrative

As has been explained earlier in this thesis, video game narrative has complexities that result in a number of nuances that can be difficult to capture. Attempts to capture such features have been done by creating game-specific models. By having these models target video game narrative in particular, they are able to capture elements that would otherwise be missed by broader hypertextual theories.

Adding Interactivity to Existing Theories

One common approach is to take existing theories that are not based on games and attempt to add support for interactivity into them, or to otherwise modify them to better suit games analysis.

One such attempt is to inject interactivity into Campbell's Hero's Journey [24]. The original theory by Campbell [20] specifies 17 phases and is ideal for mapping linear narratives that conform to the progression outlined by the model. The authors highlight the lack of support for interactivity due to having no way to handle player choice and actions undertaken, which are fundamental to most gaming experiences. The authors suggest firstly keeping the player at the center of the action, and that resolution must be a result of player choice. Second, that when a choice is given to a player (begin journey, accept/refuse help, etc.) that each possible option must be valid and allow for progression of the narrative. They also note that a lot of games cause failure or backtracking by picking the 'wrong' choice, and suggest instead sanctions to be placed on players instead of halting. Third, that some optional phases must be conditional based on player ability (i.e., the phases activate based on the player's prior ability to succeed). In their final model, a number of additional phases are added particularly to choice and agency, and phases can link to any neighboring phase, including repetition of already passed phases. As the player encounters phases that could have more than one outcome, the pathway taken is now dependent upon the player's choice. Sequences ahead of the player's progress can also dynamically adapt to the success and ability of the player. In the *Road of Trials*, for example, a setup could be made so that the player can only exit the sequence once adequate performance was reached, otherwise subjecting the player to repeated testing.

A similar approach was taken in the Narratification project [68], which attempted to unite narrative and gameplay in an analytical framework. The model upon which this work is based is Dixon's Goal, Motivation, and Conflict (GMC) model [25]. Dixon's model defines three categories for characters, each having internal (known to themselves) and external (public knowledge) factors. *Goal* refers to the objective the character would like to achieve, *Motivation* defines why they want to achieve their objectives, and *Conflict* determines what is stopping them from reaching their objectives. The authors highlight that this model is ideal for developing meaningful plots based on a character's internal and external factors, but that it does not capture any dimension of interaction from the player. They then extended this model into what they call Interactive Goal, Motivation, and Conflict (IGMC) by adding an additional column for the player alongside the character's column. The player and character columns are interconnected via the game experience. The GMC entries of the character must satisfy the entries of the player and vice versa. The authors note that this presented solution is aimed at designers being able to consciously plan out the relationship between player and character GMCs.

Propp's Morphology has also been indirectly applied to video game narrative. In Brusentsev's work [19], Propp's Morphology is the basis of a game analysis framework in an effort to discover how well the original form of the theory applies to video game narrative. In this framework, a number of key sections (some of which use Propp's functions and character archetypes) are used to deconstruct the game. The *Game Structure Overview* provides a brief description of the game, highlighting unique narrative techniques. *Character Archetypes* lists all characters present in the game and assigns them to one of Propp's character archetypes. *Overall Story Arc* assigns Proppian functions to the overall story to allow for a high-level description of the structure. *Level Narrative* then breaks down the story into acts (or similar) and maps the functions again. *Impact of Player Decisions* assesses how player choice alters the narrative. *Dialogue* looks at how the in-game player dialogue choices move the narrative. Finally, *Morality* sees if ethical choices in the game alter the narrative. These elements are all considered for any given game. The authors conclude that by using Propp's functions, they can gain oversight of the structure of video game narrative, but receive no further insight into it. They also note that the functions, in their original form is unable to handle any form of choice or agency, particularly from the player. One solution that they suggest without breaking Propp's original rules is a 'Decision Function', but this is purely speculative and was not further developed. This concept was further tackled by Bostan [16]. This project took a different approach by modifying and appending Propp's functions and their associated rule set to better suit games. Six of the original functions were modified and 15 new game-specific functions were added. In this framework, the story is broken down into any format the author sees fit (such as acts, chapters, levels) and then the extended list of Proppian functions are mapped in any order, can repeat any number of times, and can have branching simply by specifying connections between the functions. Although this approach is less rigorous than Propp's constraints, it does have the advantage of letting us analyze repeating patterns within game narrative. For instance, if there is a sequence of three functions repeated at numerous points that indicate a climax and battle, we could partly infer the pacing of the narrative from these repetitions, as well as begin to balance out the pacing by adjusting the spacing between the patterns.

Componentization and Factorization

Other approaches include the concept of componentization and factorization. These phrases are derived from the computing terms meaning to take a given system and reduce it to a number of logical components that work together. By breaking down video games into their constituents, we can generally develop a better understanding of their structure and look at how each component contributes to the narrative.

A common approach taken is to develop a taxonomy of game elements which can then be used for reference and analysis. An example of this is the interaction-centric structural framework by Bjork [13], further refined in his later contributions [50]. This work represented all objects as components and defined their involvement in events as one of three types of actions: those instigated by the player, those of components with prescribed agency, and those originating from the game system itself. Another example is the Game Ontology Project [73] which similarly relies on a set of entity manipulation actions to declare events within the game. These high-level concepts can be used to broadly represent connected events within a narrative. For example, it is possible to take logs of interactions between these components and begin to build a picture of the experienced narrative after such events have taken place [22]. This concept has also been taken further to determine the effect of these kinds of components on the narrative [17] by surveying 80 games and identifying relationships that emerge between components and various narrative forms.

A similar approach was taken by Bizzocchi [12], but with a particular focus on narrative rather than general structure. In this work, he identifies a number of factors of narrative in video games and how players interface with them. While this work defines its factors at an observational level, it can serve as initial groundwork for further structured analysis. A more component-oriented approach can be seen in Aarseth's framework [2]. In this framework, Aarseth breaks up games into the game world, objects, agents, and events. The world represents the physical structure of the game's level layout and differentiates between ludic and extra-ludic spaces in a way reminiscent of Adams' structures discussed earlier. The objects are entities within the game world, categorized by their malleability and interactivity. Agents represent characters and are likewise classified by their malleability but can be further divided into Bots, Shallow characters, or Deep characters. Events represent individual moments of narrative within the game and are considered either Satellites, which are supplementary events, or Kernels, which define particular stories. This is extended to say that all kinds of narrative can be defined as a combination of these Satellites and Kernels, and that the type of narrative (linear, branching, etc.) can be determined from the ratio between the two.

4.2 | Video Game Analysis

In order to better understand the affordances and constraints of existing models within the context of video game narrative, I applied four models to two contemporary games. The games chosen were *The Stanley Parable*, which features heavily complex and interwoven narrative, and *Portal*[1], which is comparatively linear. These two games represent two key challenges in the space of modeling video game narrative: a range of player agency and a range of presentational techniques. Prior to the application and analysis of the models, I played through each game multiple times, keeping detailed logs of each scene, interactions, narrative entities, and potential

[1]Portal. Valve Corporation. 2007.

divergence. This, combined with various online resources such as walkthroughs, secrets guides, and wiki pages, resulted in a detailed set of annotations.

After conducting the analysis of all models on both games, I formulated a list of advantages and disadvantages for each model and concluded with a summary on the suitability of this modeling approach. This listing is not intended to be complete, but instead a list of observations of the strengths and weaknesses that stand out of each modeling approach used. The details of the analysis can be found in Appendix A.

Propp's Morphology

Propp's Morphology was included not in its original form. The analysis was done using the accompanying breakdown methods and character archetypes as described by Brusentsev [19]. The stories were then mapped out using the modified function list and rule set from the work of Bostan [16].

Portal was divided into two acts: the former comprised of 11 chapters and the latter containing three chapters. Figure 4.1 shows each chapter's function listing in order of occurrence, which provides a suitable overview of the narrative arc of the game. Using this mapping, I was able to identify in act two that both the first and second chapters ended with a sequence of functions 5/6/32/43 (Delivery/Trickery/-Confrontation/Escape), which matches the plot well. However, there was a clear lack of support for certain abstract and transitioning characters. The Party Escort Bot and Turrets act as the Villain's henchmen, for instance, but do not have a suitable archetype. Similarly, the Personality Cores, which are personified robotic entities with unique traits, have no applicable archetype. GLaDOS acts as a Helper to the Hero for most of the first act, but transitions to a Villain for the remainder of the game, which is not possible to represent using the standard Propp theory.

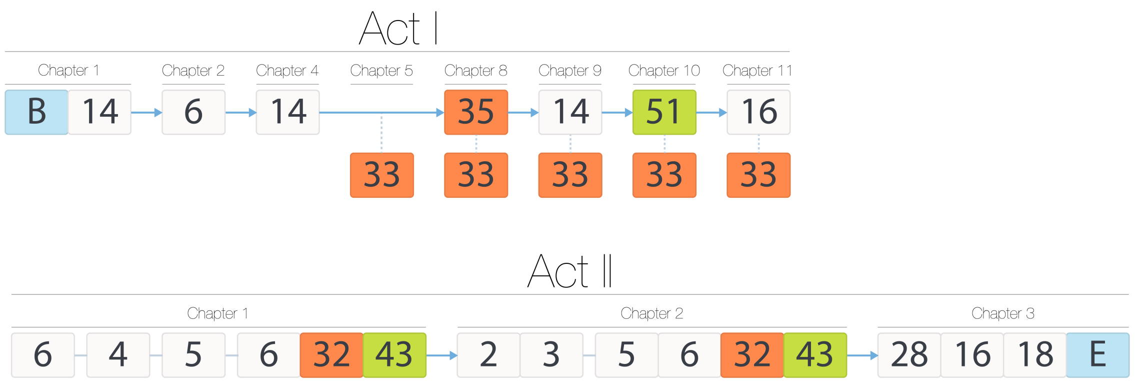
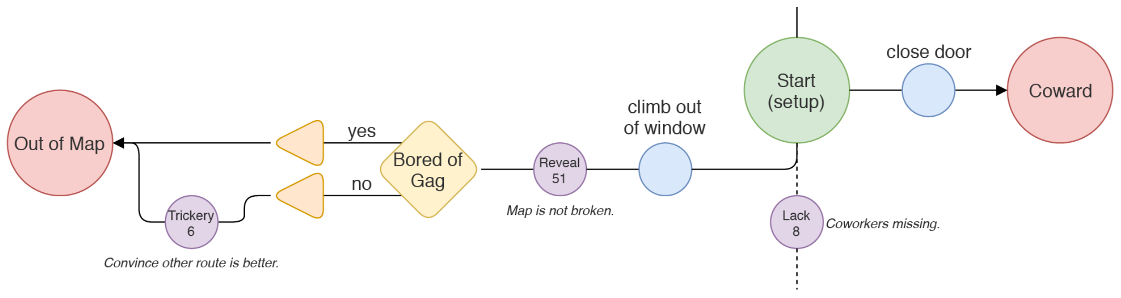
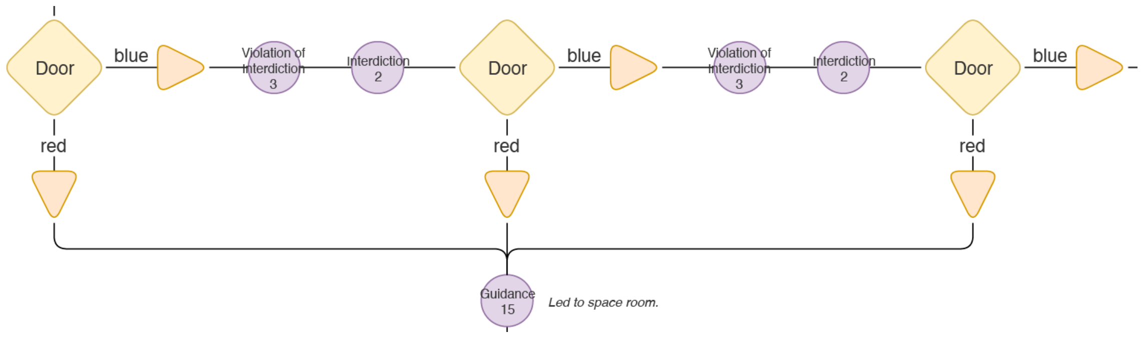


Fig. 4.1 *Portal's* structure with modified Proppian functions.

As *The Stanley Parable* has no concept of chapters and is focused on quick replays, it was instead graphed as a flowchart emanating from a single origin. From the origin, the path can be followed, including branches, to a leaf node, which represents every possible ending in the game. Along the path is a sequential listing of Proppian functions. Therefore, tracing any path through the branching narrative from start to end can result in an overview of that particular pathway's arc, just as with *Portal*. In Figure 4.2a, starting at the green *Start (setup)* node and following a path to the red *Out of Map* ending node shows us that this short pathway consists of functions 51 and 6. We can also see that function 6 is only experienced if the player chooses *no* to the *Bored of Gag* branch. Figure 4.2b, which is entered from the top-left *Door* branch, presents the player with three repeating choices. At any point, the player can break the loop by choosing the *red* option. Using this visualization, we can see not only the Proppian functions that make up the pathway, but also that no matter when the player breaks the cycle, that they are redirected to the same outcome. This form of analysis is useful not only for generating Proppian overviews but also for structural analysis. However, as with *Portal*, this model struggled to capture some of the more complex narrative scenarios such as the Broom Closet sequence, which has the game reset multiple times as part of its narrative. Additionally, some characters were not applicable to the archetypes, such as the player themselves being directly referenced, as well as others in the room with them, and Stanley's third-person self in the Not Stanley ending, to name a few.



(a) A simple path from the start to two possible endings.



(b) Visualization that three exit points lead to the same outcome.

Fig. 4.2 Subsections of *The Stanley Parable's* map.

Using Propp's Functions to map out a story provides a great overview of the narrative arc. Using a less restrictive set of rules for functions lets us detect sequences of patterns that appear frequently in the narrative but cannot represent any greater level of detail. Propp's Functions are generally simplified from their original Russian form, and some meaning is often lost in translation [64]. Their application to scenarios other than folktales becomes somewhat interpretative. The function set has been extended for game-specific scenarios, but still requires significant refinement to capture game narrative with clarity.

Aarseth

I used the breakdown methods for each category as described by Aarseth [2]. When the elements were all combined, it was possible to derive the *type* of narrative experience the game provided. However, there was no ability to discern anything more granular than that.

Portal is a *Linear Corridor* game with all but *Inventible* objects containing all character types, being plotted mostly by *Kernels* narrative with a few *Dynamic Satellites*. Similarly, *The Stanley Parable* is a *Linear Corridor* game but only has *Static* objects and does not have *Bot* character types. Its story is in the middle with both *Dynamic Satellites* and *Dynamic Kernels*. With this model, we cannot derive much more information than this. Additionally, character and object types are perhaps too vague and interpretative making assignment difficult at times. For instance, *Portal's* Personality Spheres are named with unique traits but are not deep. It is likely that more options would have to be added and existing ones refined or clarified in order to be of proper use. The kernels and satellites configuration can be used to determine the level of agency that a game provides, but do not describe what happens in the events themselves. The content of such events could come from a more detail-oriented approach.

CANVAS

The formal elements of the underlying narrative representation of the CANVAS [42] project were used. Both games were broken down into their constituents and modeled with interactivity in mind, but it quickly became apparent that this model did not support agency or choice in any form. Alternatively, the analysis was repeated using a single play session (i.e., choices were predefined, and interaction was removed), to which the model successfully represented the narrative, albeit in a linear form of a single instance of play. In applying this model, I found that it provides a level of detail allowing for the description of almost any static narrative. Unfortunately, the narrative must be predetermined and cannot include any form of branching due to player agency. Inserting interactivity into the model would help this. Additionally, due to the low-level nature of the methodology, it is a tedious process to build up a database of objects and their attributes, although since this technology is underlying an authoring system, it is expected that the process will be abstracted to the user.

Patterns of Hypertext

Bernstein's Patterns of Hypertext [9] was used to identify at least one example of occurrences of the patterns in each game where applicable.

Portal had few of the patterns, perhaps due to its narrative linearity. An example of a standard Cycle and Joyce's Cycle in *Portal* is the ninth test chamber. Near the beginning of the game, the player completes this challenge having access to only one of the two portal colors and must solve the puzzle within their limitations. During the latter portion of the game, the player encounters the test chamber again during their rogue escape attempt, but this time enters in a different way and has both portal colors available, making for a different solution. In terms of Bernstein's patterns, the player is experiencing a previously encountered node of which the perception changes due to the gathering of new information since the last time. Many instances of Missing Link can also be found throughout. There are many frosted glass observation rooms reminiscent of offices scattered around the test chambers, and many locked doors that the player cannot enter. The existence of these purposefully placed elements suggests to the player that there is a world beyond that which they inhabit and that it is seemingly simple to enter if they such desire. However, the game limits them from actually entering by making the areas inaccessible through regular play. This creates a fallacy of being able to explore deeper into the facility and therefore exhibits Missing Link.

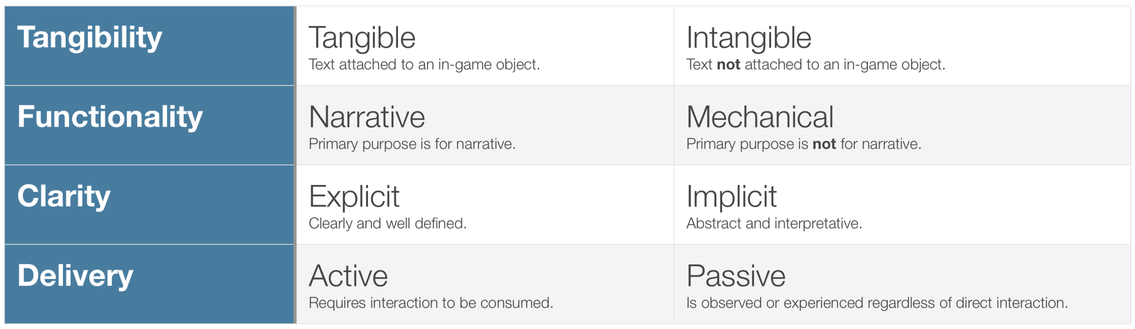
The Stanley Parable, on the other hand, contains examples of almost every pattern specified by Bernstein, likely due to its narrative complexity. The primary gameplay mechanic of player choice altering the narrative results in Cycles and Joyce's Cycles being an intrinsic part of the design. An example of such is the so-called Heaven ending. To achieve this, the player must find a specific in-game computer and activate it by interacting with it. The player must then manually restart the game, traversing a similar path, activating a different computer. Once this process has been repeated four times, the Cycle is exited, and the player transported to the appropriate ending. Another example of both a Cycle and Moulthrop's Move can be found during the red/blue door sequence toward the Zending, Divine Art, and Games endings. In this scenario, the player is presented with a red and blue door and is instructed to go through the red door. If they defy the order and enter the blue door, they are teleported back to the beginning of the sequence and requested to enter the red door. This process repeats three times, with the red door standing out more each time. The Cycle is then broken when the player either goes through the correct door at any point or defies the command three times. The way in which the Cycle is exited alters the narrative that follows.

Identifying and presenting structural patterns using this framework enables authors to be more aware and understanding of how their narrative experiences are constructed. Visualizing a Tangle's web of options could make it more maintainable, for example. When applied on its own to narrative, this model, by design, focuses on how elements relate to each other in a structural sense. Individual details themselves are not defined, and as such we cannot derive what actually happened, but instead the relationship to the surrounding narrative. This is to be expected given that the work is an observation of structural patterns rather than being concerned with the content of given narrative elements.

4.3 | Discoverable Narrative

All of the models explored both in the above analysis and in other literature do not capture elements of video game narrative that are discovered, observed, or experienced. These narrative elements are often but not always non-core and optional, serving to increase narrative depth. By 'discoverable' I mean that the narrative element is not explicitly provided to the player and that they must actively seek the items. This often comes in the form of books, letters, and other collectibles that are spread around the game world for the curious player to find. 'Observed' refers to narrative elements that are consumed through a passive act of observation rather than consciously searched for. This sometimes appears in the form of environmental storytelling in that certain elements (such as text on walls) can explicitly tell a story. The term 'experienced' refers to the way in which players may be involved in a narrative sequence that inherently tells a story. Using the game's mechanics as a metaphor for another factor (whether in-game or real) can mean that the player experiences narrative inherently by interacting with the game's systems.

As a result, I have developed a model to capture these narrative elements termed Discoverable Narrative. The representation of such elements is defined as an amalgam of four dimensions, which when combined, can describe the narrative text.



Tangibility	Tangible Text attached to an in-game object.	Intangible Text **not** attached to an in-game object.
Functionality	Narrative Primary purpose is for narrative.	Mechanical Primary purpose is **not** for narrative.
Clarity	Explicit Clearly and well defined.	Implicit Abstract and interpretative.
Delivery	Active Requires interaction to be consumed.	Passive Is observed or experienced regardless of direct interaction.

Fig. 4.3 Discoverable Narrative four-dimensional matrix descriptor.

Figure 4.3 shows a matrix defining the four dimensions. The first dimension, `Tangibility`, defines the physical presence within the game. A `Tangible` element has a physical representation within the game, such as a weapon or a book, whereas an `Intangible` element has no such representation, such as a codex entry accessed from a menu. The second dimension, `Functionality`, defines the purpose of the element that the narrative text is attached to or is representing. A `Narrative` element has the primary purpose of conveying story, such as a piece of graffiti or a book, whereas a `Mechanical` object has a core function other than narrative, such as an in-game weapon having a description (i.e., the weapon's primary purpose is as a weapon, and the narrative text is secondary to that). The third dimension is `Clarity` which represents whether the narrative text has `Explicit` content that is clearly and well-defined, such as the transcription of an audio log, or if it is instead `Implicit` meaning the content is abstract and/or interpretative. The final dimension, `Delivery` describes how the content is to be consumed. `Active` elements require conscious interaction in order to consume, such as picking up a note to read it, whereas `Passive` elements require either observing or are experienced and exist as a narrative text regardless of direct player interaction, such as overheard conversations or environmental storytelling.

