Don’t Forget to Save!

User Experience Principles for Video Game Narrative Authoring Tools

Daniel Green

Faculty of Science and Technology
Bournemouth University

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

February 2022

Abstract

Interactive Digital Narratives (IDNs) are a natural evolution of traditional storytelling melded with technological improvements brought about by the rapidly increasing digital revolution. This has and continues to enhance the complexities and functionality of the stories that we can tell. Video game narratives, both old and new, are considered close relatives of IDN, and due to their enhanced interactivity and presentational methods, further complicate the creation process. Authoring tool software aims to alleviate the complexities of this by abstracting underlying data models into accessible user interfaces that creatives, even those with limited technical experience, can use to author their stories. Unfortunately, despite the vast array of authoring tools in this space, user experience is often overlooked even though it is arguably one of the most vital components. This has resulted in a focus on the audience within IDN research rather than the authors, and consequently our knowledge and understanding of the impacts of user experience design decisions in authoring tools are limited.

This thesis tackles the modeling of complex video game narrative structures and investigates how user experience design decisions within IDN authoring tools may impact the authoring process. I first introduce my concept of *Discoverable Narrative* which establishes a vocabulary for the analysis, categorization, and comparison of aspects of video game narrative that are discovered, observed, or experienced by players — something that existing models struggle to detail. I also develop and present my *Novella Narrative Model* which provides support for video game narrative elements and makes several novel innovations that set it apart from existing narrative models. This thesis then builds upon these models by presenting two bespoke user studies that examine the user experience of the state-of-the-art in IDN authoring tool design, together building a listing of seven general Themes and five principles (*Metaphor Testing*, *Fast Track Testing*, *Structure*, *Experimentation*, *Branching*) that highlight evidenced behavioral trends of authors based on different user experience design factors within IDN authoring tools. This represents some of the first work in this space that investigates the relationships between the user experience design of IDN authoring tools and the impacts that they can have on authors. Additionally, a generalized multi-stage pipeline for the design and development of IDN authoring tools is introduced, informed by professional industry-standard design techniques, in an effort to both ensure quality user experience within my own work and to raise awareness of the importance of following proper design processes when creating authoring tools, also serving as a template for doing so.

Table of contents

List of figures

List of tables

List of Games & Interactive Fictions

Below is a listing of video games and interactive fiction works mentioned throughout the body of this thesis. *Video Games* are sorted by series, then by year, then by game. *Interactive Fictions* are sorted by year then by name. Where able, primary developers or authors are used over publishers.

Video Games

Assassin's Creed III by Ubisoft Montreal, 2012.

BioShock by Irrational Games, 2007.

Braid by Jonathan Blow, 2008.

Call of Duty: Modern Warfare 2 by Infinity Ward, Inc., 2009.

Colossal Cave Adventure by William Crowther, 1976.

Dog Star Adventure by Lance Micklus, 1979.

Dragon's Lair by Advanced Microcomputer Systems, 1983.

Dwarf Fortress by Bay 12 Games, 2002.

Everlasting Summer by Soviet Games, 2013.

Façade by Procedural Arts, 2005.

Hellblade: Senua's Sacrifice by Ninja Theory, 2017.

Left 4 Dead 2 by Valve Corporation, 2009.

Life is Strange by Don't Nod Entertainment SA, 2015.

Loneliness by Jordon Magnuson, 2010.

Mass Effect by BioWare, 2007.

Mass Effect 2 by BioWare, 2010.

Max Payne by Remedy Entertainment Oyj, 2001.

Missile Command by Atari, Inc., 1980.

Myst by Cyan, Inc., 1993.

Pac-Man by Namco Ltd., 1980.

Passage by Jason Rohrer, 2007.

Pokémon Go by Niantic, Inc, 2016.

Portal by Valve Corporation, 2007.

Return of the Obra Dinn by Lucas Pope, 2018.

Shenmue by Sega-AM2 Co., Ltd., 1999.

Shenmue II by Sega-AM2 Co., Ltd., 2001.

Spacewar! by Steve Russell, 1962.

Spec Ops: The Line by Yager Development, 2012.

Tetris by AcademySoft, 1984.

The Elder Scrolls V: Skyrim by Bethesda Game Studios, 2011.

The Marriage by Rod Humble, 2007.

The Stanley Parable by Galactic Cafe, 2013.

The Sumerian Game by William McKay & Mabel Addis, 1964.

The Talos Principle by Croteam, 2014.

Tomb Raider (series) by various developers, 1996–Present.

Uncharted 2: Among Thieves by Naughty Dog, LLC, 2009.

Uncharted (series) by Naughty Dog, LLC, 2007–Present.

Wing Commander III: Heart of the Tiger by Origin Systems, Inc., 1994.

Zork I by Infocom, 1980.