Examples

The Elder Scrolls V: Skyrim's Books

In *The Elder Scrolls V: Skyrim*, there exists a plethora of books, scrolls, and other written material scattered around the open world, many of which are hidden in catacombs or caves. Within the books is usually a mixture of text and images. As there is a physical in-game object associated with the narrative text, we can say they are all `Tangible`. Functionally, most of the books are `Narrative`, serving simply as lore or similar narrative purposes, but in some situations, finding and reading a book can grant the player a quest, and in these circumstances the books are `Mechanical`. The content of these books is `Explicit`. To be consumed, the books must be discovered and consciously interacted with by the player, therefore they are `Active`.

Left 4 Dead's Safe Room Graffiti

In *Left 4 Dead*[2], there are a number of safe rooms that act as checkpoints and no-harm zones for players spread throughout campaigns. the environments of these safe rooms are often used to present narrative filler of those lucky enough to encounter the room before the player. One such example is leftover graffiti, usually warning of some upcoming event. The graffiti can be considered `Tangible`, as there is a physical in-game object associated with the text. Functionally, they are of `Narrative` purpose as they play no other role than lore. Content is `Explicit`. The narrative text also occurs regardless of direct player interaction and is consumed through observation, so they are `Passive`.

BioShock's Audio Diaries

In *BioShock*, the majority of the world's backstory is presented through findable audio logs recorded from previous inhabitants of the city. These diaries are `Tangible` as their texts are associated with in-game discoverable objects. Functionally, most are of a `Narrative` purpose, although some do grant the player quests, in which case they would be `Mechanical`. Content for them is `Explicit`. To be consumed, they must be found and interacted with, and therefore are `Active`.

[2]Left 4 Dead. Valve Corporation. 2008.

Mass Effect's Codex Entries

The *Mass Effect* series has a large corpus of world and lore texts that are presented as an encyclopedia of sorts. These are accessible from the menu at any time. As the texts are not associated with an explicit in-game object, they are `Intangible`. Functionally, they are all of `Narrative` purpose. Content is `Explicit`. To be consumed, the texts must be located in the menu and selected, thus are `Active`.

Spec Ops: The Line's Mechanics as Metaphor

In *Spec Ops: The Line*, there is a sequence where the player's character is in a heated battle with a heavily armored and armed enemy. During this sequence, the screen repeatedly flickers completely black and attacking the enemy results in him turning into a mannequin for him to shortly appear back in human form elsewhere, which repeats a number of times. This is a demonstration of mechanics as metaphor for the gradual onset of post-traumatic stress disorder. As there is no specific in-game object associated with this text, it is `Intangible`. The functionality of this text is difficult to assign and is often dependent upon the developer's intentions. It can be considered `Mechanical` due to its nature as an in-game mechanic but could also be considered `Narrative` due to its impact on the player and the story. This could be a limitation of the model's descriptors and is an area for future clarification. In definition, it is `Implicit`, as there is no concrete instance of the text's content. The text is experienced rather than consciously choosing to interact with it, and the resulting effect is interpretative, making it `Passive`.

This page is intentionally left blank.

CHAPTER 5

Novella

Based on this analysis, an expansive review of other literature and narrative authoring systems, combined with general knowledge of video games structure, a framework for the modeling of video game narrative has been created. This framework is known as *Novella*. This section will outline the theoretical model and provide examples to demonstrate its flexibility.

5.1 | Theoretical Model

Initial works towards a model of video game narrative resulted in a partially functional yet flawed system due to the poor implementation of Discoverable Narrative, several elements that were difficult if not perhaps impossible to generally define the contents of, and a lack of meaningful extensibility [33]. The core concept behind this model is a three-layered system of Groups, Sequences, and Events, and how they are structured hierarchically and their connectedness. The three layers are similar in function but differ enough that when combined can create great complexity while remaining easy to understand. This model does not necessarily accustom itself with details of implementation but instead provides a specification and set of broad rules for interpretations to be built upon. It is intended that implementations will build up a catalog of Events and Logic functions for integration with their particular runtime. The model is structured with flexibility in mind, in that it can be easily appended to and extended for new features. Figure 5.1 shows a high-level UML diagram demonstrating the concepts behind the model. Each of the categories will be discussed in detail followed by a set of worked examples showing the functionality and flexibility of this model.

Story

The Story is a custodian of sorts for the entire narrative, being responsible for the creation and management of all narrative elements. All Variables are globally accessible and likewise managed by the Story. In this model, there always exists

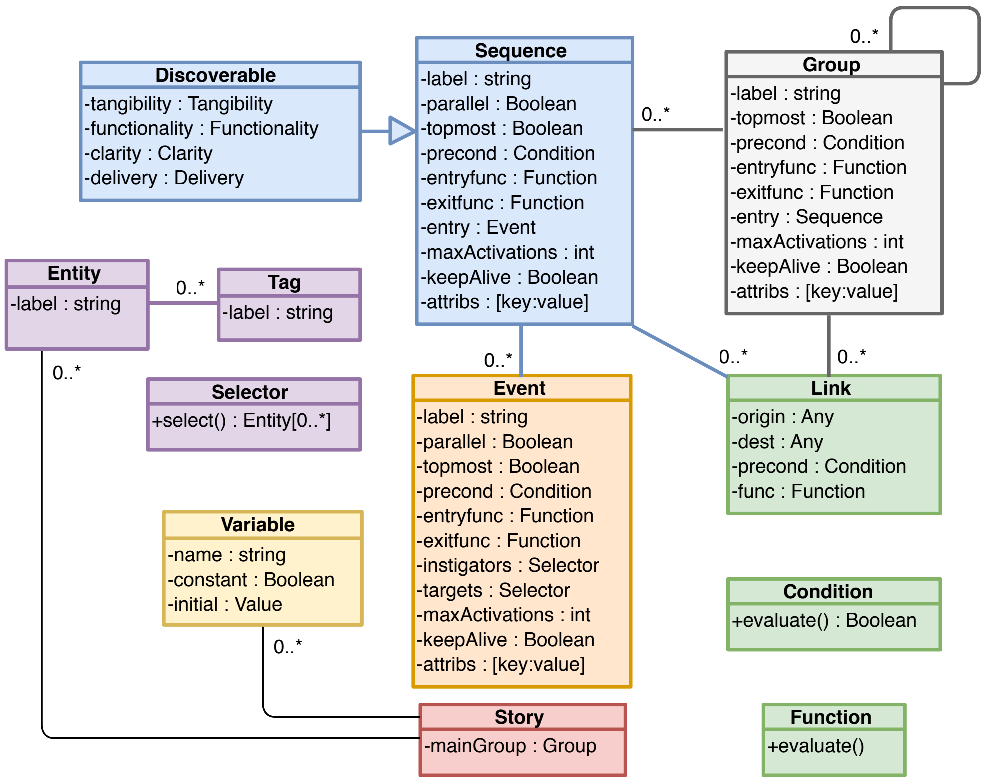


Fig. 5.1 High-level *Novella* model UML.

a single top-level Group in which all other narrative content exists, which is also stored within this Story object. At any point, the Story is able to run a set of Logic functions. This is particularly useful when external runtimes wish to query or modify the state of the Story, such as triggering an in-model element from an external source.

Variables

The state of the Story is controlled by a set of global Variables. A single Variable is a type-restricted instance of data, such as Boolean or integer, which can store arbitrary information about the Story. All Variables are mutable by default but can optionally be declared constant (i.e., cannot be modified during Story execution). Variables all have an initial starting value that is restored upon Story execution beginning. Since all Variables are global, their identifying names must be unique. As a worked example, the 'broom closet sequence' in *The Stanley Parable* has the player enter a closet a total of three times with different interactions each time. We could represent this with an integer Variable named `ClosetCounter`. This Variable could be read from to determine which sequence to present to the player and be written to so that the chosen sequence can progress.

Groups

Groups are the highest level container within the model. Within a single Story, there is only one main Group and every other Group within the story is nested. Groups can be indefinitely nested within one another for structural or organizational purposes. Groups act as a scope for their contents; elements inside are only able to act while the parent Group is also active. All nested Groups at the same depth (i.e., are siblings) run in parallel during story execution. For example, if a Group contained five nested Groups, all five would execute in parallel with no particular preference. However, Groups do not necessarily execute immediately, as they are guarded by a Condition determining when the Group will trigger. This means that while all Groups are indeed parallel to their siblings, their Conditions may result in different execution timing. It is therefore possible that Groups may execute instantly, with delay, or not at all, dependent upon their Condition. Groups also contain a list of Sequences. The connectedness of these Sequences is determined by a contained list of Links. Each Group also has a single entry point which is either empty or one of the non-parallel contained Sequences. If the entry point is set then that Sequence will be the starting point when the Group is entered during execution. If the entry point is empty, then it is assumed that there is no singular starting point of the Group (for instance, a group that contains only parallel Sequences). When a Group is entered and exited, a Function is run to optionally modify the Story state. Groups can be marked as topmost which means that when they are entered, everything in the model pauses execution until the Group exits, at which point anything paused as a result automatically resumes.

Sequences

Sequences are individual segments of narrative made up of one or more Events. They reside as children of Groups. Sequences are sequential as defined by their link structure in their owning Group, but can be optionally marked as parallel. Sequences also have a Condition that is used when marked as parallel. A parallel Sequence will attempt to trigger when its parent Group is active (first at entry and then every tick) by checking its Condition. As with Groups, this means that any parallel Sequence may instantly execute upon entry, with a delay once their Condition is met, or not at all if the Condition is never met. Within Sequences is a list of contained Events as well as a list of Links determining how they are connected. There is a single entry point of a Sequence which can be either empty or set to one of its non-parallel contained Events. If the entry point is set then that Event will be the starting point when the Sequence is entered during execution. If the entry point if empty, then it is assumed that there is no singular starting point of that Sequence. When the Sequence is entered and exited, a Function is run to optionally modify the Story state. Sequences, like Groups, can also be marked as topmost, which behaves in the same manner.

Discoverable Sequence

A Discoverable Sequence is a derivative of Sequence that partially implements Discoverable Narrative. It appends the four dimensions of Discoverable Narrative as enumerations to a Sequence, is always parallel, and therefore always has a Condition. Due to the hierarchical nature of this model, the parent Group acts as a scope to determine when the discoverable item can be found; if the parent Group is inactive, then the discoverable item is not available. The added enumerations act as labels and have no default influence over the Sequence's contents, although they can be queried by any Logic. Therefore the contained set of Events (or any area where Logic is used) can respond accordingly based on the value of the enumerations. This means that Discoverable Sequences are essentially a given condition causing a sequence of Events to trigger. Because of this, certain kinds of Discoverable Narrative that are more abstract, such as mechanics as metaphor and environmental storytelling, are not functionality that is easily represented in this model's description. This was intentional as to not over complicate the model, and that representing such abstract things is not a strength nor focus of this model. Attempting to wrangle every form of Discoverable Narrative into this model would certainly increase complexity.

Events

An Event is the lowest level object in the model, representing a single narrative event. Where Groups and Sequences can be submerged within due to their container-like nature, Events are the leaves of such a tree and cannot be divided further. This part of the model is designed to be flexible and expanded upon by deriving the basic concept of an Event for context-specific situations (such as dialogue, interactions, and so on). Therefore, the inner content of each derived Event differs based on the Event's requirements and what it represents. As with Sequences, Events are sequential with their connectedness being defined by the Links of their parent Sequence. Similarly, they can also be declared parallel during which their associated Condition will be checked as their parent Sequence is entered and every tick following. Functions likewise fire to modify Story state when the Event begins and ends. Events can also be declared topmost, acting the same as both Groups and Sequences. The base Event includes two Selectors. The first Selector determines the set of Entities that are instigating the Event, and the second determines the set of Entities that are the target of or are otherwise involved in the Event but did not necessarily instigate it.

Attributes

Groups, Sequences, and Events all have an associated dictionary that maps string-based keys to a value of any type. The values stored in the dictionary can be arbitrary based on the particular story's needs. Using this attribute system, it is possible to represent features such as the location in which the element (and its contents, where applicable) takes place. To illustrate, let's take the 'Medical Pavilion' level of *BioShock*, which is broken up into 11 subsections. We can represent the larger level as a Group and the subsections as Sequences. We can then represent locations using an attribute with the key 'location' set to a string representation of the location name. This can then be used by the runtime to determine which area the elements take place in.

While I have acknowledged the difficulty of implementing environmental storytelling as a form of Discoverable Sequence in this model, it is possible to emulate it in a simplified form using attributes. Each element can use a named attribute, such as 'environment', and store a textual string written by designers to instruct the implementation on how elements should be laid out. An interchange specification could be created that writers must adhere to that could become parsable by an implementation. As the complete modeling of environmental storytelling is a research topic in its own right, the detailed implementation of this feature is subject to future work and is not a core concern of this particular thesis.

Simulation Functionality

Groups, Sequences, and Events all have an integer determining the maximum number of times that they are allowed to (re)activate during simulation. This integer can either be positive, meaning that there is an upper bound to the number of activations, or it can be zero, meaning that there is no limit on the number of activations. This means that if one were to attempt to activate any of these elements, and that activation would cause it to exceed the activation limit (if nonzero) threshold, then the activation request should be declined. Additionally, all of these elements cannot be activated while already active (a script could attempt to activate an already active element, for example).

By default, Groups and Sequences will remain active only while their contents are currently active or have the potential to still be activated. Once the contents that could possibly activate have done so at least once, the element will automatically deactivate. If for some reason this is not desired behavior, then a Boolean for Groups and Sequences termed 'keep alive' will maintain the active state of the element even after its contents have expired. This could be used, for example, to force an exit Function not to trigger unless explicitly desired. Events, on the other hand, have no such contents and therefore this does not apply to them. Their lifetime is controlled by the duration of the in-engine event that takes place and are 'one-shot' by nature (i.e., activate, do something, deactivate).

Links

Links tie together two elements of the same type. A Sequence can only link to another Sequence, an Event only to another Event. Each Link has a static origin that cannot change, a mutable destination, a Condition determining whether or not the Link can be traversed, and a Function that fires upon traversal. During execution of the Story, if a set of Links are siblings (i.e., are outputs of the same element) then their Conditions are evaluated to determine which can be traversed. If more than one Link is able to be traversed, then the narrative must resolve the stalemate in order to proceed, such as by presenting a choice to the reader. If only a single Link remains, then it is traversed. If no links remain, the execution meets deadlock.

Functions

A Function is a set of logical statements that are able to modify the state of the Story and has no return type. Since it is mutable, it is able to both read and write Variables. Functions are used throughout the model to modify the state of the Story when structurally important moves are made.

Conditions

A Condition is a logical statement that results in a Boolean value. It can check the states of Variables but is unable to write to them. Conditions are used throughout the model to determine active states.

Entities and Tags

Characters and other narrative objects are enumerated as Entities. It is only necessary to list content that is of narrative significance, although this is not a restriction. An individual Entity is a unique instance of an object. For instance, if there are two identical doors, they must be enumerated separately. All Entities have zero or more Tags, where a Tag is a unique identifier that partially declares the owner's role within the narrative. All Entities sharing a common Tag are equally viable to fill the represented role. These Tags can be used as wildcards to enforce restrictions on Entity participation, such as only allowing Entities of Tag `Character` to be involved in a dialogue-related Event. Tags can be combined to create more pedantic restrictions. For example, in *Halo 2*[1], the enemy type Grunt acts largely as cannon fodder. Assuming all Grunts are enumerated as Entities with the common Tag `Grunt`, then all would equally qualify for any Event requiring an instance of that Tag, but not a particular instance. If we also assign the Tag `Weaponized` to some of the Grunts and require it, only those with both tags would suffice.

Selectors

A Selector is a logical statement that results in an array of Entities. Selectors query and are not mutating objects, meaning that they can only request and read the Story

[1]Halo 2. Bungie, Inc. 2004.

state rather than write to it. Selectors are used primarily within Events to determine the involved Entities.

Logic

The Logic of this model is a set of functions for interacting with the story State. This collection of functions is designed to be expanded by the implementation where fit and is in no way a complete taxonomy. This part of the model is where communication to external run-times should occur in implementations. This is ideal for adding custom per-game functions. For instance, an implementation that is integrated into a game engine may add its own function to determine whether any Entity with a given Tag is near the player, `isPlayerNearEntity(withTag: Tag, radius: Double) -> Bool`, which when evaluated would return true or false depending on whether the player Entity was within a given distance of any other Entity with a specified Tag. This could then be used as a conditional statement to trigger a Sequence, Event, and so on. A similar approach could be taken to implement any functionality for querying or manipulating an actual game through these Logic functions. The mandatory functions outlined in this model's specs can be found below in Table 5.1.

Extension and Integration

As mentioned earlier, this model provides a specification that is designed to be extended by the actual software implementation. There are chiefly two areas that can be extended by design: Events and Logic.

It is not possible to realistically imagine every kind of event that narratives have had in the past or will have in the future. Accounting for each possibility would be an endless task. Instead, this model has laid the ground rules for *what* and Event is, and then it is up to the implementation to fulfill types of Events particular to their use. For instance, an implementation requiring dialogue between two Entities may extended a *Dialogue* Event. The inclusion of instigating and target Selectors means that Entities can be dynamically assigned to any derived Event. The actual content of such Events is then left to the implementation. In this example, textual or similar content. In an ideal implementation, the data contained within an Event would be directly integrated into the runtime.

Logic is the second place this is designed to be extensible. The mandatory functions above are designed to provide minimum working functionality for most situations. However, as before, predicting every possible type of interaction between a runtime system and a theoretical model is not practical. Instead, this model provides a framework for what this Logic should do without detailing itself with the methodology used to fulfill the requirements. implementations are encouraged and expected to append this list of Logic with their own functionality that ties in directly to their runtime. This therefore allows the Logic to modify and query the state of the runtime in the implementation. This also means that in Functions throughout

Variables	
`setValue(var: Variable, value: Type) -> Void`	Sets a Variable value. Present for all data types.
`getValue(var: Variable) -> Type`	Gets a Variable value. Present for all data types.
Entities	
`entity(named: String) -> Entity`	Gets an Entity by name.
`entities(withTags: [Tag]) -> [Entity]`	Gets all Entities with the given Tags.
`add(entity: Entity, tag: Tag) -> Void`	Adds a Tag to an Entity.
`del(entity: Entity, tag: Tag) -> Void`	Removes a Tag from an Entity.
Components	
`trigger(type: Type, delay: Int) -> Void`	Triggers a given Group/Sequence/Event with a delay in ms. Present for all three.
`stop(type: Type, delay: Int) -> Void`	Stops a given Group/Sequence/Event with a delay in ms. Present for all three.
`pause(type: Type) -> Void`	Pauses a given Group/Sequence/Event. Present for all three.
`resume(type: Type) -> Void`	Resumes a given paused Group/Sequence/Event. Present for all three.
`tangibility(disc: Discoverable) -> Tangibility`	Gets the Tangibility of a Discoverable Sequence.
`functionality(disc: Discoverable) -> Functionality`	Gets the Functionality of a Discoverable Sequence.
`clarity(disc: Discoverable) -> Clarity`	Gets the Clarity of a Discoverable Sequence.
`delivery(disc: Discoverable) -> Delivery`	Gets the Delivery of a Discoverable Sequence.

Table 5.1 Mandatory Logic declarations for essential functionality.

the model, implementation-specific code can be called. For example, when a given Sequence is entered, an implementation-defined function could be called to trigger an in-game event.

5.2 | Worked Examples

In order to demonstrate the functionality of this theoretical model, a representative video game sample will be discussed in context of how to represent the narrative. This is then followed by further examples demonstrating support for Discoverable Narrative. As this model is designed to be extended in both Events and Logic, relevant assumed extensions in the pseudo-implementation will be mentioned. To keep the explanation reasonably simple, we will only look at a single instance of repeated events (such as dialogue) and observe and complexities that need to be handled.

5.2.1 | Life is Strange

A high-level overview of the sequence that we will study is displayed in Figure 5.2. Following the opening nightmare sequence, Max finds herself jolting awake in a photography lecture. Max then gathers her bearings while the lecture continues to take place. During this, Stella drops her pen, Taylor bullies Kate by throwing a paper ball at her, and Victoria's phone vibrates. These three events happen in parallel to the lecture and Max's pondering. At this stage, Max is able to interact with a photograph on her desk, and after doing so, four additional items on and near her desk become available. Interacting with such elements (except the camera) will not halt the lecture, but have Max narrate the item in parallel. The lecture begins to loop indefinitely with a series of four questions until Max interacts with her camera, upon which the narrative then continues. Max is prompted with a question for punishment of disturbing class, after which the school bell rings. This sequence was chosen as it demonstrates a number of typical narrative actions, but also includes multiple parallel events, gating of narrative progression, and explicit player choice.