Interactive Fictions

afternoon, a story by Michael Joyce, 1987.

Uncle Buddy's Phantom Funhouse by John McDaid, 1992.

pianographique by Jean-Luc Lamarque et al., 1993.

Patchwork Girl by Shelly Jackson, 1995.

The Hacker Crackdown by Bruce Sterling, 1995.

Uncle Roger by Judy Malloy, 1995.

Sunshine '69 by Bobby Rabyd, 1996.

Lair of the Marrow Monkey by Erik Loyer, 1998.

Looppool by Bastian Böttcher, 1998.

Fibonacci's Daughter by M.D. Coverley, 1999.

London Eye by Diane Greco, 1999.

Califia by M.D. Coverley, 2000.

Code X by W. Mark Sutherland, 2001.

Vniverse by S. Strickland/C. Jaramillo, 2002.

Shelley's Heart by Brad Gyori, 2018.

List of Software

Below is a listing of software mentioned in the context of research throughout this thesis. *Academic Software* are sorted alphabetically by tool name. *Commercial/Other Software* are sorted by year then by name. Dates, when present, refer to the *initial* release of the software unless a specific version is included.

Academic Software

ASAPS [110]

DraMachina [60]

ELIZA [216]

EmoEmma [40]

FearNot! [116]

FRESS [52]

GAIA [109]

GHOST [93]

HES [38]

HypeDyn [153]

HyperTIES [193]

INSCAPE [12]

Intermedia [148]

Microcosm [49]

NLS [63]

NM2 [213]

PaSSAGE [209]

Scenejo [78]

ScriptViz [133]

SHRDLU [220]

Story Canvas [196]

Story World Builder [172]

StoryPlaces [98, 149]

Storyspace [20, 21]

StoryTec [79]

SVC Editor [222]

TALE-SPIN [146]

Villanelle [139]

Virtual Human [80]

Virtual Storyteller [205, 208]

Xanadu [161]

ZOG [178]

Commercial/Other Software

The Quill by Gilsoft, 1983.

Adventure Builder by Sinclair User, 1986.

Adventure Builder System by Tartan Software, 1986.

Adventurer! by Sinclair User, 1986.

The Graphic Adventure Creator by Incentive Software Ltd., 1986.

HyperCard by Apple Computer, Inc., 1987.

Professional Adventure Writer by Gilsoft, 1987.

Director by Macromedia, Inc. (more recently by Adobe Inc.), 1988.

Text Adventure Development System by Michael J. Roberts, 1988.

ToolBook by Asymetrix Learning Systems, 1990.

RPG Maker by Multiple, 1992.

Inform by Graham Nelson, 1993.

FrontPage by Vermeer Technologies, Inc. (more recently by Microsoft Corporation), 1995.

Hugo by Kent Tessman, 1995.

Flash by Macromedia, Inc. (more recently by Adobe Inc.), 1996.

ADRIFT by Campbell Wild, 1997.

Adventure Game Studio by Chris Jones, 1997.

Dreamweaver by Macromedia, Inc. (more recently by Adobe Inc.), 1997.

Quest by Alex Warren (and community), 1998.

GameMaker by Mark Overmars (more recently by YoYo Games), 1999.

IF Creator by n-Discovery, 2002.

Ren'py by Tom Rothamel, 2004.

Unity by Unity Technologies ApS, 2005.

Microsoft Digital Image 2006 by Microsoft Corporation, 2006.

Scrivener by Literature & Latte Ltd., 2007.

GDevelop by Florian Rival, 2008.

ChoiceScript by Choice of Games LLC, 2009.

Twine by Chris Klimas (and community), 2009.

Undum by Ian Millington, 2010.

Fade In by GCC Productions Inc., 2011.

OBS Studio by Hugh Bailey (and community), 2012.

Playfic by Andy Baio & Cooper McHatton, 2012.

Dedalus by Gustavo Di Pietro, 2013.

Inklewriter by inkle Ltd., 2013.

Squiffy by Alex Warren (and community), 2014.

TextureWriter by Juhana Leinonen, Jim Munroe, 2014.

Unreal Engine 4 by Epic Games, Inc., 2014.

Chronicler by Garrett Fleischer, 2015.

Fungus by Chris Gregan & Steve Halliwell, 2015.

Genarrator by Unknown, 2015.

Inky by inkle Ltd., 2015.

Monogatari by Diego Islas Ocampo, 2015.

Adobe XD by Adobe Inc., 2017.

articy:draft 3 by articy Software GmbH & Co. KG, 2017.

Bitsy by Adam Le Doux, 2017.

Celtx Game by Celtx Inc., 2017.

Arcweave by Arcweave OU, 2019.

NVivo Pro v12.6.0.959 by QSR International, LLC, 2019.

Discord v73806 by Discord Inc., 2020.

LimeSurvey v4.2.0 by LimeSurvey GmbH, 2020.