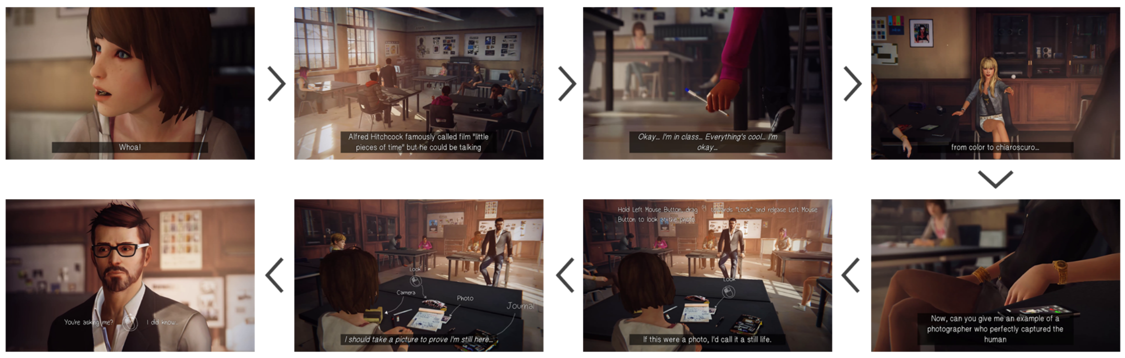

Whoa!
Alfred Hitchcock famously called film "little pieces of time" but he could be talking
Okay... I'm in class... Everything's cool... I'm okay...
from color to chiaroscuro...
Now, can you give me an example of a photographer who perfectly captured the human
If this were a photo, I'd call it a still life.
I should take a picture to prove I'm still here...

Fig. 5.2 High-level overview of the sequence.

Entities

We must first decide and enumerate which Entities are going to be involved in this scene. Depending on the granularity desired, different objects may be included. For instance, fine-grained representations may consider Taylor's paper ball as a unique Entity and define Events particular to the action of throwing it at Kate, whereas a more coarse approach may represent the act of throwing the paper ball as a single Event with only an instigator (Taylor) and target (Kate). This is largely a personal choice and can be influenced by the associated runtime. While it is possible to achieve this fine-grained approach, we will explore it from a more restrictive level for simplicity. As such, we can enumerate our characters as Entities: `Max`, `Jefferson`, `Stella`, `Taylor`, `Kate`, and `Victoria`. Tags will not be necessary in this example as we can rely on referencing Entities directly.

Groups and Sequences

Groups in this model are designed (but not restricted) as large-scale containers for a number of individual sequences. We can therefore assume that this classroom scene is part of a single Group. Since there are a number of individual happenings that run in parallel, and some that halt until given conditions are met, we need to define Sequences to represent this structure. Max talks to herself four times before being able to interact with all five items on and around her table. This happens while Jefferson's lecture is going on, and therefore we can put it in a Sequence named `Max`. Jefferson's lecture can be broken up into two separate Sequences: `Jefferson1` and `Jefferson2`. The former runs from the beginning of the scene with regular teacher-student interaction happening, but until Max interacts with her camera, it will continue to loop a series of four questions. The latter half, which involves querying Max and the class ending, is instigated by Max interacting with her camera. We can then enumerate the Sequences `Stella`, `Taylor`, and `Victoria`. The first is when Stella drops her pen. The second is Taylor bullying Kate by throwing a paper ball at her. The third is Victoria's phone vibrating. These three short Sequences happen in parallel to `Jefferson1` and `Max`.

Max's Inner Thoughts

During the `Max` Sequence, Max narrates to herself four times. This can be thought of as a form of dialogue with oneself. However, since the speech is inner and not vocal (i.e., thinking), it may be advantageous to differentiate from regular dialogue. We can assume that the implementation has defined a `Thinking` Event consisting of the textual content thought. In this kind of event, the instigating and target Entity would be the same Selector statement. In the four cases mentioned, the Selector would directly reference the `Max` Entity by name and the textual content would be whatever Max is thinking in-game.

General Dialogue

More general dialogue could be handled with a `Dialogue` Event, again provided by the implementation. This would largely mimic the `Thinking` Event but differ in that the instigator and target Entities do not have to be the same. For example, in the `Jefferson1` Sequence, Victoria and Jefferson engage in conversation. These individual speech acts could be represented as a sequential set of connected `Dialogue` Events where the Selectors directly refer to `Victoria` and `Jefferson` accordingly. In the case where Jefferson is addressing the whole class, then a Selector would be used to either gather all of the student Entities, or if a `Student` Tag had been used, select all Entities with that Tag instead.

Parallel Sequences

The `Stella`, `Taylor`, and `Victoria` Sequences all execute during `Jefferson1` at fixed times. This could be handled in a variety of ways in this model. If the specific timing is known, all three of the Sequences could be started with a delay upon entering `Jefferson1` by using the `trigger(sequence: Sequence, delay: Int)` function. They could also be triggered at a specific point such as after a `Dialogue` Event for more fine-grained control with or without delay. Alternatively, each of the Sequences could have a starting Condition that checks the truth value of a Boolean Variable, which can be set to `true` at any point, triggering the Sequences. As mentioned earlier, Max's four thoughts are parallel to Jefferson's lecture. This can be implemented by simply having the `Max` Sequence as a sibling of the `Jefferson1` Sequence and ensuring that they both execute at the same time.

Gating Progress

In the `Max` Sequence, there are five items on and around Max's table that can be interacted with: her camera, pencil case, photograph, journal, and backpack. However, four of them remain unable to be interacted with until the photograph is first picked up. This could be implemented in multiple ways. A simple way is to have all five elements as non-parallel Events within the `Max` Sequence. The photograph Event would follow Max's pondering. This Event would then connect to all of the five Events, including itself. In this model, this multiple connectivity means that the player is given a choice of which option to follow. All other Events except the camera would likewise link to all five Events. The result is that the player must firstly interact with the photograph before being able to interact with the remainder of the objects. These objects can then be looped due to their connectivity.

If Max does not interact with her camera in time, the `Jefferson1` Sequence begins to loop a sequence of four dialogs from Jefferson to the class. Once Max interacts with her camera at *any* point that it is available during `Jefferson1`, the Sequence terminates and the `Jefferson2` Sequence initiates. This can be handled in the model by setting the Function of the Event representing the camera interaction to use the `stop(Jefferson1, 0)` and `trigger(Jefferson2, 0)` functions in order. This would firstly terminate the `Jefferson1` Sequence regardless of its progress and start the `Jefferson2` Sequence immediately thereafter. This ability to wait until the loop or interrupt the Sequence at any time is how the game is designed.

Player Choice

After Max interacts with her camera, the `Jefferson2` Sequence starts. In this Sequence, Max is made to answer a question as punishment for disturbing the class. This question provides two options to the player, both of which reconcile to the same Event that follows to give the perception of agency over the narrative. This can be implemented in a simple way. As mentioned in the model description, any Event with multiple outputs is treated as a choice. Therefore, we simply need to ensure that the preceding `Dialogue` Event (Jefferson asking Max the question) is linked to both potential answers. The way in which the choice is presented to the player is again left up to the implementation, as it is not possible to incorporate every form of choice of the past and future. Similar cases later in the game are more complex, as they pause all other parallel Sequences, Groups, and Events in execution. This behavior can be achieved by using the `pause` and `resume` functions, or by marking the appropriate element as topmost.

5.2.2 | Discoverable Narrative

As mentioned earlier, Discoverable Narrative is implemented as a derivative of Sequence. The following examples demonstrate how Discoverable Sequences can be used in the model.

The Elder Scrolls V: Skyrim's Books

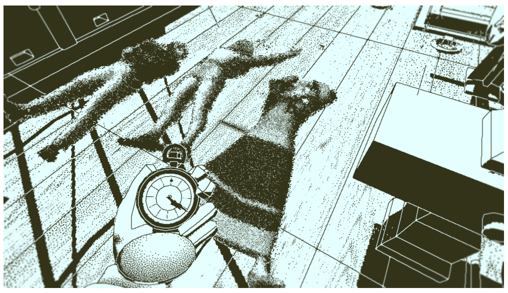
As mentioned in Chapter 4, books and other notes can be found scattered around the game and are mostly optional. The condition required to begin consumption of these texts is to locate the objects and consciously interact with them. Once activated, the item is animated towards the screen and the player is presented with an interactive book. The rest of the game suspends while the player reads the text. Assuming we have created a Discoverable Sequence and configured it accordingly, we can define its Condition in a few different ways. As we have the freedom to trigger Sequences using Logic, the runtime could communicate with the implementation and trigger the Sequence directly when it internally detects interaction. Alternatively, a Boolean Variable could be checked for and set at any point by the runtime. In both ways, the gating of the Sequence is done within the model, but the detection and triggering comes from the runtime. When the Discoverable Sequence triggers, the set of Events can differ based on what needs to be done in response to the Discoverable Narrative being activated. In this case, the implementation can declare an Event for reading of the text, of which the detailed behavior would be deferred to the runtime. This Event can be marked as topmost to ensure the rest of the elements halt while the text is consumed.

Return of the Obra Dinn

Return of the Obra Dinn[2] is a contemporary murder mystery game that makes extensive use of Discoverable Narrative concepts. The game is presented in chapters that are made up of a varying number of parts. These parts form a sequential narrative but the player does not necessarily access them in such a way. The player often begins in the latter half of a chapter and then works their way backward by

[2]Return of the Obra Dinn. Lucas Pope. 2018.

finding the deceased within each part. For example, if the player enters part 4, they will be able to find and enter at least the previous part, and sometimes more. Players must locate the deceased (or related items) and interact with them, which transports them to a freeze-frame of the moment of death, aiding them in solving the mystery.


(a) Entering Chapter IV, Part 4 by interacting with Bun-Lan Lin's body. Parts 1, 2, and 3 unavailable at this point.

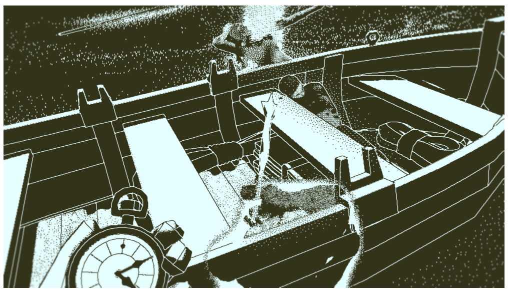

(b) Entering Part 2 from within Part 4 by locating and interacting with Patrick O'Hagen's body.

Fig. 5.3 Entering Part 4 and discovering Part 2.

Figure 5.3a shows Bun-Lan Lin's body on the main deck, which when interacted with will transport the player to Chapter IV, Part 4. There is no way to access part 1, 2, or 3 without firstly entering Part 4. Once the player enters this particular scene, they are able to locate and interact with previously deceased characters which act as triggers to backtrack through the rest of the chapter. Depending on the scene, players may be able to travel only a single part back sequentially, whereas in others they may be able to freely pick between multiple previous parts. In the scene with Bun-Lan Lin, the player is able to interact with three of the deceased, making up parts 1, 2, and 3. This can be seen in Figure 5.3b with Bun-Lan Lin on the right and Patrick O'Hagen in the foreground. Locating and interacting with Patrick's body will teleport the player back to part 3.

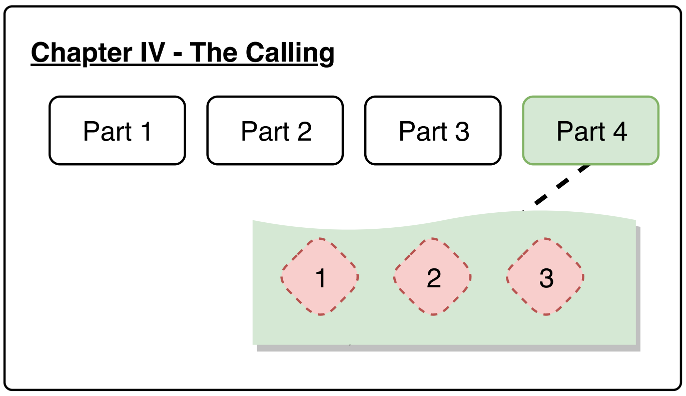


Fig. 5.4 High-level possibility of Discoverable Sequences in *Return of the Obra Dinn*.

We can model this using *Novella* by using nested Groups, Discoverable Sequences, and Logic. We can safely declare a main Group for the overall chapter named `ChapterIV`. Implementing each part as a Sequence would not be wise, as only Events can be contained within them. Instead, we can declare each part also as a

Group and nest them as siblings within the `ChapterIV` Group. This form can be seen in Figure 5.4. The Group `Part4` is expanded in the figure. Displayed in this expansion is three Discoverable Sequences, one for each of the preceding parts. We can assume for the sake of this example that the contents of such a Group has been implemented and that each of the previous parts have this format replicated for their own preceding parts accordingly. Now we can simply define a given Condition for each of the Discoverable Sequences using Logic decided by the implementation. In this case, we could perhaps define a function for checking interaction with a given Entity, or alternatively handle the interaction in the runtime and manually trigger these Discoverable Sequences by name. We can then use the exit Function of the Discoverable Sequences to handle the consequence of interacting with the deceased character. In this case, we could terminate the current Group by using the `stop(group: Group, delay: Int)` function, and start the new Group by using the `trigger(group: Group, delay: Int)` function. If we want a given set of Events to trigger before switching Group, then we can embed them within the appropriate Discoverable Sequence, but this is not necessary.

With this solution, a Discoverable Sequence is not that different from a regular Sequence. In fact, this could be implemented by a plain Sequence without any trouble and act functionally identical. The difference comes in the ability to query the Discoverable Sequence's enumerations so that the internal Events (or others) can alter themselves based on the states if so desired. It could also be useful for an implementation to differentiate between what is a Discoverable Sequence and what is a regular Sequence regardless of functionality.

CHAPTER 6

User Experience

This chapter introduces essential theory of UX. The use of UX in this work chiefly surrounds the examination of user interface paradigms. While I acknowledge that the staggeringly large field of UX goes far beyond interfaces, this is the primary context in which it will be discussed and applied. Therefore, the background discussion of UX principles is kept compact and scoped, serving as an illustration of UX concepts surrounding interfaces. This chapter will begin by broadly defining what is meant by UX, followed by a description and analysis of related works and heuristics mostly relating to interfaces.

6.1 | Defining User Experience

We can trace practices reminiscent to that of modern UX with the introduction of the *Human Factors Engineering Department* at Bell Labs in 1947, founded by the renowned psychologist John Karlin. In Bell Labs' own historical recounting [35] of its use of applied behavioral sciences, they state that while plenty of user-oriented design had been done prior to the 1940s, particularly with the physical and audible design of telephone handsets, that it was around this time that true dedicated behavioral research started to take place. One of the most prominent works of this department, led by Karlin, was the user-based research made into keypad layouts for telephones. At this point in time, it is clear that UX work was taking place, but there was not yet a dedicated label for such practices.

The elusive notion of what 'User Experience' is defined as has troubled the academic space for quite some time. There have been attempts to consolidate various definitions and analyze their individual contributions to determine a single description of what UX really means [51]. Despite this, many definitions are still disputed [6]. Even in recent years, after the release of the ISO standard of what UX is[1], people are still coming up with new ways of tackling the definition of UX, including the use of formal methods to analyze the ISO descriptions into its constituents [57].

[1]ISO 9241-210:2010

The actual phrase 'User Experience' is supposedly coined by one of the forefathers of modern UX, Don Norman. In the 1990s, he was hired at *Apple* and established what they named the *User Experience Architect's Office* [60]. The goal of this group was to form a specific department of UX professionals and have them be part of the design process throughout development. Norman's vision of UX encompasses all aspects of the end user's interaction with the company, any of its services, and the products which it creates, whether indirect or direct. In an interview[2], Norman states that he created the term 'User Experience' as he wanted the phrase to cover *"all aspects of a person's experience with a system, including industrial design, graphics, the interface, the physical interaction, and the manual"*. He also notes that since this original coinage, the phrase has become widespread, so much that it is starting to lose its meaning or becoming ambiguous, as we have seen earlier.

Regardless of which definition we use, UX has expanded greatly since its inception and now encompasses a broad variety of practices that all relate to the experience with a product or service that an end-user has. One of the most common elements, and often the most visual, is interface design. With respect to this thesis, we will be using interface design theory to determine the impact design and interaction decisions have on the authoring process.

6.2 | Interface Principles

It is essential to understand some interface design heuristics in order to be able to effectively identify and criticize them in real-world applications. In particular, this comes in useful in experiments involving interface evaluation, of which is later tackled in this thesis, as we will be able to not only identify issues with interfaces but look at corresponding theories as to why this may be.

Both literature and industry professional make it clear that most if not all interface design heuristics are broad rules of thumb and are not specific laws that must be followed. Therefore, when discussing these principles of interface design, regardless of terminology used, the notion that they are not hard-set rules must be considered. In a majority of cases, the heuristics will apply, but an interface violating such definitions is not necessarily incorrect, as it may have a good reason for doing so. On the contrary, if a significant amount of violations take place, then there is reason for concern.

One of the most prominent works on interface design is Ben Shneiderman's *Designing the User Interface* [66]. This work, originally published in 1987, still remains one of the foundational works in interface design theory. As part of this book, Shneiderman outlines '8 Golden Rules of Interface Design'. These rules act as essential principles for designers to follow in most interactive systems. The eight rules can be summarized by their titles: *Strive for consistency*, *Seek universal usability*,

[2]www.adaptivepath.org/ideas/e000862

Offer informative feedback, *Design dialogs to yield closure*, *Prevent errors*, *Permit easy reversal of actions*, *Keep users in control*, and *Reduce short-term memory load.* Shneiderman makes it clear that these principles must be interpreted, refined, and extended based on the context they are used in. They act as starting points for informed decisions to be made upon and do not necessarily need to be adhered to.

A similar seminal work is Nielsen's[3] work on 'Usability Heuristics' [59]. This work took a large set of existing usability heuristics and attempted to synthesize a new set of heuristics that are as good as possible at capturing the usability problems that occur in real-world systems. Nielsen used a large set of real-world problems and scored the collection of existing heuristics based on how well they explained the problem. A range of factor analysis techniques was then used to determine how well the individual heuristics captured real-world usability problems that users encounter. This work was different from others as it not only analyzed existing heuristics but determined which were statistically the most important to consider when designing interfaces and interactions. Broadly, the top ten heuristics identified can be summarized by their titles: *Visibility of system status*, *Match between system and the real world*, *User control and freedom*, *Consistency and standards*, *Error prevention*, *Flexibility and efficiency of use*, *Aesthetic and minimalist design*, *Help users recognize, diagnose, and recover from errors*, and *Help and documentation.*

6.2.1 | Borrowing from Psychology

Many UX principles are derivatives of or directly applied theories from psychology and social sciences. Given that a large part of UX focuses on the innate interactions, perceptions, and behavior of humans with interfaces, even subconsciously, this makes sense. Understanding a variety of these principles will help us in the analysis and identification of behavior of participants in usability testing, which this thesis later deals with.

Fitts' Law

Description. The psychologist Paul Fitts investigated the human motor system and determined that the time required to move to a target depends on the distance to it, but also inversely relates to the target's size [29]. In other words, long movement and smaller targets results in greater error and lesser accuracy, whereas short movement and larger targets have the opposite effect.
Application. The equations and takeaways from Fitts' original work have come to be known as *Fitts' Law.* This law is easily applicable to UX practices and interface design. Simply put, if you want elements to be easily selectable and noticeable, then make them large and reduce the distance that the user must travel to a minimum (usually with a pointing device, but increasingly with their hands such as with phones and tablets).

[3]Jakob Nielsen is the co-founder of the Nielsen Norman Group along with Don Norman.

Hick-Hyman Law

Description. The British and American psychologist team of Hick and Hyman set out to identify relationships between stimulatory intensity and reaction time [37, 38]. The pair found through multiple experiments that the more stimuli a user is presented with, the longer it takes them to make a decision. Explained by Hick directly, *"the amount of information extracted is proportional to the time taken to extract it, on the average"*.
Application. In terms of UX and interface design, this means that the more choices a user has to make, the more they will have to think about making the choices, and therefore their experience will be slowed down. Consequently, reducing the number of choices a user has available, or at least reducing the complexity of the choices involved (such as by breaking them down into subsequent steps), can aid users in making choices quickly and efficiently.

Gestalt Laws of Grouping

Description. The Gestalt Laws of Grouping (otherwise known as Gestalt Principles or Laws) refers to a large body of seminal work that observed organization of perceptual scenes. This work was initiated by Wertheimer [72] and was further developed by Köhler [47], Koffka [46], and Metzger [55]. Their works combine to give us a set of observations about the psychology of perceptions of relationships between objects.
Application. All of the Gestalt Principles have influenced UX and interface design. The Proximity Principle, for example, defines that elements tend to be perceived as aggregated into groups if they are near each other or share a common border, and are considered separate if visually disconnected. On the other hand, the Similarity Principle states that elements tend to be interpreted as grouped together if they are visually similar to each other, even if separated. This means that interfaces designers must take care when determining whether or not elements should be considered grouped and be careful as to the method they choose to avoid accidental groupings. The original principles can be found in many interpreted forms, particularly reworded to fit modern UX.

Miller's Law

Description. In 1956, psychologist George Miller contributed one of the most well-known papers in psychology, which has come to be known as *Miller's Law* [56]. Miller determined that the average person can hold seven, plus or minus two, objects in working memory (that is, the memory that temporarily holds information for processing rather than for long-term purposes).
Application. In terms of UX, this means that designers should ensure content and interactions are presented in a way that is manageable by users. Delivering content in groups of five to nine at a time is a safe amount to use. If designers present users

with too much information at once, they may struggle to cope with it all at the same time and become lost.

Serial Position Effect

Description. Another prominent work contributed to the field of psychology was that of Ebbinghaus on human memory [26]. The original work is in German and was written in 1885, and as a result many interpretations exist. One of the results of Ebbinghaus' work was what we now call the *Serial Position Effect.* This states that when people attempt to recall items from a list, that they will remember the first and last items better than the items in the middle. Remembering the first items is called the *primacy effect* and remembering the last items is called the *recency effect.*
Application. The Serial Position Effect is widely used in UX but is also easily violated. Ideally, it means that designers should strive to place important items at the extremities of lists with items of lesser importance occupying the middle. This does not only apply to literal lists. Where there is flow between elements, this heuristic can generally apply, such as in navigation bars or even the extremities of application windows.

This page is intentionally left blank.

CHAPTER 7

Conclusion & Future Work

This final chapter aims to highlight and summarize the important aspects of this thesis and to set out a broad plan that will guide this PhD to completion.

7.1 | Summary

This thesis sought to investigate modeling of video game narrative and to identify relationships between UX practices and their impact on the workflow on authoring for interactive narrative authoring systems.

Video Game Narrative

The main body of work started by defining what is meant by ‘video game’. The infamous ‘Ludology vs. Narratology’ debate was briefly commented on, noting that video games can indeed have narratological theory applied to them. The remainder of this chapter then highlighted the difficulties that can be encountered when attempting to model video game narrative. These difficulties usually come in the form of heightened difficulties in other fields such as increased player agency over the narrative, numerous methods of interaction with the narrative, and some unique methods of narrative presentation seldom found in other mediums.

Hypertext & Interactive Narrative

Hypertext, which is often associated with narratological studies and development, was then covered from its origin to its applications, including that of the related field of interactive narrative. This chapter sought to outline the theoretical background of hypertext as it forms the basis of many theoretical models, and can be used to form models of video game narrative, too.

Modeling Video Game Narrative

A number of contemporary models, some game-specific and some not, were applied to a selection of video games to identify successes and downfalls. Finding what elements of video game narrative existing models can and cannot capture is vital when devising a new model, as the problematic areas can be targeted. The result of the analysis aided in the development of the concept of Discoverable Narrative.

Novella

The analysis results and discussions from the previous chapters were combined to create a theoretical model of interactive narrative, chiefly targeting video games. This model was covered in detail and some worked examples were provided to demonstrate its flexibility. Additionally, a prototype implementation of the model in an authoring tool context was looked at, but further development is reserved for future work.

User Experience

A high-level overview of the history of the phrase 'User Experience' was conducted. This was followed by highlighting a number of key principles of interface design in the form of heuristics, as well as briefly exploring several specific heuristics that originate from psychology and social sciences. The principles discussed here will be of most assistance once further usability experiments take place in the analysis stage.

7.2 | Contributions

This thesis has explored a few areas of research and made contributions in each. The first three contributions directly target the first research question. The fourth contribution begins to tackle the remaining two research questions.

1. *A Discussion of Video Game Narrative Difficulties*
   A number of difficulties to consider when modeling video game narrative were identified and detailed with clear examples. I acknowledge that this listing is incomplete, and perhaps will never be complete, due to the rapid expansion of video game technology. However, it still serves as guidance to those designing or developing a model of video game narrative.

2. *A Model of Discoverable Narrative*
   A method for representing narrative elements that are discovered, observed, or experienced was proposed. This consisted of a 4D matrix with each dimension capturing a different property of the narrative element. A number of examples were provided to demonstrate how the method can be applied to broadly represent real-world video game narrative situations.

3. *Novella, A Theoretical Video Game Narrative Model*
   A primary contribution of this thesis is the theoretical model, *Novella*. This model is suitable for representation of interactive narrative, chiefly focusing on video game narrative. *Novella* is a hierarchical model based on Groups, Sequences, and Events. It is designed from the ground up with extensibility and integration into existing runtimes in mind.

4. *Clustering of Authoring Tools by Feature*
   As part of the upcoming experiment (which will be discussed shortly), an analysis and clustering of 29 authoring tools based on their UX and system features was conducted and analyzed. The result is a listing of popular authoring tools grouped by similarity in terms of their UX and system features.

Future contributions this thesis is working towards include an *implementation of the Novella model in an authoring tool*, an *initial experiment identifying links between UX practices and impact on authoring workflow*, and a *further experiment confirming relationships between UX and impact on authoring workflow.*

7.3 | Roadmap

This section will outline the high-level plan for the remainder of this project. Figure 7.1 shows a Gantt chart of the remaining time and the tasks involved. This plan will serve as guidance to ensure that the original research questions are answered. All times listed are subject to flexibility.

First Experiment & Analysis

This experiment, of which the procedure is outlined in detail below, is expected to take place from sometime in March. The entire process, including analysis, should take around one and a half months. Please see the section below for more details.

Refined Novella Model & Authoring Tool Prototype

Following completion of the first experiment, refinement of the theoretical model, development on the prototype implementation, and further research will take place. This is expected to continue until near the end of the PhD.

Second Experiment & Analysis

The second experiment has yet to be designed in full. The first experiment is trying to discover which common UX paradigms there are in game narrative authoring tools and what their impact is on authoring workflow. This second experiment would instead seek to test combinations of these identified paradigms, or exclusions of paradigms from otherwise complete applications, to test and confirm what impact the paradigms have when extracted from their original source. This would be done chiefly within the context of an authoring tool developed around the *Novella* model.

Write Thesis

The thesis would be written after the final experiment while the back end of development continues. As development ceases, focus would then turn to the thesis exclusively. Ample time has been prepared for the final thesis.

Expert Feedback

During the latter month(s) of development of the prototype implementation, detailed expert feedback will be sought after. While this could not be interpreted as statistical findings, having feedback from subject experts is invaluable.

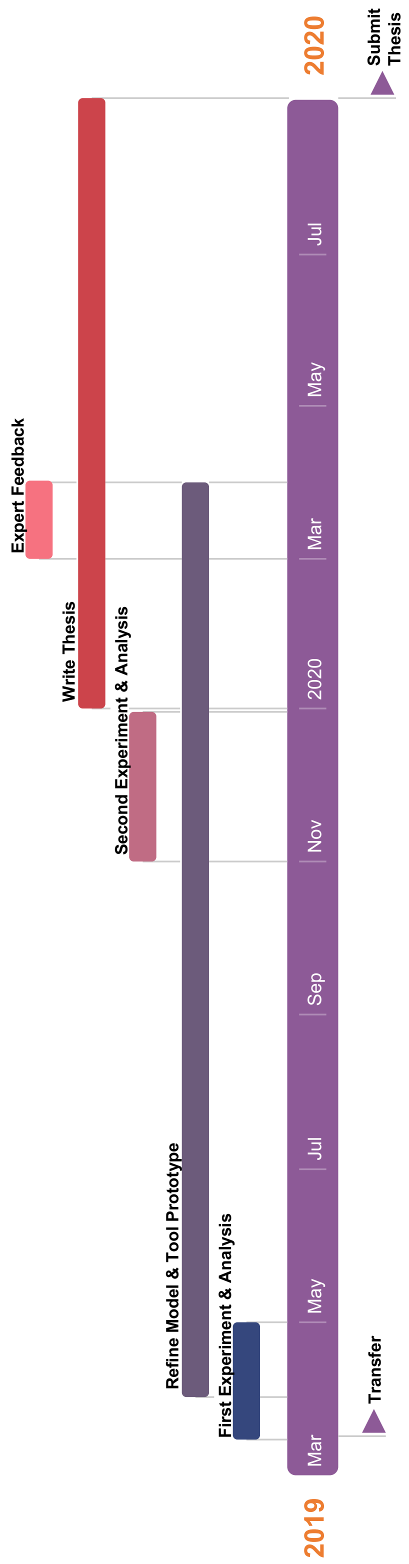


Fig. 7.1 Remaining PhD timeline.

7.4 | Experiment 1 Preparation & Methodology

The work presented in this thesis thus far is subject to modification and further experimentation. In particular, working towards validating the impact of various interface paradigms on the workflow of authoring interactive stories is a priority. This section will detail the current progress of an upcoming experiment that will gather data on this topic for analysis. The resulting data will then be used to influence my own application's design and serve as a base for building a set of informed best practices for developers and authors alike.

7.4.1 | Overview

A major outcome of this work is the analysis of links between interface paradigms and their impact the on workflow of authors creating stories in interactive narrative authoring systems. In order to begin tackling this query, I am going to be running an experiment that intends to sample the current state-of-the-art in interactive fiction authoring tools. The expected outcome is the identification of common features between the systems, detection and analysis of any clusters, and observation of the spread of different approaches taken to support authoring of interactive storytelling. While this project is not scoped exclusively to video games narrative, it is a primary focus, and as such, the experiment's methodology design reflects this.

Research Questions

For this experiment, I am trying to find out two things. Firstly, *what is the impact on UX of UI paradigms in interactive narrative (chiefly games) authoring tools*? I want to find out what impact the various user interface paradigms have on the UX in interactive narrative authoring tools. This is being asked in an attempt to discover any links between paradigms and their impact on the authoring for game writing. If it so happens that a given tool correlates with poor participant task completion rates, this could then be further investigated as a problematic link between the given paradigms and the UX of authoring. Second, I want to find out *what is the impact on authoring workflow for given UI paradigms*? It would be useful to know the impact on the ability to create the desired result within each given tool and their paradigms. I am attempting to better understand what creative impact a given tool has with respect to its interface paradigms on the authoring experience. From this, I could identify which paradigms better support creative freedom and which hinder it.

7.4.2 | Representative Tools

While in an ideal situation every authoring tool related to the field (and perhaps more) would be tested individually in a longitudinal study, this is not feasible due to the high demand on resources. As such, both the tasks and tools involved will be replaced with a representative subset instead.

In total, I have included 29 individual tools. 14 were sourced from academically published literature. Four are developed and sold as commercial products. The remaining 11 come from other non-commercial, non-academic sources such as open-source and otherwise free products. This distinction is important as the purpose of these tools can differ based on their origin; commercial products are developed with a different goal than research projects, which can impact the focus and quality of the tools. This must be kept in mind when choosing a representative sample and when interpreting results.

Availability

When evaluating included authoring tools, it is important to determine their availability. Availability not only refers to the accessibility of their end product (i.e., binary, web service, etc.), but also their online presence and whether their source code is available in the case of products that are able to do so. Historical data is also taken into account with this definition, such as determining that a software used to be available but has since become unavailable.

The availability of a tool directly affects its ability to be adopted by an authoring community and its ability to be further developed or otherwise used for research. Systems that have long become dormant can become devalued relative to their initial contribution, as they are unlikely to be adopted in the long term by authoring communities. *Inform*, for example, has been used and developed since 1993, but due to its strong online presence and dedicated community-driven development, remains a strong contender among the interactive fiction community even today.

Online presence is an area that academic authoring tools struggle to maintain. Of the sampled tools from an academic background, only six ever had a dedicated website, of which only four are functional as of today and are seldom, if at all updated. Temporary websites often result in unreachable links and are not a suitable substitute for a dedicated page where tools can be publicized. To illustrate, an article of Emo-Emma[1] provides links to academic papers and binaries, but all links reside on a university staff page that no longer exists making it difficult if not impossible to source the original contribution.

Another area that the sampled academic tools struggled with is the distribution of binaries and source code. Some projects are understandably protected intellectually, but those that are not should attempt to share their work in order to better maximize their chance of adoption by authoring communities, and provide opportunity for further research to be conducted. Of the fourteen academic tools, only five ever offered binaries at some point, with only one remaining publicly available today (StoryPlaces [36], which, at the time of writing, offers both an end-user web service

[1]www.redcap.interactive-storytelling.de/authoring-tools/emo-emma

as well as source code). Sometimes software is made available to authors upon request. StoryTec [32], for instance, used to provide a software request form but has since removed it in 2015[2]. ASAPS [45] has a request procedure that I had followed in mid-2018, but was unsuccessful in obtaining the software as no response was provided. This highlights the need for care to be taken to ensure that if software is intended public use that it is made, and remains, easily available, otherwise any traction gained could be rapidly negated. The lack of availability in the academic circle has serious research implications for the interactive fiction authoring research community, as it prevents the reproduction of any experimental results and hinders their study so that the community might incrementally improve and iterate upon existing work, or form a greater understanding of the existing works. While some conclusions can be drawn from documenting articles or publications, this is not a suitable replacement for the software itself.

Features

In order to determine potential clusters of the tools, it's important to have a wide selection of features that help distinguish programs from one another. Having an abundance of features that are overly common or under expressed will not result in any useful clustering outcomes. It was decided that the features should come from UX and systematic features of the applications instead of the model details. The way in which the models work is not considered important as that would distinguish the actual models themselves, which is not what this experiment is interested in. The features are listed below with descriptions.

- **Error Handling** (`None`/`Buildtime`/`Runtime`)
  The way in which errors are presented to the user.
- **Syntax Highlighting** (`Yes`/`No`)
  Scripts have keywords or phrases distinguished in some way, usually by color or other similar attributes.
- **Autocomplete** (`Yes`/`No`)
  There is some form of intelligent assistance for completion of input.
- **Launcher** (`Yes`/`No`)
  There is a launcher or dashboard interface at startup, often including helpful information and project management.
- **Node View** (`None`/`Static`/`Dynamic`)
  A node view is an interface paradigm for a schematic-like interface with individual nodes/objects being connected usually by lines. Node views that are only for visualization (i.e., cannot edit content using them) are static, and those that can edit content are dynamic.

[2]The download page still exists but is not accessible from the main website as the link was removed. The download itself is now password protected and the form to request a login no longer exists. This was discovered using web crawling with Archive.org's WaybackMachine.

- **Can Duplicate Content** (`Yes`/`No`)
  Is the user table to duplicate existing content to save time recreating them?
- **Structural Shortcuts** (`Yes`/`No`)
  Accelerators or shortcuts are present to quicken creation of common content linkage structures.
- **Autolayout Content** (`Yes`/`No`)
  A method is present to automatically layout or clean up content structure.
- **Link Parking** (`Yes`/`No`)
  The ability to temporarily host a link between pieces of content to ease the burden of connecting visually separated content.
- **Source Editor** (`Yes`/`No`)
  An editor, usually augmented in some way, for script entry.
- **Content Browser** (`Yes`/`No`)
  Some form of browser that lists the content, such as a filmstrip or a list of entries.
- **Searchable/Filterable** (`Yes`/`No`)
  Content can be searched or filtered to make things easier to find.
- **Relationships Method** (`Visual`/`Event`/`Embedded`)
  The way in which content is connected together such as visually (lines, etc.), embedded inside content, or triggered by events.
- **Statistics** (`Yes`/`No`)
  Some kind of story and/or content stats are available for analysis (e.g., word count).
- **Edit Method** (`Inline`/`Inspector`/`Modal`)
  The primary method of editing content. Inline fields allow direct editing of content without opening a separate interface. Inspectors are property panels that show once something is selected. Modal popup editors block all other activity until dismissed.
- **Can Preview** (`Yes`/`No`)
  Is the system able to preview the story?
- **Simple Debugging** (`Yes`/`No`)
  Does the preview system provide any features beyond end-user reading such as tracking and observation of story state?
- **Modify State Debug** (`Yes`/`No`)
  Is it possible to modify the state of the story during debug previewing without causing permanent changes?
- **Standalone App** (`Yes`/`No`)
  The application runs as an independent program on a traditional operating system.
- **Web App** (`Yes`/`No`)
  The application runs in a web environment.

- **Integrated App** (`Yes`/`No`)
  The application is built into an existing system (i.e., as a plugin).
- **Mobile App** (`Yes`/`No`)
  The application runs on mobile devices such as tablets.
- **Documentation** (`Yes`/`No`)
  Documentation is available for non-technical users.
- **Templates/Samples** (`Yes`/`No`)
  Premade samples or templates are available to users as starting points or tutorials.
- **File Format** (`JSON`/`HTML`/`XML`/`GBLORB`/`Custom`)
  The final data format the application outputs.
- **Able to Export** (`Yes`/`No`)
  Is the program able to export its data to another format?
- **Export to Runtime** (`Yes`/`No`)
  Is the application export suitable for use in another existing system (e.g., another reader/interpreter)?

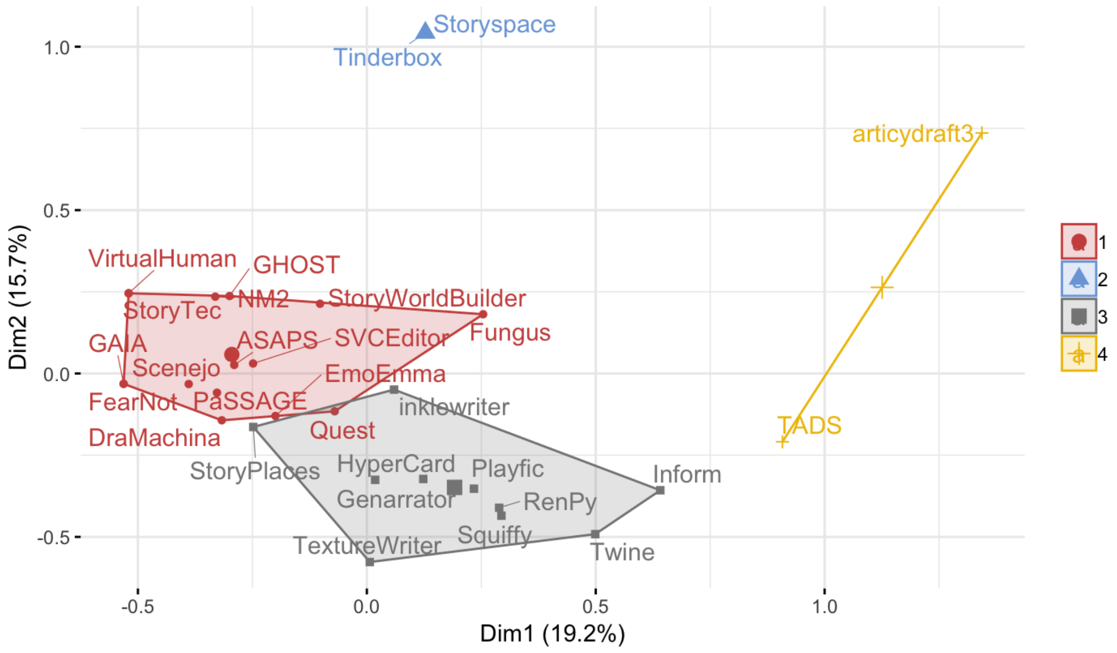
Each tool was systematically reviewed against the features through a combination of usage (where possible) and a review of relevant documentation and research papers. It should be noted that while every effort was made to retrieve and use the tools, in some cases, as described above, the tools were not available. In the case of a tool not being available to use, features were inferred from research papers or other published material. If a given feature for a tool could not be ascertained, the feature was assumed to not be supported instead of using imputation to declare the possibility of the feature, as that was considered unfair.

Analysis

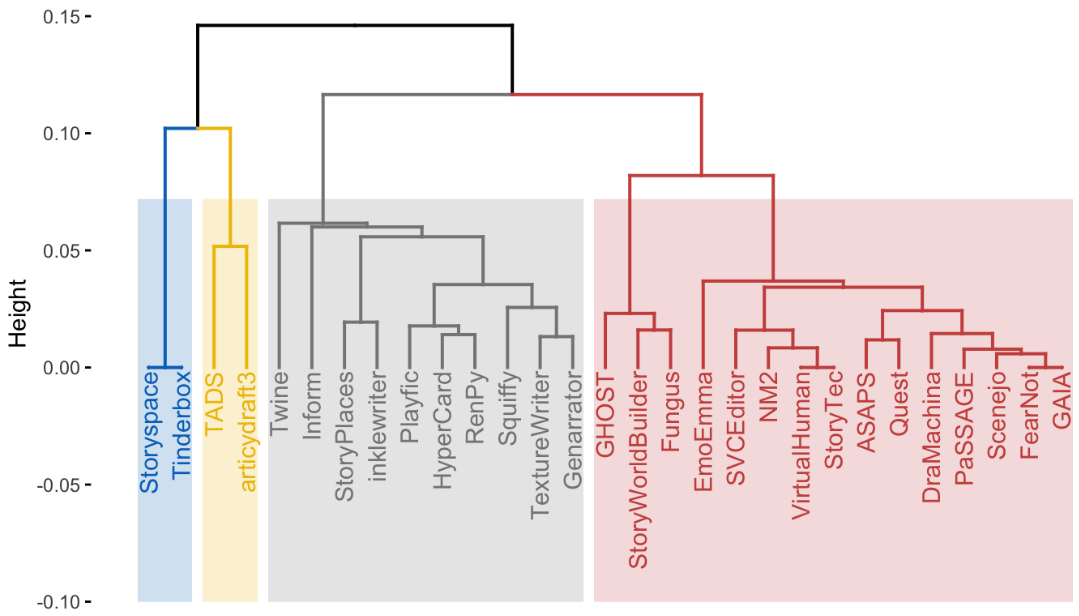
Statistical analysis was done using the **R** language and related third-party packages. As the data is purely categorical (i.e., only enumeration and binary columns), Multiple Correspondence Analysis (MCA) was used as a preprocessing stage, followed by Hierarchical Clustering on Principle Components (HCPC) to determine clustering of the tools based on the features. Both of these came from the `FactoMineR` [52] package. HCPC generates a tree structure that can be 'cut' to determine the number of clusters. By default, HCPC requires the number of clusters to be specified by the user. However, it is possible to request HCPC to suggest a best-case number of clusters calculated by inertia gain[3] as the number of clusters decreases. In this case, I used the number of clusters that was suggested by the algorithm, which was four.