Skype v8.65.0.78 by Microsoft Corporation, 2020.

Publications

Daniel Green, Charlie Hargood, and Fred Charles, "Use of Tools: UX Principles for Interactive Narrative Authoring Tools," *Journal on Computing and Cultural Heritage*, volume 14, number 3, 2021. DOI: 10.1145/3458769

Daniel Green, Charlie Hargood, and Fred Charles, "A Novel Design Pipeline for Authoring Tools," in *Interactive Storytelling*, Lecture Notes in Computer Science, Cham: Springer, 2020, pages 102–110, ISBN: 978-3-030-62516-0. DOI: 10.1007/978-3-030-62516-0_9

Daniel Green, Charlie Hargood, and Fred Charles, "Novella 2.0: A Hypertextual Architecture for Interactive Narrative in Games," in *Proceedings of the 30th ACM Conference on Hypertext and Social Media*, HT '19, Hof, Germany: ACM, 2019, pages 77–86, ISBN: 978-1-4503-6885-8. DOI: 10.1145/3342220.3343655

Daniel Green, Charlie Hargood, and Fred Charles, "Define "Authoring Tool": A Survey of Interactive Narrative Authoring Tools," presented at the Authoring for Interactive Storytelling Workshop, 2018

Daniel Green, Charlie Hargood, and Fred Charles, "Contemporary Issues in Interactive Storytelling Authoring Systems," in *Interactive Storytelling*, Lecture Notes in Computer Science, Cham: Springer, 2018, pages 501–513, ISBN: 978-3-030-04028-4. DOI: 10.1007/978-3-030-04028-4_59

Daniel Green, "Novella: An Authoring Tool for Interactive Storytelling in Games," in *Interactive Storytelling*, Lecture Notes in Computer Science, Cham: Springer, 2018, pages 556–559, ISBN: 978-3-030-04028-4. DOI: 10.1007/978-3-030-04028-4_66

Daniel Green, Charlie Hargood, Fred Charles, and Alex Jones, "Novella: A Proposition for Game-Based Storytelling," presented at the Narrative and Hypertext Workshop, 2018

Alex Jones, Brad Gyori, Charlie Hargood, Fred Charles, and Daniel Green, "Shelley's Heart: Experiences in Designing a Multi-Reader Locative Narrative," presented at the Narrative and Hypertext Workshop, 2018

Acknowledgments

I would like to thank all of those who have actively engaged with this project over the years both academically and personally; your guidance and support have helped to make this thesis a reality and have helped me to grow as a person. I'd like to particularly highlight my primary supervisor, Dr. Charlie Hargood, who has often gone above and beyond his role, has shown a genuine interest throughout the project, and has been a great mentor and friend over the years.

I offer my gratitude to my family and friends who have supported and tolerated me throughout this chapter of my life, particularly my wife, Mengna.

I also offer my extended thanks to my fellow students for the endless banter and support in a wide variety of topics, and for being great friends beyond sharing an office.

Finally, I would like to thank University Music for all that they do and for providing such a wide range of pianos that I have had the pleasure of enjoying over the years.

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
February 2022

Nomenclature

Glossary

Narrative Model 2
An abstracted representation of narrative structure that describes the constituents that make up a narrative and the relationship between them.

Hypertext 13
A body of written or pictorial material interconnected in such a complex way that it could not be conveniently presented or represented on paper.

Arborescent 23
Having the shape or characteristics of a tree.

Spatial Hypertext 50
Presentation of interconnected hypertextual content as a visual network of links and nodes, typically referred to as a map, making use of both the spatial relationships between nodes and the properties of the nodes themselves.

Interchange Format 78
An intermediate file format that's used to transfer data between two usually separate pieces of software.

Acronyms / Abbreviations

IDN 1
Interactive Digital Narrative.

UX 2
User Experience.

IDE 49
Integrated Development Environment.

UR 155
User Research.

UXR 155
User Experience Research.

"I'm Commander Shepard, and this is my favorite ~~store~~ thesis on the Citadel!"

— Commander Shepard

"If our lives are already written, it would take a courageous man to change the script."

— Alan Wake

CHAPTER 1

Introduction

In this thesis, I explore the modeling of video game narrative as a subset of interactive digital narrative (IDN) and explore the impact that domain-specific user experience design choices within IDN authoring tools have upon authors. This thesis makes several novel contributions in this research space which are briefly outlined below and detailed in §7.

In §2.4.2, a wide range of IDN authoring tools are categorized by their methods of software delivery and utilized interface design paradigms, resulting in an understanding of the distribution of existing authoring tools.

In §3.1, a generalized architecture for video game narrative authorship independent of genre or runtime distribution is proposed which enables contextual development of its individual components rather than in isolation from each other.

In §3.3, my concept of *Discoverable Narrative* is introduced for descriptive classification of video game narrative aspects that are discoverable, observable, or experienced, which is something that existing models struggle to sufficiently capture.

In §3.4, my game-centric and genre-independent *Novella Narrative Model* is presented along with demonstrative worked examples which makes several key innovations over existing video game narrative modeling approaches.

In §5, a custom multi-stage pipeline for the design and development of IDN authoring tools is presented, informed by professional industry-standard techniques, which can be followed to better inform the authoring tool design and development process, enabling a more refined user experience but also a reduction in potential usability problems.

Through two studies outlined in §4 and §6, a listing of design principles is created that highlight relationships between user experience aspects of IDN authoring tools and their potential impact upon the workflow of authors. A set of Themes based on qualitative data from the second study is also contributed that describe evidenced behavioral authoring trends of user experience design factors between involved participants.

Storytelling is one of the oldest and most ubiquitous forms of cultural expression. Whether it be in the form of ancient hieroglyphics forming some of the first known texts, the contents of a good book, the latest video game, or even browsing the news over breakfast, they all describe a narrative; they tell a story. The majority of traditional narrative works

— poetry, novels, folklore and such — follow linear structures where the reader is passive, with a few notable exceptions[1]. This was disrupted with the emergence of interactive narrative which actively encourages nonlinear structures in otherwise traditionally linear mediums, inviting the reader to instead participate in the shaping of the narrative through choice. The advent of the home computing revolution has brought about interactive *digital* narrative, which further enhances the presentational methods and interaction paradigms available to authors by utilizing computer interfaces for delivery of their narratives. Video game narrative, both old and new, falls under the umbrella of IDN, although this has been a point of contention in times past (see §2.3). As the capabilities of IDN experiences grow, the challenge and difficulty of authoring such experiences naturally increases, in part due to the narrative variation that occurs with added interactivity. The overall authoring process covers a wide range of activities that include ideation and exploration, creation of narrative structure and content, and even deployment or distribution of the resulting narrative experience. This thesis chiefly focuses on the creation of narrative structure and content which is the most technically demanding phase, and in video game production is typically handled by a Narrative Designer — a developer who specializes in both writing and technical skills. To simplify this process, software was introduced to enable creatives to design and build IDNs, referred to as *authoring tools*. The technical skills required to develop IDNs can act as a gatekeeper preventing creatives from using the medium [149, 201], and indeed these authoring tools aim to lower the barrier for entry by providing a user-friendly interface to create IDN experiences. Although a large quantity of IDN authoring tools have been created over the years (see §2.4), not all were publicly available for use, and even fewer remain available for use today. Moreover, while the user interface design and user experience (UX) considerations of these authoring tools are of vital importance, it is a comparatively neglected area of research in part due to the difficulties involved with evaluation (see §4). Although properly following best UX practices when designing authoring tools will create competent and usable interfaces (see §2.5), there may be unforeseen domain-specific consequences of our design decisions for the user interface and UX interaction paradigms that have yet to be properly explored. Identifying these links could help to better understand and inform our designs for authoring tools to ultimately improve the authoring experience. Although this thesis will focus on the UX of authoring tools for IDN, its primary interest lies within the specific domain of video game narrative viewed as a relative of IDN.

To understand the user experience of IDN authoring tools, we must also discuss the foundations upon which they are built. The user interface of an authoring tool bidirectionally abstracts an underlying data model (i.e., reads from to display, and writes to through interaction) such that authors can create their narratives in a user-friendly visual environment. Data models, in the case of IDN authoring tools, are computer-readable representations of an equivalent narrative model. These narrative models are derived from the systematic process of breaking down a narrative into an abstraction of

its regular form, describing the constituents that make up a narrative and the relationship between them (see §2.1). As these narrative models increase in complexity, such changes bubble up to the user interface of the authoring tool. We can consider how simple linear and nonlinear narrative structures differ from dynamic nonlinear structures and the impact that the respective models could have upon the resulting authoring tool interface.

A simple temporally linear structure refers to the sequential portrayal of narrative events in their actual chronological order. A trivial nonlinear structure may use techniques such as analepsis (flashback) and prolepsis (flashforward) to disturb this natural ordering by presenting the narrative events out of sequence, but still nonetheless in a linear format. Reading of these simple linear and nonlinear structures does not require active participation from the reader, and each time a reading takes place, the chronology of the events presented to the reader does not change; two people can read the same narrative and will always have the same structural experience. Narrative models of this form can therefore safely assume that events always follow a single previous event (whether chronological or not) and that their ordering does not change nor have any structural impact when encountered. When an authoring tool implements abstractions for such a model, these assumptions can also be taken advantage of by simplifying the interface to best suit stringing together individual events that cannot be changed or influenced during a reading of the resulting narrative.