A planar projection of the endpoints of the 3D tree generated by the HCPC algorithm can be seen in Figure 7.2a. This figure visualizes the clusters by color. Note that distance in two dimensions does not necessarily correlate to similarity as

[3]Inertia is a measure of variation used to determine when it is no longer beneficial to add clusters.



(a) Planar projection of the 3D HCPC clustering algorithm tree.



(b) Dendrogram of HCPC result clusters 1 (Red), 2 (Blue), 3 (Green), 4 (Yellow).

Fig. 7.2 HCPC tree projection and dendrogram.

Cluster	Factor	Overall %	Cluster %
1	Using XML	100%	53%
	Not being able to preview	90%	60%
	Not having documentation	85%	73%
	Editing using Inspectors	81%	87%
	Not editing using Inline	74%	93%
	Not having a launcher	71%	100%
	Not being able to export	68%	87%
	Not having a custom format	68%	87%
	Not having a web app	67%	93%
	No syntax highlighting	64%	93%
2	Being able to park links	100%	100%
	Can autolayout content	66%	100%
	Can autocomplete	66%	100%
3	Having HTML format	100%	30%
	Web application	88%	70%
	Not having a content browser	75%	60%
	Having a launcher	75%	60%
	Editing using Inline	70%	90%
	Not editing with Inspectors	69%	90%
	Not having a dynamic node view	50%	90%
4	Being able to export to runtime	100%	100%
	Having error handling at Build time	66%	100%
	Being able to make changes during debug	40%	100%

Table 7.1 Simplified factors for each cluster. Only key variables are listed.

the points are the projected result of a branching hierarchical tree in three dimensions. However, the dendrogram depicted in Figure 7.2b is a 2D representation of such a tree and we can use the branches of this to infer similarity and divergence. The dendrogram's height value (y-axis) can be used to determine the potential result of increasing or decreasing the requested cluster count. For instance, increasing to five clusters would split the rightmost group unevenly in two. However, as the inertia gain generated by HCPC suggested four clusters as optimal, this was not changed.

A simplified listing of the factors contributing to each cluster are listed in Table 7.1. In this table, 'Overall %' refers to the total percentage of all tools that have this feature that are within a given cluster. If the overall percentage is 100, then every tool containing that factor exists within the given cluster and that feature is not present in any other clusters. The 'Cluster %' refers to the total percentage of tools within a given cluster having a specific factor. Not all members of a cluster will have the exact same properties. If the cluster percentage is 50, then only half of the tools contained within the cluster have a given factor. Closer inspection of the HCPC variables helps to identify the key factors that lead to the computed clustering of these tools. The archetypical characteristics of each cluster are as follows:

- **Cluster 1**: Inspector editing, lack of examples or documentation, XML based, not web-based, unable to export.
- **Cluster 2**: Link parking, autolayout, autocomplete.
- **Cluster 3**: Web apps, launcher present, non-inspector editing (mostly inline), examples and documentation.
- **Cluster 4**: Runtime export, debug editing, build time error handling.

The first cluster is composed almost exclusively of academic tools. More than half of these tools use XML, and most are not able to preview. There is also a lack of documentation and any form of launcher. Editing is mostly done through inspectors. There is a strong bias towards non-web applications. This reveals important information about the academic tools in that they are poorly supported (i.e., no user documentation) and do not provide much in the way of a complete editing experience. This makes sense since research projects more often than not tackle a given part of a field rather than attempting to create a whole package.

In contrast, the third cluster contains mostly community-driven projects. A majority of these tools have web applications paired with some kind of launcher. Editing is overwhelmingly done using inline fields although there is a lack of content browsers. Node views used for editing are also not as popular in this category. These tools are mostly modern and seem to favor web applications, which given the rise in popularity of web applications in recent years makes sense. Inline content is heavily favored which suggests that web interfaces could perhaps be better suited to editing using fields rather than inspectors.

The second cluster consists of two similar products from the same company. These two are highly supported commercial tools with a particular editing style based on hypertext. It is not a coincidence that they would form their own cluster.

The fourth cluster, on the other hand, is the opposite. This cluster is made up of two dramatically different systems. However, they were clustered based on some limited similarity that no other tools share like being able to export to existing runtimes, having strong debugging abilities, and presenting errors to users at build time.

Conclusion

We must consider the side effects of small data sets when interpreting these results. In particular that the number of tools is too few to generate statistically significant data. Having only 29 entries can result in supposed correlations being a result of noise whereas with more samples this would gradually be reduced. Unfortunately, the number of specialized interactive narrative authoring tools is not something that can be controlled. Even with this limitation, the data generated clusters that make sense. For the purposes of this experiment - detecting clusters to determine a representative sample of tools - the results are sufficient. A more detailed and in-depth study of a broader range of tools is a possible solution, although this may introduce false positives, as this experiment focused on tools dedicated for interactive narrative.

7.4.3 | Representative Tasks

Since this experiment is chiefly focusing on authoring of video game narrative, it makes sense to provide a set of tasks to the participants that mimic contemporary video game writing styles. Since there are countless variations in video game writing, we need a representative sample of these tasks, just as with the tools. To generate such a task list, feature analysis of a set of games was done to determine a set of common employed storytelling features. The frequency of occurrence across the sampled games can then be used to influence the task design.

Features

In total, 17 games were included, with as much variation of genre as possible. They were then played through and analyzed to determine high-level narrative features. Any features that were overly or under expressed were excluded. Below is a list of all included features and their descriptions:

- **Primary Plot Progression** (`Task`/`Interaction`)
  The main method to progress story. Sets of tasks may gate progress, holding the player back until completed. Otherwise, simple interactions may progress the story.
- **Narrator Type** (`None`/`Dramatized`/`Undramatized`)
  How dramatized the narrators are, based on Booth [15].
- **Narrator Significance** (`None`/`Substantial`/`Insubstantial`)
  Whether the narrator plays a key role in the story or not.
- **Cutscene** (`Yes`/`No`)
  Does the game have cutscenes?
- **Interruptible Cutscene** (`Yes`/`No`)
  Does the game have cutscenes that can be interrupted or influenced?
- **Movie** (`Yes`/`No`)
  Does the game feature cinematic sequences?
- **In-Game Event** (`Yes`/`No`)
  Does the game break regular behavior of non-playable characters or objects for the purpose of narrative.
- **Environmental Storytelling** (`Yes`/`No`)
  Does the game feature environmental storytelling?
- **Choices** (`None`/`Diegetic`/`Extra-Diegetic`)
  Are choices diegetic (happen in the moment without suspension of story flow) or extra-diegetic (suspension of flow while choice is made)?

- **Mechanics as Metaphor** (`Yes`/`No`)
  Does the game use mechanics as a metaphor?
- **Final Resolution** (`Linear`/`Branching`/`Perpetual`)
  How does the story end? Is it linear, branching, or does it not end?
- **Player Traits** (`Yes`/`No`)
  Do player traits alter the narrative, such as morality points affecting which conversation options are available?
- **INEA** (`Yes`/`No`)
  Does the game feature Intangible/Narrative/Explicit/Active elements of Discoverable Narrative?
- **TNEA** (`Yes`/`No`)
  Does the game feature Tangible/Narrative/Explicit/Active elements of Discoverable Narrative?

Analysis

R was used to conduct a simple frequency analysis of the individual features over all games. Results can be seen in Figure 7.3. It is clear from this graph that some features, such as perpetual endings, are insignificant within the corpus analyzed.

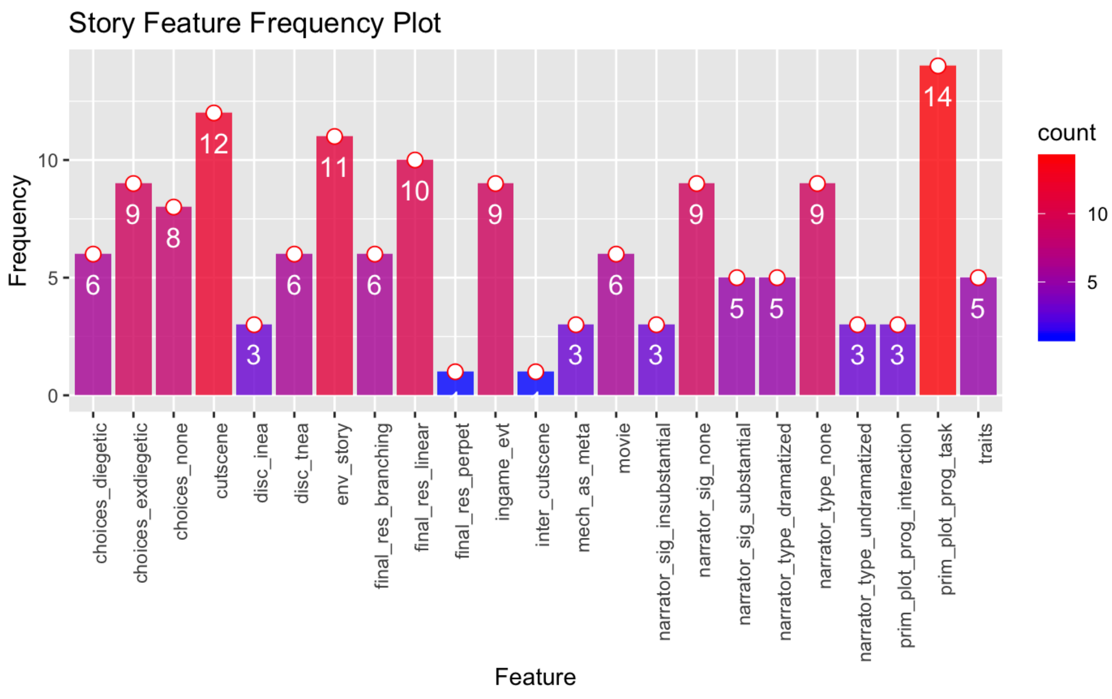


Fig. 7.3 Game story features frequency analysis.

7.4.4 | Methodology

Now that I have declared the methods for choosing representative tools and discovered the frequency of a set of game writing tasks, they must be used within an experiment that answers the research questions outlined at the beginning of this chapter.

Overview

To answer the proposed research questions, participants must use tools under typical writing circumstances and have data about the usage gathered and analyzed. The tools that participants will be using in the experiment are taken from the clusters identified in the earlier discussed analysis. Three of these tools will be included in the experiment: *Quest*, *Inform 7*, and *articy:draft 3*. The first two of these tools come from the two largest clusters in the analysis, and the third comes from one of the smaller clusters. Each of these programs present variety in their UX approaches. *Quest* uses multifaceted controls, *Inform 7* is largely based on scripting with an augmented text editor, and *articy:draft 3* primarily uses a dynamic node graph. This variation will ensure that we can test as many of the primary UI paradigms. From their individual clusters, *Quest* was chosen due to its availability and its difference from the other systems, *Inform 7* was chosen due to its popularity[4], and *articy:draft 3* was chosen due to its popularity and prominence in games development. Participants will use these tools to implement a set of representative tasks with respect to video game narrative authoring.

Participants

The target demographic for this experiment is students of digital narrative who have an interest in creative writing, particularly for video games. Participants are not expected to be experts in writing, but it is expected that they are competent. Participants will mostly be sourced from related courses in the university, but others that meet the criteria and have no conflict of interest are also welcome.

An incentive is offered to encourage participation and to reward participants for their contribution. Any participant that completes the experiment can be optionally entered into a prize draw, where they have the chance to win one of three £30 cash prizes. The draw will be done randomly after the experiment's data gathering phase has finished. The draw is entirely randomized and the performance of each participant is not taken into account.

Training

To ensure that all participants have at least a base level of knowledge, brief training videos will be presented to them prior to the experiment proper. These videos cover the essential topics necessary to operate the programs and teaches the essential skills required to fulfill the tasks. Care has been taken to ensure that the videos do not in any way disclose the actual content of the experimental tasks they will perform. Accompanying sample files from the training videos are provided to participants for them to explore.

[4]As of writing, IFDB (www.ifdb.tads.org) shows over 3000 stories written using *Inform*.

Writing Tasks

The tasks that the participants will be asked to complete have been formulated into an interactive version of Little Red Riding Hood (LRRH). This fairy tale has been broken up into an introduction and three distinct chapters to aid the challenge that participants face, which will be discussed shortly. In the introduction, LRRH is set on her way by her mother. In the first chapter, LRRH makes her way through the woodland; this happens prior to her encounter with the wolf. In the second chapter, LRRH is confronted by the wolf and then continues to her grandmother's cottage. In the third chapter, the wolf had already arrived and attempts to eat LRRH. A complete script of this story can be found in Appendix B. Most story features from the previous analysis are included in one of the three chapters, implemented as so:

- **Chapter 1** - The Woodland

  Environmental Storytelling
  Describe the environment surrounding LRRH as she walks into the forest.

  In-Game Event
  Include a sequence where LRRH spots a shadow of the wolf in the distance prior to meeting him.

  Undramatized Narrator/Progress via Interaction
  Have an undramatized narrator dictate LRRH's adventure. Additionally, make their narration trigger only once LRRH has reached milestones within the woodland.

  Mechanics as Metaphor
  Have LRRH slow down from her initial walking speed as she gets frightened in the forest (such as after spotting the wolf's shadow).

- **Chapter 2** - The Meeting

  Extra-Diegetic Choice
  Have the wolf offer LRRH a quest, such as gathering a number of type of flowers, that she can accept or decline.

  Progress via Task
  LRRH cannot continue until she has found and collected a number of items. If accepting the wolf's quest, the items could be those that he suggests.

  Discoverable Narrative (TNEA)
  Add an optionally discoverable letter than LRRH can find (perhaps from another soul warning about the wolf's true nature).

  Dramatized Narrator/Player Traits Alter Narrative
  Have LRRH self-narrate her walking through the woods after meeting the wolf. Additionally, have her dialogue reflect on her choice with the during the wolf's confrontation.

- **Chapter 3** - The Battle

  Cutscene/Interruptible Cutscene

  Describe a cutscene where grandma is swallowed, and the sequence following until LRRH is also swallowed, leading to their demise. Alternatively, modify the sequence so that LRRH can interrupt and scream for help, triggering the lumberjack to arrive and save the day.

  Linear Resolve

  Have the story end in a single way with either the wolf being victorious or the lumberjack saving the day, without any form of choice.

  Branching Resolve

  Modify the story linear ending so that both options can be taken.

  Diegetic Choice

  In a resolve where the wolf does not die at the hand of the lumberjack, provide the option to kill or spare the wolf.

The participants receive the complete implemented story file for their assigned tool. However, either Chapter 2 or 3 are missing. The challenge to the participants is to then complete the missing part of the story by completing the appropriate tasks for the given section. Chapter 2 and 3 were chosen as they are non-linear and allow for more detailed use of program functionality, whereas Chapter 1 is comparatively linear and may produce significantly less interesting results. The participant is provided with the complete script to copy from if they would like assistance or struggle to come up with creative writing in such a short time, but they are not limited to the script as long as their implementation matches the features required. Participants will be also asked to think aloud during this phase which will be recorded.

Participant Flow

A single session takes around one hour in total. Each participant is randomly distributed one of the three tools, with care taken to ensure that the distribution is as uniform as possible. In the beginning, the participant is given a documentation page which contains information about the experiment, a listing of descriptions of all of the tasks, and a copy of the script broken up into sections and characters for them to use in the experiment. In the first 15 minutes, the participant watches the corresponding training video and is given the opportunity to briefly explore the program along with the provided sample file used in the video to familiarize themselves with the program prior to their analyzed usage. Following training, the participant is given a hard limit of 30 minutes to complete their tasks in their assigned tool, as described above. Finally, the participant is interviewed for around 15 minutes.

Data Gathering & Data Usage

Data will be gathered from a number of sources to maximize output of the experiment. Each participant will be assigned a unique number which will be used to collate all data gathered about them during the experiment and serve as a way of retaining this collation once anonymization has taken place.

Participants' screens will be recorded for the duration of their task implementation. The purpose of this recording is to calculate experiment metrics (how many tasks were attempted, how many tasks were successful, how long each task took) and to analyze the behavior when using the program (aiding in identifying sources of frustration or struggle).

Additionally, a webcam (or free-standing camera) will be used to capture video of the participant during their task implementation. This footage is used to cross-reference against the screen recordings to help triangulate sources of frustration and to gauge emotion (frustrated, happy, etc.) while using the program. This is particularly important for identifying difficulties that may not show from a screen recording alone. Audio is also recorded and used in this process, as the participants are asked to think out loud while using the program.

Observational notes are taken by the researcher while the participant implements their tasks as to their responses and behavior while using the program. This is used to further assist in triangulating sources of difficulty. Care must be taken to ensure the participant is not disturbed during the experiment, and therefore the researcher must distance themselves and make sure not to interrupt the participant.

An interview will be conducted with the participant after they have completed the task implementation portion of the experiment. It is expected to take around 15 minutes or less, depending on the length and detail of the answers given. The questions aim to collect mostly qualitative data about the participant's experience with the tool, as more quantitative data is gathered from other sources.

The first set of questions in the interview aims to determine groupings of participants and potential sources of bias that could skew results.

- **What is your profession or field of study?**
  Differentiate between disciplines (student or otherwise).
- **What was your previous knowledge of the tool before this experiment (none, a little, familiar)?**
  Gauge prior exposure to the specific tool that is being tested.
- **What is your previous knowledge of any other similar tools before this experiment (none, a little, familiar)?**
  Gauge prior exposure to similar tools (for example, somebody with experience in *Twine* may be more suited to using *articy:draft 3* over somebody who has only used a word processing package).
- **What was your previous knowledge of interactive digital narrative before this experiment (none, a little, familiar)?**

Determine the general knowledge of interactive digital narrative prior to the experiment.

The second set of questions aims to gather details about the participants' opinions about the tool and their experience with it. These questions directly correspond with answering the earlier proposed research questions of this experiment.

- **Tell me about how you felt the features in the tool were a hindrance to your goal of implementing your vision.**
  What elements of the tool caused a negative experience for this participant?
- **Tell me about how you felt the features in the tool aided your goal of implementing your vision.**
  What elements of the tool caused a positive experience for this participant?
- **If you could add any features to make implementing your vision easier, what would they be and why?**
  Discover what elements the participant would like to add that were not present to their make job easier.
- **If you could alter any existing features to make implementing your vision easier, what would they be and why?**
  Discover what already existing elements in the tool participants would change, and how, to make their job easier.
- **How would you have completed or improved your implemented solution given more time?**
  Determine how the participant would have continued refining or completing their solution in the tool if given more time.

All information collected about participants will be kept strictly in accordance with the current Data Protection Regulations. Data will be anonymized as soon as possible and no published data will be in any way able to identify participants. Once participant data has been analyzed, the identifying camera footage will be deleted. Screen recordings are not identifiable and therefore are not required to be anonymized or deleted until the project is complete. Audio recordings made during interviews will be deleted as soon as transcriptions are made and any phrases spoken that could identify a participant will be anonymized during transcription. Participants who wish to enter into the prize draw will have their email stored separately from the rest of the data. This email will only be accessible to the lead researcher and will be deleted as soon as the prize draw concludes. The emails will only be used for the sake of contacting the winners of the prize draw and no further contact will be made as a result of this experiment.

References

[1] Aarne, A. (1961). *The Types of the Folktale: A Classification and Bibliography*. Suomalainen Tiedeakatemia.

[2] Aarseth, E. (2012). A Narrative Theory of Games. In *Proceedings of the International Conference on the Foundations of Digital Games*, FDG '12, pages 129–133, New York, NY, USA. ACM.

[3] Adams, E. (2014). *Fundamentals of Game Design*. New Riders Publishing, Thousand Oaks, CA, USA, 3rd edition.

[4] Andrews, M. (2010). Game UI Discoveries: What Players Want. *Gamasutra*. http://www.gamasutra.com/view/feature/4286/game_ui_discoveries_what_players_.php.

[5] Avedon, E. (1981). The Structural Elements of Games. In *The Psychology of Social Situations: Selected Readings*, pages 11–17. John Wiley & Sons, Inc, New York.

[6] Bargas-Avila, J. A. and Hornbæk, K. (2011). Old Wine in New Bottles or Novel Challenges: A Critical Analysis of Empirical Studies of User Experience. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 2689–2698, New York, NY, USA. ACM.

[7] Barthes, R. and Duisit, L. (1975). An Introduction to the Structural Analysis of Narrative. *New Literary History*, 6(2):237–272.

[8] Berners-Lee, T. and Connolly, D. (1995). Hypertext Markup Language - 2.0. Technical report. http://www.rfc-editor.org/rfc/rfc1866.txt.

[9] Bernstein, M. (1998). Patterns of Hypertext. In *Proceedings of the Ninth ACM Conference on Hypertext and Hypermedia : Links, Objects, Time and Space-structure in Hypermedia Systems: Links, Objects, Time and Space-structure in Hypermedia Systems*, HYPERTEXT '98, pages 21–29, New York, NY, USA. ACM.

[10] Bernstein, M. (2001). Card Shark and Thespis: Exotic Tools for Hypertext Narrative. In *Proceedings of the 12th ACM Conference on Hypertext and Hypermedia*, HYPERTEXT '01, pages 41–50, New York, NY, USA. ACM.

[11] Bernstein, M. (2002). Storyspace 1. In *Proceedings of the Thirteenth ACM Conference on Hypertext and Hypermedia*, HYPERTEXT '02, pages 172–181, New York, NY, USA. ACM.