On the other hand, dynamic nonlinear structures place no such constraints upon the organization of contained events, actively encouraging narratives with features such as no fixed entry point, multiple branching pathways, and impactful choice for the reader. The introduction of even trivial dynamic narrative structures can have a significant impact upon the user interface requirements for authoring tools. Take, for example, gamebooks such as *Choose Your Own Adventure* [43], the *Time Machine* series [210] and the *Endless Quest* series [62], which invite the reader to become an active participant in the text, able to make choices (usually by means of choosing which page to turn to from a list of options) that impact the temporal structure of events for that particular reading. This means that two people can read the same book and it is unlikely that they will have the same structural experience, although the total number of permutations still remains limited. Narrative models can no longer assume a fixed sequential line of isolated events and instead must consider multiple pathways of events that can interleave and branch based upon reader choice. This complicates the requirements for the user interface of an authoring tool, as the author must be able to create and link arbitrary sequences of narrative events together, be able to offer several pathways to a reader that diverge from a single event, and conversely be able to handle the merging of several distinct pathways into a singular event.

When considering the enhanced presentational and interaction features that IDN brings, narrative modeling, and through it the user interface requirements for authoring tools, are complicated further. The early text adventure game *Zork I*, for instance,

combined a rich narrative found in traditional media like novels with the interactivity of contemporary computer terminals. Players took control of a virtual avatar by entering commands into a computer terminal which allows for exploring of the environment (with some regions conditionally locked) solving puzzles along the way, collecting and using items, as well as making story-critical choices, all at their own leisure. The player could revisit a scene that they had previously encountered, with the content possibly differing based on items they have since collected, quests completed, time elapsed, and other similar attributes. This kind of dynamic behavior can contribute to the richness of the narrative and believability of a living, breathing world, as the player's actions in the world have a clear impact upon its state. All of these features introduce the concept of state management into narrative models to represent not only individual instances of choice, but also accumulation or presence of things within the game world (number of items collected, whether a door is unlocked, total traversal steps, passage of time, etc.). This alters the user interface requirements for IDN authoring tools as they must now be able to create, manage, and utilize arbitrary state information within the narrative.

These examples demonstrate how increasing the presentational methods, interaction paradigms, and dynamic functionality of narrative available to authors will translate to more complex narrative models, ultimately impacting the user experience design considerations for corresponding authoring tools. Moreover, it shows how narrative complexity can incrementally build upon the techniques of its predecessor. Given the additional challenges presented by video game narrative (see §2.3) and the influence this can have upon authoring tool design, this raises the questions of how different authoring interfaces impact the user experience, how well suited existing narrative models are at representing the nuances that it introduces, and how to create new models informed by existing approaches that successfully abstract video game narrative whilst supporting other forms of IDN.

1.1 Research Questions and Scope

Throughout this thesis, I will explore the following three research questions:

1. To what extent do existing narrative models capture the complexities of video game narrative and how might a new model better support this?

2. How do existing paradigms within authoring tool interface designs impact the user experience and workflow of authoring video game narrative?

3. What are the user experience principles for the design of video game narrative authoring tools based on empirical and experimental methodologies?

The first research question focuses on understanding how and to what extent existing approaches to narrative modeling are able to capture the nuances and complexities of

video game narrative and inversely what they are unable to capture. It also questions how we may build upon these approaches to inform a new narrative model that better represents video game narrative. In order to answer this research question, we will have to identify existing narrative models and determine, through practical application, how a representative subset of these models can be applied to video game narrative, evaluating which elements of the narrative they were able to capture and which they struggled with. The goal of this application would be to understand the contemporary approaches available to modeling video game narratives that underpin authoring tools, particularly the granularities of the models, to influence and inform a newer narrative model.

The second research question investigates how interface design paradigms found within authoring tools impact the authoring experience and the way that authors choose to work. As mentioned earlier, there is little work into the impact of design decisions specifically for authoring tools. Identifying and understanding these relationships will allow for more informed design choices to be made when building new authoring tools such that authors are impacted in a minimal or preferable way. To answer this research question, we will have to identify the different paradigms found in interfaces of existing authoring tools, and using a representative selection, determine if and how the observed behaviors of authors differ based on the paradigms exerted by the particular tools.

The third research question seeks to identify how we can create user experience principle guidelines for the design of video game narrative authoring tools from empirical evidence gathered in experimental methodologies. These guidelines could then be used as indicators by future developers to better understand the impact of their design decisions upon the user's experience with the tool. Acting as an extension of the second research question, the impact of various user experience aspects of the included authoring tools can be generalized into a framework of principles. This then warrants further investigation and validation of the initial principles through an exploratory experiment in which they may be refined by means of editing, appending, or removal based on newfound evidence that supports or refutes their definitions. The result would be a set of refined principles based upon empirical evidence that could determine the potential impact of design decisions, serving as a reference for designers and developers of authoring tools.

1.2 Thesis Structure

This thesis makes several contributions in narrative modeling and user experience design of authoring tools focusing on the subset of video game narrative, that are distributed throughout the seven chapters that make up the document body.

This first chapter introduced the topic and motivations of the thesis, specified the research questions that this thesis focuses upon, and outlines its structure.

The second chapter presents significant related background material beginning with narratology and structuralism as the foundations of narrative structure before moving

onto narrative systems where foundations of hypertext and its influence across works of fiction, early video games, and computational narrative are explored. This is followed by a discussion on approaching contemporary video games as a narrative medium along with a brief review of how narrative can find itself interwoven into video games. The chapter then focuses on the authoring of such narrative experiences, covering a history of interactive narrative authoring tools and their evolution over time, analyzing existing authoring tools, and considering how they fit into the bigger picture. The chapter then concludes with a detailed exploration into the importance of user experience in the context of authoring tools from a design and research perspective.

The third chapter focuses on modeling of video game narrative, starting with the presentation of a generalized architecture of video game narrative authorship which helps to position the research into a coherent and contextualized framework. Following is an exploration into approaches to game narrative modeling through either direct support or adaptation, followed by an application of various existing narrative models to video games in order to evaluate and understand their applicability in the video game narrative space. In response to an identified gap, a descriptive concept named *Discoverable Narrative* is created and demonstrated. A more complete narrative model named *Novella* is then introduced, highlighting its details and unique contributions to the representation of video game narrative structures.

The fourth chapter describes the preparation, execution, and analysis of the first major study of this thesis, resulting in a set of preliminary design principles that highlight relationships between the design of user interfaces in IDN authoring tools and their potential impact upon the workflow of authors, serving as guidance for future authoring tools and as a vocabulary for further discussion of UX issues facing authorship technology.

The fifth chapter returns focus to the development of authoring tools with the creation of an informed custom design pipeline specifically crafted for IDN authoring tool design and development. This is accompanied by a complete execution and reporting of the pipeline to serve as a case study for the pipeline's practicality and for the resulting artifact to be used as an experimental device in the chapter that follows.

In the sixth chapter, the second major study of the thesis is presented, which builds upon the preliminary design principles outlined in the fourth chapter to further explore, expand, validate, and refine them in new contexts, with the output from the previous chapter being a major component in the bespoke methodology.

The seventh chapter concludes this thesis, summarizing the work that has been done, outlining the key contributions that it makes, describes complementary publications made during the course of the PhD and how they relate to the chapters, and outlines avenues for how this research can be taken forward in the future.

CHAPTER 2

Related Background

This chapter presents the historical and technical background upon which this thesis builds. I first discuss relevant narratological theory which serves as the foundation for modeling of narrative structures. Next, I investigate existing narrative systems by looking at their hypertextual foundations, discussing hypertext fiction and interactive fiction, and providing an overview of computational techniques for narrative generation. Attention is then turned to video games, exploring if and how they can present narrative experiences. This is followed by a deep dive into the evolution of authoring tools for IDN, an analysis of delivery methods and interface paradigms of existing authoring tools, and a discussion of how different tools fit within a larger context. Building from that, the chapter then concludes with a discussion of user experience design and research concepts in the context of authoring tool design.

2.1 Narratology & Structuralism

When discussing narratology, I am referring to the collective theories used for systematic study of narratives. These theories allow us to break down constituents of narratives, identify what they have in common, how they differ, and determine collections of rules for narratives and their structures. The term 'narratology', derived from the French 'narratologie', was introduced by Todorov in *Grammaire du Décaméron* [211]. Two of the largest influences on the development of narratology are the works of Russian Formalists and French Structuralists (which includes Todorov). These fields are of interest to this thesis due to their popularity in the modeling of hypertextual systems, and increasingly as a method of study in video games.

We can trace back one of the oldest surviving texts demonstrating literary theory to the Greek philosopher Aristotle, who in his seminal work *Poetics* [95] laid out six elements of tragedy. Mythos (plot) refers to the organization of incidents that occur in the narrative in that events should come logically based on what has happened before. Ethos (character) determines the archetypical behaviors that participants within the narrative display, much alike behaviors expected by protagonists and antagonists. Dianoia (thought) extends ethos to include reasoning about the characters to explain their motives or other

background. Lexis (language) again relates to characters in determining their speech characteristics and how they reflect the character's archetype. Melos (melody) dictates that chorus should too be considered an important factor in delivering a narrative. Opsis (spectacle) refers to supplementary apparatus used to enhance the narrative such as costumes, set pieces, and so on. Within this work, Aristotle also outlines the overall structure of narratives, consisting of a beginning, a middle, and an end. Today, this theory is known as the *three-act structure*, which was further formalized by Syd Field in his book *Screenplay: The Foundations of Screenwriting* [67].