[12] Bizzochi, J. (2007). Games and Narrative: An Analytical Framework. *Loading - The Journal of the Canadian Games Studies Association*, 1(1):5–10.

[13] Björk, S. and Holopainen, J. (2003). Describing Games - An Interaction-Centric Structural Framework. In *Proceedings of the 2003 DiGRA International Conference: Level Up*, volume 2, pages 4–6.

[14] Bolter, J. D. and Joyce, M. (1987). Hypertext and Creative Writing. In *Proceedings of the ACM Conference on Hypertext*, HYPERTEXT ’87, pages 41–50, New York, NY, USA. ACM.

[15] Booth, W. C. (1961). *The Rhetoric of Fiction.* University of Chicago Press.

[16] Bostan, B. and Turan, O. (2017). Deconstructing Game Stories with Propp’s Morphology. volume 17 of *system*, page 18, Bahcesehir University, İstanbul, TURKEY.

[17] Brand, J. E. and Knight, S. J. (2005). The Narrative and Ludic Nexus in Computer Games: Diverse Worlds II. In *Proceedings of the 2005 DiGRA International Conference: Changing Views: Worlds in Play*, volume 3.

[18] Bruner, J. (1991). The Narrative Construction of Reality. *Critical Inquiry*, 18(1):1–21.

[19] Brusentsev, A., Hitchens, M., and Richards, D. (2012). An Investigation of Vladimir Propp’s 31 Functions and 8 Broad Character Types and How They Apply to the Analysis of Video Games. In *Proceedings of The 8th Australasian Conference on Interactive Entertainment: Playing the System*, IE ’12, pages 2:1–2:10, New York, NY, USA. ACM.

[20] Campbell, J. (2008). *The Hero with a Thousand Faces.* New World Library.

[21] Carson, D. (2000). Environmental Storytelling: Creating Immersive 3D Worlds Using Lessons Learned from the Theme Park Industry. *Gamasutra.* http://www.gamasutra.com/view/feature/3186/environmental_storytelling_.php.

[22] Cheong, Y.-G. and Young, R. M. (2006). A Framework for Summarizing Game Experiences As Narratives. In *Proceedings of the Second AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, AIIDE’06, pages 106–108, Marina del Rey, California. AAAI Press.

[23] Compton, K., Kybartas, B., and Mateas, M. (2015). Tracery: An Author-Focused Generative Text Tool. In *Interactive Storytelling*, Lecture Notes in Computer Science, pages 154–161. Springer, Cham.

[24] Delmas, G., Champagnat, R., and Augeraud, M. (2007). Bringing Interactivity into Campbell’s Hero’s Journey. In *Proceedings of the 4th International Conference on Virtual Storytelling: Using Virtual Reality Technologies for Storytelling*, ICVS’07, pages 187–195, Berlin, Heidelberg. Springer-Verlag.

[25] Dixon, D. (2013). *Goal, Motivation and Conflict: The Building Blocks of Good Fiction.* Bell Bridge Books.

[26] Ebbinghaus, H. (1913). *Memory: A Contribution to Experimental Psychology.* Teachers College, Columbia University, New York. (Original, Über das Gedächtnis, Leipzig, 1885).

[27] Eskelinen, M. (2001). The Gaming Situation. *Game studies*, 1:68.

[28] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T. (1999). Hypertext Transfer Protocol – HTTP/1.1. Technical report. http://www.rfc-editor.org/info/rfc2616.

[29] Fitts, P. M. (1954). The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement. *Journal of Experimental Psychology*, 47(6):381–391.

[30] Frasca, G. (2003). Ludologists Love Stories, Too: Notes From a Debate That Never Took Place. In *Proceedings of the 2003 DiGRA International Conference.*

[31] Gervás, P. (2013). Propp's Morphology of the Folk Tale as a Grammar for Generation. In *2013 Workshop on Computational Models of Narrative*, volume 32 of *OpenAccess Series in Informatics (OASIcs)*, pages 106–122, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[32] Göbel, S., Salvatore, L., and Konrad, R. (2008). StoryTec: A Digital Storytelling Platform for the Authoring and Experiencing of Interactive and Non-Linear Stories. In *2008 International Conference on Automated Solutions for Cross Media Content and Multi-Channel Distribution*, pages 103–110, Florence, Italy. IEEE.

[33] Green, D., Hargood, C., Charles, F., and Jones, A. (2018). Novella: A Proposition for Game-Based Storytelling. In *Narrative and Hypertext 2018*. ACM.

[34] Halasz, F. and Schwartz, M. (1994). The Dexter Hypertext Reference Model. *Commun. ACM*, 37(2):30–39.

[35] Hanson, B. L. (1983-07). Human factors and behavioral science: A brief history of applied behavioral science at Bell Laboratories. *The Bell System Technical Journal*, 62(6):1571–1590.

[36] Hargood, C., Weal, M. J., and Millard, D. E. (2018). The Storyplaces Platform: Building a Web-Based Locative Hypertext System. In *Proceedings of the 29th ACM Conference on Hypertext and Social Media*, HT '18, New York, NY, USA. ACM.

[37] Hick, W. E. (1952). On the Rate of Gain of Information. *Quarterly Journal of Experimental Psychology*, 4(1):11–26.

[38] Hyman, R. (1953). Stimulus Information as a Determinant of Reaction Time. *Journal of Experimental Psychology*, 45(3):188–196.

[39] Jenkins, H. (2004). Game Design as Narrative Architecture. *First Person. New Media as Story, Performance, and Game (eds.) Pat Harrigan and Noah Wardrip-Fruin*, 44:53.

[40] Juul, J. (2001). Games Telling Stories? A Brief Note on Games and Narratives. *Game studies*, 1:1–12.

[41] Kapadia, M., Falk, J., Zünd, F., Marti, M., Sumner, R. W., and Gross, M. (2015). Computer-assisted Authoring of Interactive Narratives. In *Proceedings of the 19th Symposium on Interactive 3D Graphics and Games*, i3D '15, pages 85–92, New York, NY, USA. ACM.

[42] Kapadia, M., Frey, S., Shoulson, A., Sumner, R. W., and Gross, M. (2016a). CANVAS: Computer-assisted Narrative Animation Synthesis. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '16, pages 199–209, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

[43] Kapadia, M., Poulakos, S., Gross, M., and Sumner, R. W. (2017). Computational Narrative. In *ACM SIGGRAPH 2017 Courses*, SIGGRAPH '17, pages 4:1–4:118, New York, NY, USA. ACM.

[44] Kapadia, M., Shoulson, A., Steimer, C., Oberholzer, S., Sumner, R. W., and Gross, M. (2016b). An Event-centric Approach to Authoring Stories in Crowds. In *Proceedings of the 9th International Conference on Motion in Games*, MIG '16, pages 15–24, New York, NY, USA. ACM.

[45] Koenitz, H. (2011). Extensible Tools for Practical Experiments in IDN: The Advanced Stories Authoring and Presentation System. In *Proceedings of the 4th International Conference on Interactive Digital Storytelling*, ICIDS'11, pages 79–84, Berlin, Heidelberg. Springer.

[46] Koffka, K. (1935). *Principles of Gestalt Psychology.* Harcourt, Brace, New York.

[47] Köhler, W. (1967). Gestalt Psychology. *Psychologische Forschung*, 31(1):XVIII–XXX.

[48] Kybartas, B. and Bidarra, R. (2017). A Survey on Story Generation Techniques for Authoring Computational Narratives. *IEEE Transactions on Computational Intelligence and AI in Games*, 9(3):239–253.

[49] Lakoff, G. (1972). *Structural Complexity in Fairy Tales.* UC Berkeley.

[50] Lankoski, P. and Björk, S. (2015). Formal Analysis of Gameplay. In *Game Research Methods*, pages 23–35. ETC Press, Pittsburgh, PA, USA.

[51] Law, E., Roto, V., Vermeeren, A. P., Kort, J., and Hassenzahl, M. (2008). Towards a Shared Definition of User Experience. In *CHI '08 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '08, pages 2395–2398, New York, NY, USA. ACM.

[52] Lê, S., Josse, J., Husson, F., et al. (2008). FactoMineR: An R Package for Multivariate Analysis. *Journal of Statistical Software*, 25(1):1–18.

[53] Malec, S. (2001). Proppian Structural Analysis and XML Modeling. In *Computers, Literature and Philology (CLiP 2001).*

[54] Mason, S. (2013). On Games and Links: Extending the Vocabulary of Agency and Immersion in Interactive Narratives. In *Interactive Storytelling*, Lecture Notes in Computer Science, pages 25–34. Springer International Publishing.

[55] Metzger, W. (2006). *Laws of seeing.* MIT Press, Cambridge, MA, US.

[56] Miller, G. A. (1956). The Magical Number Seven, Plus or Minus Two: Some Limits on our Capacity for Processing Information. *Psychological Review*, 63(2):81–97.

[57] Mirnig, A. G., Meschtscherjakov, A., Wurhofer, D., Meneweger, T., and Tscheligi, M. (2015). A Formal Analysis of the ISO 9241-210 Definition of User Experience. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '15, pages 437–450, New York, NY, USA. ACM.

[58] Nelson, T. H. (1965). Complex Information Processing: A File Structure for the Complex, the Changing and the Indeterminate. In *Proceedings of the 1965 20th National Conference*, ACM '65, pages 84–100, New York, NY, USA. ACM.

[59] Nielsen, J. (1994). Enhancing the Explanatory Power of Usability Heuristics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '94, pages 152–158, New York, NY, USA. ACM.

[60] Norman, D. (1996). Design as Practiced. In Winograd, T., editor, *Bringing Design to Software*, pages 233–251. ACM, New York, NY, USA.

[61] Packard, E. (1979). *Choose Your Own Adventure #1: The Cave of Time.* Bantam Books.

[62] Poulakos, S., Kapadia, M., Maiga, G. M., Zünd, F., Gross, M., and Sumner, R. W. (2016). Evaluating Accessible Graphical Interfaces for Building Story Worlds. In *Interactive Storytelling*, Lecture Notes in Computer Science, pages 184–196. Springer, Cham.

[63] Poulakos, S., Kapadia, M., Schüpfer, A., Zünd, F., Sumner, R. W., and Gross, M. (2015). Towards an Accessible Interface for Story World Building. In *Eleventh Artificial Intelligence and Interactive Digital Entertainment Conference*.

[64] Propp, V. (2010). *Morphology of the Folktale: Second Edition*. University of Texas Press.

[65] Ryan, J. (2017). Grimes' Fairy Tales: A 1960s Story Generator. In *Interactive Storytelling*, Lecture Notes in Computer Science, pages 89–103. Springer, Cham.

[66] Shneiderman, B., Plaisant, C., Cohen, M., Jacobs, S., Elmqvist, N., and Diakopoulos, N. (2016). *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Pearson, 6th edition.

[67] Smith, T. and Bernhardt, S. (1988). Expectations and Experiences with HyperCard: A Pilot Study. In *Proceedings of the 6th Annual International Conference on Systems Documentation*, SIGDOC '88, pages 47–56, New York, NY, USA. ACM.

[68] Sørensen, N. and Pødenphant, M. (2013). Narratification: Unifying Narrative and Gameplay.

[69] Uther, H. (2004). *The Types of International Folktales: A Classification and Bibliography. Animal Tales, Tales of Magic, Religious Tales, and Realistic Tales, With an Introduction*, volume 1. Academia scientiarum Fennica.

[70] Walker, J. (1999). Piecing Together and Tearing Apart: Finding the Story in Afternoon. In *Proceedings of the Tenth ACM Conference on Hypertext and Hypermedia : Returning to Our Diverse Roots: Returning to Our Diverse Roots*, HYPERTEXT '99, pages 111–117, New York, NY, USA. ACM.

[71] Weir, G. (2008). Opinion: Grim Fandango And Diegesis In Games. *Gamasutra*. http://www.gamasutra.com/view/news/112122/Opinion_Grim_Fandango_And_Diegesis_In_Games.php.

[72] Wertheimer, M. (1938). Laws of Organization in Perceptual Forms. In Ellis, W., editor, *A Source Book of Gestalt Psychology*, pages 71–88. Routledge & Kegan Paul, London.

[73] Zagal, J. P., Mateas, M., Fernández-Vara, C., Hochhalter, B., and Lichti, N. (2005). Towards an Ontological Language for Game Analysis. In *Proceedings of the 2005 DiGRA International Conference: Changing Views: Worlds in Play*, volume 3.

[74] Zünd, F., Poulakos, S., Kapadia, M., and Sumner, R. W. (2017). Story Version Control and Graphical Visualization for Collaborative Story Authoring. In *Proceedings of the 14th European Conference on Visual Media Production (CVMP 2017)*, CVMP 2017, pages 10:1–10:10, New York, NY, USA. ACM.

This page is intentionally left blank.

Appendix A

Video Game Narrative Analysis

This appendix contains supplementary notes to my application of various models to video game narrative. They are based on my own experience with the models. While they are not core to this thesis, it is important that the methods used to determine usefulness is demonstrated in full. It is divided first by theory and then by game.

Propp's Morphology

Portal

Game Structure. *Portal* follows a linear narrative. The game officially lists 11 chapters, but they are not ideal as the final chapter contains almost half of the game. Instead, we can clearly divide the game into two acts. The first act is comprised of 19 "test chambers" spanning the original 11 chapters where the player is relentlessly tested. The second act continues after the player's near-death experience following the test chambers and can be subdivided into three chapters: the initial escape, the villain trying to kill the player, and the final confrontation. The narrative is delivered mostly through the environment coupled with narration from the main antagonist, GLaDOS. In the first act, she appears prerecorded and robotic, but in the second act, she is revealed to be a true AI with malicious intent and communicates as such. Story is only progressed as the player completes linear segments of gameplay, or discovers hidden areas.

Character Archetypes. Table A.1 displays all characters along with their broad in-game roles and Proppian character archetypes. The player controls Chell, who is the Hero. GLaDOS is primarily a Villain, although for most of the first act she is largely a Helper to the Hero. Doug Rattmann is never seen in the game (but is present in prequel comics) but his legacy remains in the various hidden chambers, warning the Hero of the Villain's true nature, thus being a Helper to the Hero regardless of not being present. The Companion Cube, while inanimate, is personified by GLaDOS as a friend and assists the Hero in completing tests, therefore making it a possible Helper. The Party Escort Bot and the Turrets mostly act as henchmen to the Villain but don't have a suitable archetype. purpose. Scientists are only suggested at in the narrative (but are present in prequel comics) and play no role other than narrative background. the Personality Cores of GLaDOS, like the Companion Cube, are personified and have unique traits, but contribute no more than narrative coherence and as a gameplay mechanic.

Name	Role	Archetype
Chell	Silent protagonist, Player	Hero
GLaDOS	Mentor, Tutor, Antagonist	Helper turn Villain
Doug Rattmann	Leaves messages for Chell	Helper (in spirit)
Companion Cube	Inanimate personalized ally	Helper (inanimate)
Party Escort Bot	Drags Chell back to facility	n/a
Turrets	Enemies	Villain's henchmen
Scientists	Disappeared; only referenced	n/a
Core 1 (curiosity)	n/a	n/a
Core 2 (cake recipe)	n/a	n/a
Core 3 (angry)	n/a	n/a

Table A.1 Proppian character archetypes of *Portal*.

Overall Story Arc. In the first act, the primary functions are *Initial Setup*, *Acquisition* of the portal gun and Companion Cube, and then *Struggle* as the player is relentlessly tested and unknowingly manipulated by the villain. The second act contains numerous instances of *Trickery* and *Reconnaissance* as the villain reveals their true nature and attempts to capture the player, followed by *Confrontation*, *Struggle*, and *Victory* as the game ultimately leads to a face-off between the player and the villain.

Level Narrative. Figure 4.1 shows the broad plot summary of the chapters of both acts. In the individual chambers, there would be a lot of repeated *Testing/Donor* and *Reaction* functions due to the puzzle-reward nature of the game. In Chamber 13 (chapter 5) the player is able to fail the test and become stuck, at which point GLaDOS helps the player bypass the test; this is a form of *Outsider Help.* In chapters 8 through 11 there are discoverable 'Rattmann Dens' previously inhabited by Doug Rattmann which not only serve as interpretative background narrative, but also provide *Outsider Help* to the player by warning them of the true deceiving nature of GLaDOS that up until this point they suppose as a friend.

Player Decisions. There is no decision the player can make that alters the narrative pathway in any significant way.

Dialogue and Interaction. Dialogue is always with GLaDOS, although it is one-sided as the player is mute. While the dialogue appears to react to the player, it is predetermined. The player has no way to influence the dialogue other than meeting certain predefined criteria within the game world which then activates a deterministic branch of dialogue that's otherwise unspoken (for instance, knocking a security camera off a wall prompts a dialogue sequence about not destroying facility

equipment).
Morality System. *Portal* has no morality system.

The Stanley Parable

Game Structure. *The Stanley Parable* has a complex and interwoven branching narrative by design. While the game's structure can be graphed, it is designed to allow for arbitrary player choice. The narrative is conveyed at times via cutscenes, but the primary delivery method is through narration of the (human) player and (in-game) avatar's actions. There are two narrators in the game. The primary narrator comments to himself, about Stanley and NPCs, about the player, and about people in the same room as the player. The secondary narrator talks to the player about Stanley and the primary narrator. Narrative is otherwise provided through the environment, having notable changes in the world occur when events transpire. There are a fixed number of predefined endings, although there is often more than one way to reach the same ending. When the game is broken down, it is clear that there is no single story, but instead an interwoven network of individual stories that have their own ending, where elements are shared until branching points.
Character Archetypes. Table A.2 displays all characters along with their broad in-game roles and Proppian character archetypes. The human player controls a silent protagonist named Stanley, who is the Hero. During the Confusion ending, Stanley can be viewed from 3rd person (meaning the 'Hero' all along in this particular branch was actually the human player and not Stanley), and labeling this is not possible with Propp's archetypes. The primary narrator acts mostly as a Helper throughout the game, but in some branches, becomes a Villain. The secondary narrator enters only during a single branch but attempts to be a Helper to the player briefly. The human player themselves are referenced in the story, often communicated directly to by the narrator, although they do only act as a controller of the Hero mostly. Similarly, under only one segment, potential human players in the same room as the current player are communicated with as a means to replace the main player. These are obviously not possible to replicate using Propp's archetypes. Stanley's wife is only mentioned by name (and appears in the Phone Call Easter egg), but is only used to entice Stanley into a trap and is never actually in the game. Finally, Mariella is a bystander found in the Insanity ending, watching over Stanley after he had died, but provides no further contribution to the narrative.
Overall Story Arc. As each ending has its own unique plot, there is not an overall arc of the story as a whole. How closely two endings share an overall arc is directly dependent upon their position in the complete story graph partially depicted in Figure 4.2a and Figure 4.2b. For instance, the Freedom and Explosion endings differ only by the press of the button and thus share a majority of the overall arc. Freedom and Divine Art, on the other hand, are only similar up until the first decision the player makes. Additionally, there are some ways of jumping between branches of

Name	Role	Archetype
Stanley	Player character	Hero
Stanley (3rd person)	n/a	n/a
Primary narrator	Friend/foe (branch-based)	Helper/Villain
Secondary narrator	Helper	Helper
Player	Controls the Hero	Hero('s controller)
Potential players	Player replacement	n/a (Potential Hero)
Stanley's wife	Entice Stanley	n/a
Mariella	Bystander	n/a

Table A.2 Proppian character archetypes of *The Stanley Parable*.

the story, such as using the maintenance elevator to get from the Confusion ending pathway to another set of endings. The Setup/Absentation (Stanley's coworkers are missing) is consistent throughout all playthroughs.

Level Narrative. Following any path from this starting point will reveal the Proppian overview of the narrative the player follows. For instance, if the player listens to the narrator at every step, obtaining the 'vanilla' ending, then the functions and decisions would be: *Setup, Lack (8), Interdiction(2), Choose Left Door, Guidance (15), Interdiction (2), Go Upstairs, Enter Secret Door, Reveal(51), Interdiction(2), Enter Mind Control Facility, Reveal(51), Interdiction(2), Turn Off Power, Wedding(31)*. Using this method, we can observe the overall story arc of each individual pathway that the player chooses.

Player Decisions. *The Stanley Parable* is based entirely upon player decision. These decisions can be as simple as triggering specific dialogue or sequences that are not story-critical to explicitly causing a non-reversible branch within the story, locking out otherwise accessible endings.

Dialogue and Interaction. Stanley and the player are unable to speak, and therefore the communication with the narrators is one-way. Instead of dialogue, the player can elicit responses based on their actions (or lack of actions). The player has no influence on the predetermined dialogue other than their choices in the game. The dialogue is sometimes spoken at random from a predetermined list or can change in the same environment based on the state of the story. For instance, there is a random chance at the beginning of the game that a phone in the first room Stanley encounters will ring, and if it does, one of a random selection of calls will play out.

Dialogue and Interaction. *The Stanley Parable* has no explicit morality system other than making the player feel responsible for their actions. This does not affect gameplay, however.

Conclusion

Advantages

Overall Story Arc
Breaking down a story at a high level using Propp's functions is often not too difficult, as long as the function purposes are interpreted and converted into context (such as *Wedding* being updated for non-fairytale purposes). This results in a broad description of what happens throughout the story, retaining an overview, but not the details. This can sometimes be applied at lower levels of granularity, such as individual levels, as long as their arc is broad enough to be encapsulated with meaning. Propp's functions can therefore generally describe the overall structure of a story arc for most scenarios, and in some circumstances, lower granularity such as individual levels.

Disadvantages

Function Level of Detail
While Propp's functions often capture high-level story well, it often fails to apply to individual encounters and situations. Some circumstances fit well, such as *Interdiction* and *Violation of Interdiction* being able to represent the narrator's orders to Stanley and his tendency to violate them. However, more complex situations such as the Broom Closet scenario, which sees interaction with the player's character, the player themselves, other non-players in the room with the player, and even requires restarting the game, all as part of this sequence are often difficult if not impossible to map. This is logical, as the model is not intended to encapsulate every minute interaction. This could be resolved by altering existing functions to contain a wider variety of descriptors to match modern media, or to create new functions entirely, as touched upon by Bostan [16].

Branching Complexity
Propp's functions were never designed to be used in branching narrative. If used with Propp's original restrictions surrounding repetition and order, then this system cannot represent branching at all. If we simply choose to ignore these rules, as Bostan [16] has, and apply functions where they map, then we can use the functions for analysis of story patterns. Brusentsev [19] too found this limitation and proposed a 'Decision Function' that would allow for the original theory to handle the complete flow of branching games, although the suggestion was purely speculative and there seems to have been no further work following this path.

Stochastic and Variable Details
Propp's functions struggle to handle any level of randomness of variability. For example, *The Stanley Parable* sometimes chooses dialogue and events at random (i.e., the same sequence has multiple possible experiences) which could change the interpretation of the narrative. Additionally, sometimes parts of the same game are

modified or shortened entirely to save time with each repetition of the game once initially experienced, making branches even more difficult to map to functions.

Repetition
In *The Stanley Parable*, resetting the current play session is part of the game itself (although certain states are retained), and as such, branches are usually traversed more than one time. Just as Propp's Functions struggle to handle branching, they likewise cannot map to a repeating narrative.

Character Archetypes
The character archetypes of Propp's Morphology, while detailed and numerous, do not cover certain kinds of characters in modern game storytelling. For instance, in *Portal*, Doug Rattmann acts as a Helper for Chell (the Hero) by leaving subtle hidden warning messages about the Villain, but is never directly present in the game itself; he is a helper in spirit, and Propp's archetypes cannot capture this detail. The archetypes also fail to represent characters who transition between two archetypes, such as with GLaDOS initially being a Helper and then changing into a Villain throughout the story. Similarly, in *The Stanley Parable*, the narrator acts as a Helper for most of the game, but only under specific branches does he become a Villain. This level of representation is not possible with Propp's character archetypes.

Aarseth

Portal

World Ludicity. *Portal* combines both ludic and non-ludic spaces. The play area consists largely of enclosed ludic spaces with either visual or suggested non-ludic spaces blocked from player access.
World Structure. *Portal* exhibits a purely linear corridor structure with each level following on from the previous with no way to backtrack or go off the planned path.
Objects. Objects are far too numerous to list completely. Instead, elements showing off each category have been chosen as well as some that stretch the definitions. Cubes, for instance, fit well into Aarseth's *Static usable* listing, but in fact, they are also dynamic in the game. Similarly, the level signs are *Static not usable*, but can change.
Characters. Chell is the player-controlled avatar, has a full backstory and personality, but does not speak throughout the entire game. GLaDOS, the main protagonist, has a rich history and unique personality and interacts directly with the player. Doug Rattmann is present only in spirit via his abandoned hideouts. While he has an individual name, little concrete information is known about him as his narrative is largely interpretative. The Companion Cube is personified but has no personality. The player uses it as an object to solve puzzles. Turrets have no names. They do have personality, but it is equally shared between all turrets. The player can interact with them in combat. The Party Escort Bot drags the player back into the facility

Object	Interactivity	Malleability
Portals	Create, traverse through	Creatable
Portal gun	Creates portals	Changeable
Portalable surfaces	Can have portal on them	Static usable
Non-portalable surfaces	Cannot have portals on them	Static not usable
All other static props	Collide	Static not usable
Dynamic objects	Pick up, sometimes destroy	Static usable
Level signs	Static, but change visually	Static not usable
Cubes	Physics, moveable	Static usable
Buttons (for cubes)	Triggered by weight	Static usable
Fizzlers	Level barriers	Static usable
Turrets	Physics, movable, kills player	Static usable

Table A.3 Select examples of Aarseth's object types for *Portal*.

Name	Malleability
Chell	Deep
GLaDOS	Deep
Doug Rattmann	Shallow
Companion Cube	Shallow
Turrets	Bots
Party Escort Bot	Bots
Scientists	Bots
Core 1 (curiosity)	Shallow
Core 2 (cake recipe)	Shallow
Core 3 (angry)	Shallow

Table A.4 Aarseth's character types for *Portal*.

at the end of the game but has no defined personality. Scientists are a collective mentioned by GLaDOS but do not appear in the game and have no personality beyond motives. All three Personality Cores have unique personality, but are of no real significance other than the ending boss battle.

World Space. *Portal* has fixed kernels and flexible satellites, making it linear.

Variable Model. The game's structure is linear corridor, it contains all kinds of objects except Inventible, it exhibits all character types, and the story is fixed but with flexible choices in how to solve encounters and puzzles.

The Stanley Parable

World Ludicity. *The Stanley Parable* contains both ludic and non-ludic spaces. The majority of the game is playable ludic area. Some areas, such as office spaces with restricted areas, offer clearly-visible spaces to the player that are non-ludic, purposefully limited for gameplay or narrative purposes.

World Structure. The particular path in which the player traverses is not linear due to the organization of kernel events (mostly player choices) in the game. Generally, the player is not able to backtrack except when explicitly allowed for the purpose of narrative.

Objects. Objects are far too numerous to list completely. Instead, elements showing off each category have been chosen as well as some that stretch the definitions. The most common types are static and usable objects, perhaps due to the interactive nature of the game. A number of interactive objects are only possible to interact with once the story sufficiently advances.

Object	Interactivity	Malleability
Locked doors	Only narrator can open	Static not usable
Unlocked doors	Player can open	Static usable
All static props	Colliders; cannot interact	Static not usable
Powered computers	Input code; turn off	Static usable
Lifts	Use (usually automatic)	Static usable
Clickable buttons	Context-specific reaction	Static usable
Phone Plug	Can unplug from wall	Static usable (Dynamic)

Table A.5 Select examples of Aarseth's object types for *The Stanley Parable.*

Characters. While Stanley doesn't speak, he is the main focus of the narrative, and as such is well-rounded and detailed. The player directly controls Stanley most of the time. In one ending, Stanley can be seen from a third-person perspective, but this copy of Stanley is only present in a specific scenario and is not referenced elsewhere. The primary narrator, while nameless and godlike, has strong personality traits, and is clearly an important character. The secondary narrator is only present for a brief period during one specific sequence. The literal human player and people surrounding them are not represented in this model, although they do play a role in the narrative. Stanley's wife is mentioned only by name and exists by voice in an Easter egg during a call to remind Stanley to bring home groceries. Mariella is uniquely named and described but otherwise is a shallow character in one ending.

Name	Malleability
Stanley	Deep
Stanley (3rd person)	Shallow
Primary narrator	Deep
Secondary narrator	Shallow
Human Player	n/a
Other Players	n/a
Stanley's wife	Shallow
Mariella	Shallow

Table A.6 Aarseth's character types for *The Stanley Parable.*

World Space. The game features both dynamic satellites and dynamic kernels. The narrative pathways are technically predefined and linear, but the way in which the player traverses these branches is directly due to the player's choice of narratively-significant branching events.

Variable Model. The game's structure is a linear corridor, although this is a misnomer, as the linear path taken depends upon the player's choice. It contains mostly both static usable and non-usable objects, with interactivity being shared between Stanley and the narrators. Both shallow and deep characters are involved. The story is flexible both in satellites and kernels, letting the player pick their path via significant decisions that branch the narrative.

Conclusion

Advantages

Event Significance
The idea of separating significant (kernel) and insignificant (satellite) events is useful and could be used in an authoring system to tag events based on how they modify the story state.

Malleability
Categorizing objects and characters by their malleability and/or interactivity could be a useful grouping system for some authoring interfaces.

World Types
Having structural definitions of worlds is useful for authors. For example, it could be used as a template for a type of narrative or to abstract structure somehow.

Disadvantages

Character Definitions
The game breaks the fourth wall and directly includes both the actual player and people surrounding the player by including them in the narrative. The model cannot capture these as there is no suitable category available.

Changing Environments
The game sometimes reshuffles the level to either shorten a repeated playthrough or to provide access to alternative areas. This is not readily representable by the model, although it is a rather unique situation.

Malleability Limitations
The malleability of both objects and characters is unable to completely capture details. Objects can be difficult to assign, and the terminology used misses out a number of real-world object types. Refining these descriptions and adding new elements could perhaps enhance this technique for use in a grouping system.

CANVAS

It quickly became apparent when modeling *Portal* that this model is not designed to be detached from its authoring system, and as a consequence, without simulation suitable for representing predefined stories rather than those with interaction. As such, two approaches were taken. The first had the game mapped as I played it in a single instance, and the other attempted to emulate what players are expected to do but not a particular method of doing so. Listed below are tables from the former approach. This model was not applied to *The Stanley Parable* due to the early realization of its limitations.

Portal

Roles

Elevator, *Player*, *LevelSign*, *GLaDOS*, *Door*, *Turret*, *Portal*, *Cube*, *Button*.

States

Open, *On*, *Alive*, *Pressed*.

Affordances

Affordances are listed by name, participants, and postconditions.

ToggleOpen

ω_o, ω_u

$Open(\omega_o) = \neg Open(\omega_o)$

ToggleOn

ω_o, ω_u

$On(\omega_o) = \neg On(\omega_o)$

NavigateTo

ω_o, ω_u

$PathTo(\omega_u, \omega_o)$

SpeakTo

ω_o, ω_u

$Dialogue(\omega_o, \omega_u)$

GoThrough

ω_o, ω_u

$GoThrough(\omega_u, \omega_o)$

ShootAt

ω_o, ω_u

$FireAt(\omega_o, \omega_u)$

PickUp

ω_o, ω_u

$Carry(\omega_o, \omega_u)$

Throw

ω_o, ω_u

$Throw(\omega_o, \omega_u)$

DropIn

$\omega_o, \omega_{u1}, \omega_{u2}$

$DropIn(\omega_o, \omega_{u1}, \omega_{u2})$

PlaceBluePortal

ω_o, ω_u

$PortalBlue(\omega_o, \omega_u)$

PlaceOrangePortal

ω_o, ω_u

$PortalOrange(\omega_o, \omega_u)$

TogglePressed

ω_o, ω_u

$Pressed(\omega_o) = \neg Pressed(\omega_o)$

Events

Events are listed by name, preconditions, and logic.

OpenEntrance
ω_1 is EntranceElevator
$ToggleOpen(\omega_1)$

MoveToSign
ω_1 is Player, ω_2 is LevelSign
$NavigateTo(\omega_1, \omega_2)$

CloseEntrance
ω_1 is EntranceElevator
$ToggleOpen(\omega_1)$

GLaDOSWarnPlayer
ω_1 is Player, ω_2 is GLaDOS
$SpeakTo(\omega_1, \omega_2)$

OpenStartDoor
ω_1 is StartDoor
$ToggleOpen(\omega_1)$

EnterStartDoor
ω_1 is Player, ω_2 is StartDoor
$GoThrough(\omega_1, \omega_2)$

TurretShootPlayer
ω_1 is Turret role, ω_2 is Player
$ShootAt(\omega_1, \omega_2)$

ThrowTurret
ω_1 is Player, ω_2 is Turret role
$PickUp(\omega_1, \omega_2)$, $Throw(\omega_1, \omega_2)$

PlaceBluePortal
ω_1 is Player, ω_2 is Target Smart Object
$PlacePortalBlue(\omega_1, \omega_2)$

PlaceOrangePortal
ω_1 is Player, ω_2 is Target Smart Object
$PlacePortalOrange(\omega_1, \omega_2)$

EnterBluePortal
ω_1 is Player, ω_2 is BluePortal
$GoThrough(\omega_1, \omega_2)$

EnterOrangePortal
ω_1 is Player, ω_2 is OrangePortal
$GoThrough(\omega_1, \omega_2)$

PickupCube
ω_1 is Player, ω_2 is Cube role
$PickUp(\omega_1, \omega_2)$

DropCubeInPortal
ω_1 is Player, ω_2 is Cube, ω_3 is Portal
$DropIn(\omega_1, \omega_2, \omega_3)$

PressDoorButton
ω_1 is DoorButton, ω_2 is Door role
$TogglePressed(\omega_1)$, $ToggleOpen(\omega_2)$

GoToExitElevator
ω_1 is Player, ω_2 is ExitElevator
$NavigateTo(\omega_1, \omega_2)$

Smart Objects

The **EntranceElevator** Smart Object has a role of *Elevator*, the state *Open*, and the affordance *ToggleOpen*.

The **LevelSign** Smart Object has a role of *LevelSign*, the state *On*, and the affordance *ToggleOn*.

The **Player** Smart Object has a role of *Player*, has no states, and has the affordances *NavigateTo*, *SpeakTo*, *GoThrough*, *PickUp*, *Throw*, *DropIn*, *PlaceBluePortal*, and *PlaceOrangePortal*.

The **GLaDOS** Smart Object has a role of *GLaDOS* with no states or affordances.

The **SmartDoor** Smart Object has a role of *Door*, the state *Open*, and the affordance *ToggleOpen*.

Turrets 1..12 Smart Objects have a role of *Turret*, the state *Alive*, and the affordance *ShootAt*.

The **BluePortal** Smart Object has a role of *Portal* with no states or affordances.

The **OrangePortal** Smart Object has a role of *Portal* with no states or affordances.

Cube 1..3 Smart Objects have a role of *Cube* with no states or affordances.

The **DoorButton** Smart Object has a role of *Button* and the state *Pressed* with no affordances.

The **EndDoor** Smart Object has a role of *Door*, the state *Open*, and the affordance *ToggleOpen*.

The **ExitElevator** Smart Object has a role of *Elevator*, the state *Open*, and the affordance *ToggleOpen*.

Narrative Arcs

Narrative arc **Ch16P1** has the events *OpenEntrance*, *MoveToSign*, *CloseEntrance*, *GLaDOSWarnPlayer*, *OpenStartDoor*, *EnterStartDoor*.

Narrative arc **Ch16P2** has the events *TurretShootPlayer*, *ThrowTurret*, *PlaceBluePortal*, *PlaceOrangePortal*, *EnterBluePortal*, *ThrowTurret*, *PlaceBluePortal*, *EnterOrangePortal*.

Narrative arc **Ch16P3** has the events *PlaceBluePortal*, *PlaceOrangePortal*, *PickupCube*, *DropCubeInPortal*, *PlaceBluePortal*, *PickupCube*, *DropCubeInPortal*.

Narrative arc **Ch16P4** has the events *TurretShootPlayer*, *PlaceBluePortal*, *EnterOrangePortal*, *ThrowTurret*, *ThrowTurret*.

Narrative arc **Ch16P5** has the events *PlaceBluePortal, PlaceOrangePortal, PickupCube, DropCubeInPortal, ThrowTurret, PlaceBluePortal, EnterOrangePortal, ThrowTurret.*

Narrative arc **Ch16P6** has the events *PressDoorButton*, *PlaceBluePortal*, *EnterOrangePortal*, *ThrowTurret*, *PlaceBluePortal*, *PlaceOrangePortal*, *EnterOrangePortal*, *ThrowTurret*, *GoToExitElevator*, *GLaDOSWarnPlayer*.

Conclusion

Advantages

Level of Mapping
The use of attributes, roles, and affordances to define Smart Objects means that most situations can be modeled. This process is lengthy although becomes easier as more information is defined.

Disadvantages

Manual Labor
The effort to create something meaningful from this model is quite large. This appears to be due to the model not being intended to be separated from the authoring system, which eases the burden.
Lack of Agency
The model represents predefined situations without problem, such as cutscenes or animated sequence, but does not inherently support agency or freedom of choice. It could be possibly emulated using attributes.
Branching
As with agency, there is no concept of branching built into the model natively.
Generics and Triggers
This model struggles to represent situations where objects are part of a larger system, such as multiple buttons needing to be activated at the same time to unlock a door. This is achievable with attributes, but the attribute, roles, archetype, and event spaces quickly become polluted with specific elements. Similarly, there is no obvious way of handling triggers, such as a button with differing functions based on its state.

Patterns of Hypertext

Portal

Portal is a mostly linear game, and as such, does not demonstrate complex patterns. There is little in the way of repetition with *Portal*.

Montage

When playing *Portal*, the player can find abandoned dens that hint at the presence of a previous occupant. These are named Rattmann dens after the previously-inhabiting character Doug Rattmann. These dens increase narrative depth by providing the player with hints about the true nature of the game's antagonist, as well as giving an interesting insight into the past events of the facility, enriching the overall narrative experience. These dens and their presented information exist in parallel with the main story, although the character that was previously present exists in a different timeline. The dens compliment and reinforce the core narrative.

Cycle and Joyce's Cycle

Test Chamber 9 is first encountered near the beginning of the game. At this point, the player has access to only one of the two portals, and must solve the puzzle by placing only one color portal and using a single weighted cube. The exit is the same as any other standard test chamber. During the escape sequence, the player returns to this same chamber through a delivery tube rather than the regular entrance, and has to solve the same puzzle in a different way as they now have access to both portals but not access to a weighted cube. The player also exits the chamber differently. In the first instance of this chamber, the player is guided by GLaDOS, but in the second, they are rebelling against GLaDOS making an escape. The experience of this chamber the second time is different, as information gathered since the last encounter changes the understanding of the scene. The two instances of this chamber are displayed in Figure A.1.

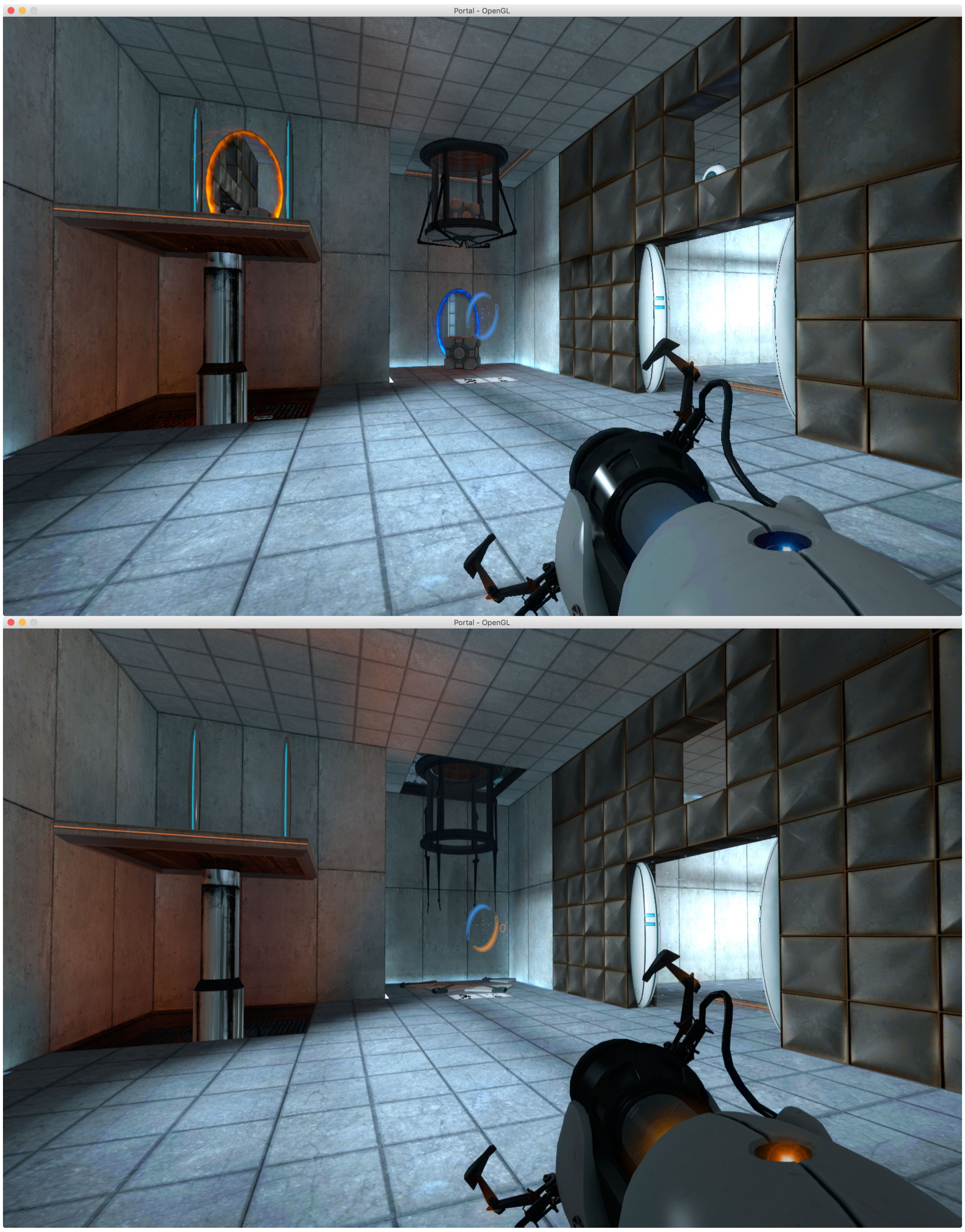


Fig. A.1 Test Chamber 9 before and during the escape sequence.