Although it has been over *two thousand years* since Aristotle first began to enumerate and explore narrative, the same techniques can be still found in contemporary analysis. In 1927, Edward Forster published *Aspects of the Novel* [70] in which he tackled six characteristics of novels. He defines story as a series of narrative events arranged in a time sequence and demonstrates this as a requirement by critiquing the American writer Gertrude Stein, who in her experimental works attempted to abolish time from her novels, but ultimately did not succeed as the results were dense and devoid of plot or dialogue. Another contribution of his is the enumeration of character profiles within a story based on their complexity; 'flat' characters are relatively simple and straightforward, whereas 'round' characters are more complex and developed. This categorization of characters based on their development and prominence has applications beyond novels, such as Aarseth's extension of the concept to represent video game characters [3]. Forster distinguishes between his definition of story and plot. He states where story is a temporal sequence of events, plot builds upon this by emphasizing causality. He presents an example to clearly separate the two concepts. "The king died, and then the queen died" is a story, as there is no causality as to what brought about the deaths. However, "The king died, and then the queen died *of grief*" is a plot, because causality is added, explaining the demise of the queen. In the second example, the temporality is preserved, but a sense of causality is brought about, differentiating it from the original form.

Around the same time, Russian Formalists began exploring the distinction between narrative events and the way that they are presented. In particular, the differentiation between fable (*fabula*) and plot (*sjuzet*). The exact origin of these terms is uncertain, although we can trace influence back to works of Aristotle in that his *mythos* is closely related to *fabula*. One of the earliest direct references to the concept is in 1925 by Viktor Shklovsky in his *Theory of Prose* [192]. Within this, Shklovsky briefly outlines that "the concept of plot (sjuzet) is too often confused with a description of the events in the novel, with what I'd tentatively call the story line (fabula). As a matter of fact, though, the story line is nothing more than material for plot formation". He also discusses at length Lawrence Sterne's *Tristram Shandy* [203] from the point of view of ostranenie (or defamiliarization, a concept that he pioneered in prior works from 1917 [191]), highlighting its artful deviations from the natural chronological sequence of events of the story. This concept of defamiliarization of the story is what became sjuzet. However, Shklovsky's

idea of sjuzet went beyond just the artistic arrangement of a story, also including devices employed in the delivery of the story, such as digressions from the main theme, which were found to be "as integral a part of the 'plot' as the story itself" [64].

Shortly thereafter, French Structuralists adapted and proposed their own terms — *histoire* was used in place of fabula, and either *récit* or *discours* as equivalents to sjuzet. As investigated by Brooks [33], the corresponding English translations can be misleading. Generally, fabula is translated as 'story' and sjuzet as 'plot', both of which are valid substitutions. However, Brooks highlights how the English term *plot* actually straddles the line between fabula and sjuzet, as "to speak of plot is to consider both story elements and their ordering", concluding that "plot is thus the dynamic shaping force of the narrative discourse". Brooks' position is reaffirmed by Ricoeur who defined plot as "the intelligible whole that governs a succession of events in any story" [177]. Ricoeur favors the terms 'events' and 'story' over fabula and sjuzet. Brooks argues that Ricoeur's definition, in this context, demonstrates that "a story is *made out of* events to the extent that plot *makes* events *into* a story", therefore resolving plot at the crossing point between temporality and narrativity. That Ricoeur favors individual narrative events shows effort to isolate and identify units of narrative — a crucial part of structuralism.

Mieke Bal has more recently investigated the constituents of both fabula and sjuzet and their relationship to one another [11]. According to Bal, a fabula can be described by four core elements: events, actors, time, and location. Events refer to "the transition from one state to another state, caused or experienced by actors", are typically chronological, and are further defined by any combination of three criteria. The first criterion is change — the event describes a change in state, and in more complex cases can describe causality. The second criterion is choice — the description of consequence of sequential events, or events that suggest a deliberate choice performed by the actor. The third criterion is confrontation — the idea that events describe two actors or groups of actors being confronted by each other in some way; this is defined by the form *subject-predicate-(direct)object*, where both the subject and object must be actors. Bal's concept of actors refers to entities within the fabula that take part in events (split by being anthropomorphic or not), with the relationships between them, in the context of events, being described by Algirdas Greimas' actantial model [91], which is discussed in more detail below. Time is an attribute of events that determines the duration over which the change in state occurs, and inherently lays out the events into a natural order based on the absolute time of occurrence. Finally, location refers to the place within which an event occurs, whether explicitly stated or implicitly implied from context. Bal's concept of sjuzet[2] similarly consists of several factors. Where the fabula is made up of naturally ordered events, her concept of sjuzet states that events are ordered one after another in no particular order. The relationships between these events have several attributes including directionality and distance, where directionality refers to the relative ordering within which the events occur — roughly equivalent to analepsis and prolepsis — and distance

refers to the elapsed time between the occurrence of events and its impact upon the narrative's comprehensibility. Bal also introduces a concept of rhythm, which models the speed and pacing at which the events of the sjuzet are delivered, akin to a narrative tempo that can change throughout a reading. Bal also differentiates between the place defined in the fabula and what she calls 'space'. Where 'place' describes a measurable location within which an event or series of events occur, 'space' instead contextualizes the location as perceived by an actor (e.g., as seen through a character's perspective situated in the place, observing it, and reacting to it).