Missing Link

In *Portal*, the fallacy of being able to explore deeper into the facility's guts, beyond the walls of the test chambers and into the observation offices, is created through clever visual design of the levels. This creates a Missing Link, as the user comes to the logical conclusion that if they were able to reach said areas, then they would be able to explore them and possibly see what lies beyond further into the facility. These potential pathways, however, are in fact not accessible at all and are excluded from the narrative beyond the player's imagination. For instance, most test chambers contain a number of observation windows as displayed in Figure A.2. These windows usually contain a distorted layout reminiscent of an office with a door for access. Seeing this suggests to the player that there is a world beyond the door, providing they could reach the observation window area. In reality, however, there is no way to access the observation windows (with the exception of a set few as part of the story later in the game). In most cases, the area beyond the observation window literally does not exist within the game, even if cheats are used to get to the location. Similarly, there are a number of locked doors that the player can interact with (i.e., attempt to open by turning the handle), but each of these have either intentionally inaccessible areas or no areas at all behind them.

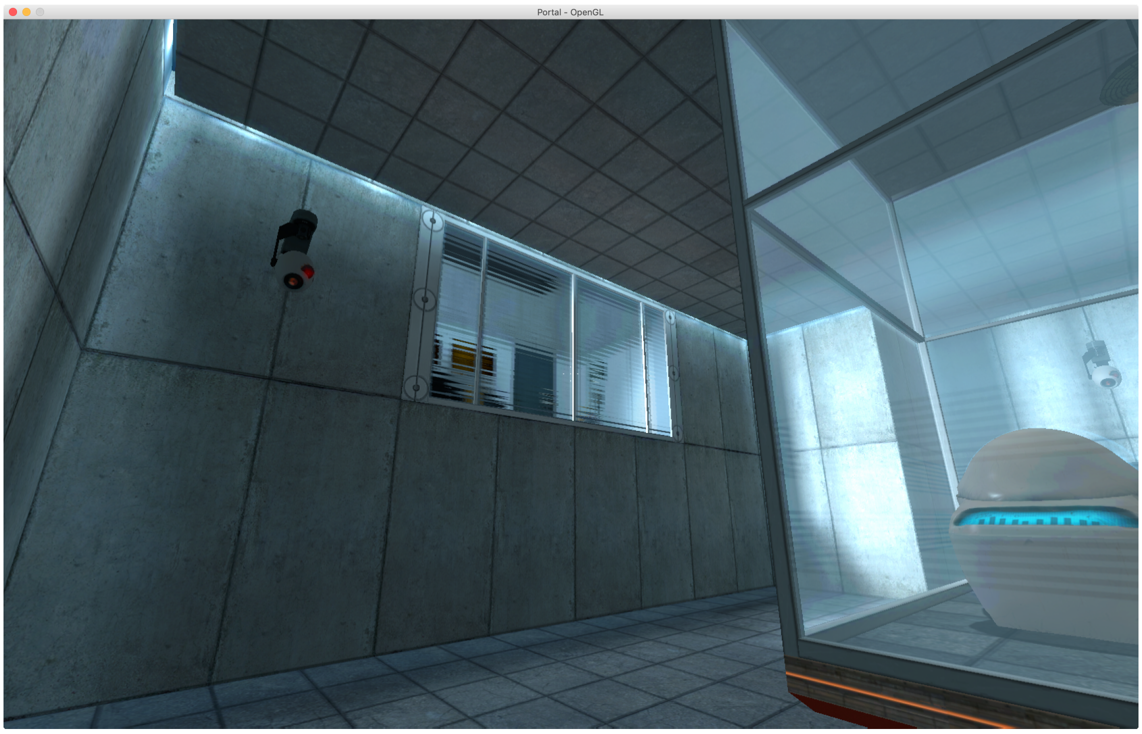


Fig. A.2 An inaccessible office window out of the player's reach.

The Stanley Parable

The Stanley Parable achieves its narrative power through excessive use of cycles and by breaking the player's expectations with respect to their conscious decisions.

Cycle and Joyce's Cycle

The Stanley Parable's narrative strength chiefly lies within the excessive use of Cycles and Joyce's Cycles. The two are deeply intertwined into the experience of the game. Therefore, there are countless examples that cannot be reasonably enumerated.

An example of a Cycle is the process of achieving the Heaven Ending. To do this, the player starts the game from the beginning and finds a specific computer and interacts with it. The player must then manually restart the game from the menu, traverse a similar path, experience the same office space, and find a different computer to interact with. This repeats four times, upon which the Cycle is broken and the player is transported to the Heaven ending.

Another example of both a Cycle and Moulthrop's Move is during the red/blue door sequence towards the Zending, Divine Art, and Games endings. In this scenario, the player is presented with a red door and blue door and is instructed to enter the red door. If they instead choose the blue door, they are teleported back to the beginning of the sequence and requested more fiercely to enter the red door again. This loops three times, and each time the narrator is more than aware of their defiance. The Cycle is broken when either the player goes through the red door at any point, or defiance of the narrator's instructions three times, with each breaking the Cycle into a different pathway.

Contour

There are no obvious Contours in this game. However, there are a number of Cycles that *appear* to overlap, but in reality, do not. For instance, during the Confusion ending, the player traverses the office many, many times, but during these repeated traversals, pathways initially available are visually present but not actually available for the player to follow.

Mirror World

There is no obvious Mirror World. However, it could be said that different pathways to different endings are alternative storylines within the same world. These are running not in parallel but separately.

Tangle

Restriction of information when presented with many options is used throughout the game. This can be both deceitful and for fun. An example of this is at the beginning of the Games ending. The player encounters the initial double door decision again, but this time with many doors instead of just two. They are given no real instruction on the significance of any choice and are instead left to guess which door is the right one to take. In reality this makes no difference, as all doors lead to the same geometrically-impossible looping corridor system. The uninformed choice that is presented to the player makes for an interesting experience and perhaps a false sense of agency over the narrative.

Sieve

The entire game can be thought of as a Sieve. When graphed, each ending can be reached from the origin by following a set sequence of predetermined choices. These choices act as a Sieve to filter the player through the same game to eventually achieve an ending.

Split/Join

The game's core mechanic is the presentation of two or more choices, an instruction, and the ability for the player to either obey or disobey with the suggested course of action. As such, Split/Join sequences are plentiful. Due to the complex nature of the pathways the player can follow, and the ability to jump between certain pathways, these patterns can be identified at numerous levels of detail.

For example, the initial double door decision provides an obvious binary split. Taking the left and right doors can put the player along dramatically different pathways. However, if the player chooses the right door and then follows a given set of directions, they can merge back to the path that would have occurred if they had taken the left door instead. In this case, their decision of whether to take the left or right door leads them to the same location. This Split/Join is only fulfilled if the player does indeed return to the same location.

A more concrete example of a lower-level Split/Join can be found in the earlier mentioned multiple doors decision towards the Games ending. The player is presented with many doors and instructed to pick any door to see where they lead, but in reality they loop around an infinite corridor with an identical result regardless of which door is chosen.

Rashomon

As described in other examples, there are numerous Cycles, each of which are broken by a Split/Join, therefore indicating the presence of Rashomon Split/Join patterns.

Overview/Tour

During the Not Stanley ending, the player is presented with the initial doors decision again. Defying the narrator by taking the right door continues this pathway. Taking the left door, however, has the player enter an alternative boss' office which eventually results in the narrator teleporting the player back to the decision to forcefully take the right door. Therefore, regardless of which choice the player makes, the destination is the same. However, the left choice provides significantly more narrative detail than the right.

Moulthrop's Move

An example of a Moulthrop's Move, as mentioned earlier, is the sequence of blue and red doors leading to the Zending, Divine Art, and Games endings. In this scene, the player is presented with a choice of a blue door and a red door and is instructed to enter the red door. If the player ignored the instruction and enter the blue door, they are teleported back and informed that they made the wrong choice. This presents the player with a Split/Join but resists the player's conscious choice. IF at any point the player chooses the red door, their decision is accepted and the containing Cycle is broken. Otherwise, the player's choice is overridden only two times before the narrative allows the player to fully defy the narrator's instructions, which also breaks the containing Cycle.

Missing Link

As with a large number of games, the existence of false doorways, offices, or other similar environmental elements, tricks the player into thinking the ludic space is larger than it actually is. In this game, a lot of navigation is via doors, but there exists countless more that the player cannot access (even if cheating). These extra doors suggest a larger world but are completely inaccessible.

An interesting implementation of this is the infinite elevator sequence in the boss' office. This elevator suggests to the player that they will reach another location - perhaps another floor with more area to explore. When the player is in the lift, it actively shakes and has ambient music, as if functional. In reality, however, the elevator is completely stationary and will never reach the suggested location, or any location.

Another narratively significant example is in the Not Stanley ending. When the player reaches the alternative boss' office as described above, they are instructed to speak into a device in order to unlock a door to progress but are unable to do so. The narrator explains that if they open the door, there is more narrative to follow, but there is no way for the player to complete this task. Even if cheating, the door cannot be passed. This false door and device suggest a larger scope than actually exists.

Navigational Feint

In this game, the player is exposed to an extortionate number of choices, each with their own potential consequences, yet only one at a time can be followed. As such, there exist a large number of potential pathways for the player to explore, suggesting a larger scope to the player, increasing curiosity and replayability. This exposure to potential navigation is also used as a narrative reveal during the Confusion ending. The player is accidentally guided to a catwalk above the Mine Control Facility's TV room, and the narrator hastily states how the player shouldn't have seen this area yet, explaining what is supposed to happen, including the scope of that pathway and summarizing the story arc to the player. This lets the player know that if they take an alternative pathway on another playthrough that they have the potential to access this teased area.

Conclusion

Advantages

Ability to Influence Structure
The strength of these patterns is to aid authors to create structures and well-formed narrative experiences. The patterns are not exhaustive. They do allow the writer to consciously think of their structure, and provide them a toolbox for designing how their narrative flows.

Disadvantages

Level of Detail
It is perhaps unfair to say that the patterns do not capture details of characters, events, and so on, as the patterns are not supposed to identify such things by design.

This page is intentionally left blank.

APPENDIX B

Little Red Riding Hood Script

This appendix contains the script provided to participants for the corresponding experiment. The page following this is where the document begins.

Little Red Riding Hood

Written by

Daniel Green

INT. INTRODUCTION: HOME

NARRATOR
Once upon a time, there was a little blond girl who lived in a village near a forest with her mother. One day, her grandmother bought her a nice red cloak. Everybody in the village called her Little Red Riding Hood. One day...

LITTLE RED RIDING HOOD
Did you call me, Mother?

MOTHER
Yes, Little Red Riding Hood. Your Grandma is rather sick. I want you to go to her house in the forest and take her this basket of food and drink.

LITTLE RED RIDING HOOD
Okay, Mother.

MOTHER
But do not stop in the forest and do not talk to strangers! Make sure you stay on the path and get to your Grandma's house safely and quickly.

LITTLE RED RIDING HOOD
Of course, Mother. I will.

NARRATOR
And so, Little Red Riding Hood left her home and ventured to the entrance of the nearby forest where her Grandma lives. However, little did she know there was an adventure to be had...

EXT. THE WOODLAND: FOREST

NARRATOR
Little Red Riding Hood then cautiously entered the forest proper.

LITTLE RED RIDING HOOD
I best stay to the path as Mother told me to.

NARRATOR

The brown, muddy path Little Red Riding Hood walked was welltrodden. The surrounding trees were still dense in beautiful autumn colors, yet beginning to shed their leaves as the season approaches. Various small branches and twigs lay scattered along the path, making for all but a silent journey.

LITTLE RED RIDING HOOD

Oh, these leaves are slippery! I must be careful.

NARRATOR

Little Red Riding Hood Was vigilant in her ways, but was often distracted by the beauty of nature. A butterfly could be seen fluttering over a nearby shrubbery off the path.

LITTLE RED RIDING HOOD

What a beautiful colored butterfly! Let me take a closer look. Come here, Mr. Butterfly!

NARRATOR

Little Red Riding Hood broke from the path in pursuit of this butterfly, as it spread its wings and glided blissfully in the gentle breeze. As Little Red Riding Hood approached, the butterfly, startled, flew into the distance.

NARRATOR (cont'd)

As Little Red Riding Hood peered through the brush in search for the butterfly, she noticed a dark shadow that appeared to be lurking and observing her in the distance. Her body tensed as she tried to make sense of what she was seeing.

LITTLE RED RIDING HOOD

Ahh! What is that in the distance? It must be a woodcutter for certain. Who else would be here in the forest?

NARRATOR
A sudden wave of anxiety overcome Little Red Riding Hood as she searched for answers. A sudden darkness appeared to encompass the forest encroaching on her from every direction. Another shadow appeared. This time, it dashed quickly from tree to tree, seemingly closing its distance. Little Red Riding Hood was terrified. She felt a lump in her throat and her heart pounced.

LITTLE RED RIDING HOOD
Oh, no! Oh, dear! I should have stayed the path like Mother told me to.

NARRATOR
She took slow steps forward through the brush to return to the path.

LITTLE RED RIDING HOOD
Careful, now. I just... need... to get through this... back to the... ouch!... path.

NARRATOR
As Little Red Riding Hood returned back to the path, she noticed another figure in the distance, but this time it was clear. It was a Woodcutter. She felt a wave of relief and concluded that she was seeing things. The Woodcutter, noticing Little Red Riding Hood, waved. She waved back, feeling much better. Then, the Woodcutter was on his way, and out of sight very quickly.

LITTLE RED RIDING HOOD
Phew! It was just a Woodcutter. I must have been seeing things. Well, I should continue to Grandma's. She needs these gifts!

NARRATOR
And so, Little Red Riding Hood, having returned to the path, continued on her way.

EXT. THE MEETING: FOREST

NARRATOR
As Little Red Riding Hood was following the path, she noticed something in among the leaves on the ground a letter, perhaps.

[**CHOICE**: "INVESTIGATE", "IGNORE"]

[**IF**: *INVESTIGATE*]

LITTLE RED RIDING HOOD
What's that over there?

NARRATOR
Little Red Riding Hood rummaged through the leaves and picked out a dirty piece of paper. After wiping it down, she noticed it was a letter with some kind of warning. The paper was worn and the text only just legible. It read "*Beware! Do not trust the Wolf! He is not what he seems and will try to deceive you.*" The letter was mysteriously not signed.

LITTLE RED RIDING HOOD
How strange. I must remember not to trust this Wolf if I meet him.

NARRATOR
Little Red Riding Hood pocketed the letter and continued on her way.

[**IF**: *IGNORE*]

NOTHING HAPPENS WHEN IGNORING. CONTINUE ON AS BELOW.

[**END**]

NARRATOR (cont'd)
As Little Red Riding Hood was walking along the path, a friendlylooking Wolf suddenly emerged from the brush. He said to the girl...

WOLF
Hello there, little girl. What's your name?

LITTLE RED RIDING HOOD
I'm Little Red Riding Hood. And who
are you?

WOLF
My name is Wolfgang Amadeus Mozart,
but you can call me Wolf. What are
you doing walking alone in such a
dangerous place?

NARRATOR
Little Red Riding Hood naively
conversed with the Wolf, revealing
perhaps too much information.

LITTLE RED RIDING HOOD
I'm taking these gifts to my
Grandma's cottage in the clearing.

WOLF
Do you mean the lovely little one
with the white door?

LITTLE RED RIDING HOOD
Yes, that's the one. How do you know
that?

WOLF
Because I live here! The forest is
my home. I know everybody here. I
think your Grandma would love some
flowers. You should collect her
three different colors. That will
make her feel even better.

[**IF**: Little Red Riding Hood picked up the letter]

LITTLE RED RIDING HOOD
(Thinking) That letter I found
earlier warned me about a mischievous
Wolf. Maybe I shouldn't trust him?

[**IF**: Little Red Riding Hood **did not** pick up the letter]

LITTLE RED RIDING HOOD (cont'd)
(Thinking) Wolfgang seems friendly.
I think it would be a good idea to
get the flowers for Grandma.

[**END**]

NARRATOR
Now Little Red Riding Hood, facing with the Wolf, had to make a decision. Would she take on the task of collecting flowers for her sick Grandma, or would she ignore the Wolf's advice?

[**CHOICE**: "Collect Flowers", "Ignore"]

[**IF**: Collect Flowers]

LITTLE RED RIDING HOOD
I think that sounds like a good idea. Grandma would like some nice colorful flowers. I'll go pick some now.

WOLF
A wise choice, little girl! I have to leave now; I have to tend to my decomposing forest!

NARRATOR
And with that, the Wolf took off. Little did she know that while she was collecting flowers, he was rushing to Grandma's house.

LITTLE RED RIDING HOOD
Now, where are those flowers? I want a red one, a blue one, and a yellow one!

NARRATOR
Little Red Riding Hood proceeded to find and pick the flowers for her Grandma, during which the Wolf was encroaching on Grandma's place.

[**IF**: Ignore]

LITTLE RED RIDING HOOD
I don't think that's a good idea. My Grandma has bad allergies and I don't want to make her worse.

WOLF
I'm sorry. I didn't know. I have to go now, little girl. Have a safe journey.

NARRATOR
And with that, the Wolf took off. He didn't have time on his side, so he took a shortcut to beat Little Red Riding Hood to Grandma's house.

[**END**]

NARRATOR (cont'd)
After her encounter with the Wolf, Little Red Riding Hood continued on her way to Grandma's house.

INT./EXT. THE CONFRONTATION: COTTAGE

NARRATOR
The sneaky Wolf managed to reach Grandma's house before Little Red Riding Hood and entered to find Grandma sleeping.

NARRATOR (cont'd)
The Wolf then crept up on Grandma and swallowed her whole! Gathering what he could, the Wolf then disguised himself as Grandma and lay in bed awaiting the arrival of Little Red Riding Hood with the intention of swallowing her, too.

LITTLE RED RIDING HOOD
I hope I'm not too late!

NARRATOR
As Little Red Riding Hood arrived, she knocked on the door, to which the Wolf replied in the best impression he could...

WOLF
Is that you, my dear? Come on in, I'm still sick in bed.

NARRATOR
Little Red Riding Hood proceeded to open the door and entered her Grandma's cottage.

WOLF
Oh! Are those gifts? Thank you, my dear.
(MORE)

WOLF (cont'd)
You can leave them on my chair. Now, come here so I can see my favorite granddaughter.

NARRATOR
Little Red Riding Hood inches closer to the bed to be near her Grandma. Noticing something is off with her Grandma, Little Red Riding Hood queried the Wolf...

LITTLE RED RIDING HOOD
Grandma, what a deep voice you have!

WOLF
All the better to greet you with, my dear!

LITTLE RED RIDING HOOD
And goodness, what large eyes you have!

WOLF
All the better to see you with!

LITTLE RED RIDING HOOD
And what big, furry hands you have!

WOLF
All the better to hug you with.

LITTLE RED RIDING HOOD
And what sharp teeth you have!

~~WOLF~~
~~(Clearly perturbed) You are by far my least favorite grandchild.~~

WOLF (cont'd)
All the better to **eat you with**!

NARRATOR
The Wolf then pounced from the bed and stood tall over Little Red Riding Hood. He was preparing to gobble her up!

[**CHOICE**: "Scream", "Do nothing"]

[**IF**: Scream]

LITTLE RED RIDING HOOD
Ahhhhhhh! Somebody help me!

NARRATOR
Little Red Riding Hood let out a sudden squeal loud enough to shatter glass. A nearby Woodcutter heard the scream and rushed to the scene, entering the cottage with force.

WOODCUTTER
What seems to be going on here?

NARRATOR
After quickly making sense of the situation, the Woodcutter confronted the Wolf and hit him in the stomach. Grandma popped right out, safe and sound, although a little traumatized.

WOODCUTTER
Take **this**!

NARRATOR
The Wolf collapsed to the floor, winded. While he was crawling away, the Woodcutter comforted Little Red Riding Hood.

WOODCUTTER
Are you both okay? I came as soon as I heard your scream.

GRANDMA
We are both fine now, thank you for saving my granddaughter and I. We are in your debt.

WOODCUTTER
I'm just glad you're both safe. The Wolf shouldn't bother you any longer.

LITTLE RED RIDING HOOD
Thank you, kind sir! I thought I was done for!

WOODCUTTER
My pleasure, little girl. If you hadn't alerted me, we may not be in this situation. Well done, your Grandma will be proud.

[**CHOICE**: "Kill the Wolf", "Spare the Wolf"]

[**IF**: Kill the Wolf]

LITTLE RED RIDING HOOD
I'm afraid the Wolf may come back.

WOODCUTTER
You're right. He's a danger and I will make sure he never bothers you or your family again.

[**IF**: Spare the Wolf]

LITTLE RED RIDING HOOD
That Wolf was so scary!

WOODCUTTER
Don't worry, I'll take him outside and give him a stern talking to.

[**END**]

NARRATOR
It was at this point that the Woodcutter left, Wolf in hand, and disappeared into the forest. He disappeared into the sunset. Both Little Red Riding Hood and her Grandma waved him goodbye as they return to their normal, Wolffree lives. **THE END**.

[**IF**: Do nothing]

WOLF
You look awfully tasty, little girl!

NARRATOR
The Wolf slowly encroached on Little Red Riding Hood, looming over her like a tower.

WOLF
It's time to join your Grandma!

NARRATOR
And with one large gulp, he swallowed Little Red Riding Hood whole. All fell silent. The Wolf, full and satisfied, returned to bed to sleep off his twoforone mealdeal. **THE END**.

[**END**]