Regardless of the semantics of the terminology being used, all variations share a common theme of separating the chronological ordering of events (*fabula*, *histoire*, *story*) with the order in which the events are presented to the reader (*sjuzet*, *discours*, *plot*). This separation can be found in all forms of storytelling, including narratives within video games. For example, in the video game *Uncharted 2: Among Thieves*, the player begins in a dramatic sequence with the main character, Nathan Drake, awakening injured in a destroyed train carriage hanging off the edge of a cliff, requiring the player to make their way up the train and get to safe ground. Shortly after doing so, Drake passes out and the game transitions to a cutscene set significantly prior to the events just played. Eventually, after many hours of gameplay, the player returns back to play through the introduction scene but now in context, with the game continuing this time. This is an example of video games using the narrative device of prolepsis, where sequences from the future are shown in a non-chronological order to increase curiosity and suspense. The *fabula* here is the actual happenings of the events in the game's universe, and the *sjuzet* is the order in which the player experiences them, altered through narrative devices such as analepsis and prolepsis.

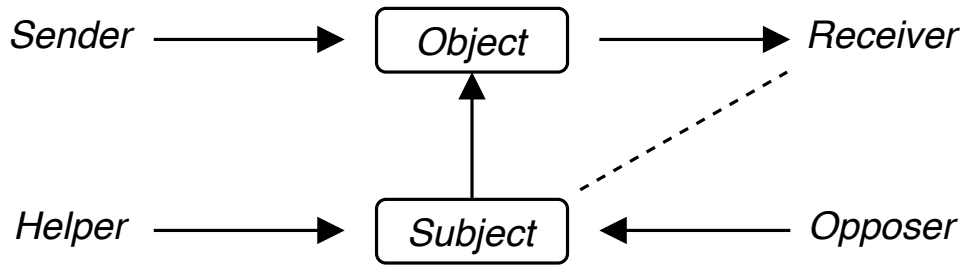
Prior to this, early efforts to break down and classify narrative texts were taking place. Antii Aarne, in his work *Verzeichnis der Märchentypen* [2], proposed a classification system for the indexing and grouping of tales based on related structures. This concept was refined and expanded upon by Stith Thompson [1] and Hans-Jörg Uther [214], evolving into what is today known as the *ATU-Index*. In this system, tales are divided into sections, each given a unique identifier and a thematic title either taken from the motif it represents or from a famous example within the category. Each entry in the index has a summary of the principal traits that make up the narrative feature, lists individual variations (between cultures, for example), and concludes with bibliographic information related to the entry. This categorization of narratives based on their shared traits allowed similarities between tales from around the world to be compared based on the motifs that they contained. Its primary use was as an index, where a particular subject could be looked up, a description given of the motif in question, and examples from various tales listed. This work was criticized by Russian Formalist Vladimir Propp [174], who claimed that the functions of the motifs were ignored in the classification process, and expressed concern that the

method of classification could result in some tales that begin similar but later greatly diverge being incorrectly classified together due to the macro scale of the classification.

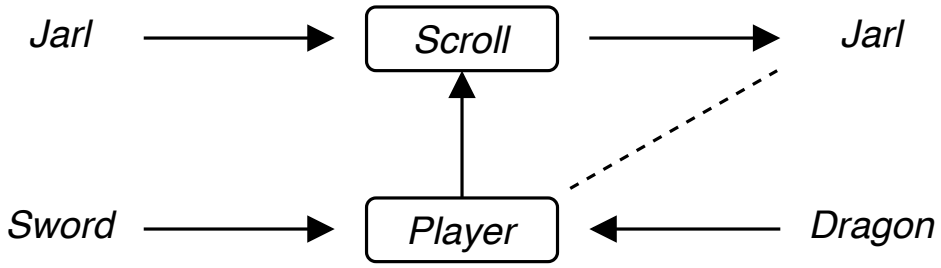
Russian Formalists had been working on their own ways of isolating and identifying minimal units of narrative, formulating principles about their combinatorics and interconnectivity. Vladimir Propp's *Morphology of the Folktale* [174] is a prime example of this effort. Taking 100 folktales and other reference material, Propp sought to describe them according to their constituents, their relation to one another and the overall tale, and to serve as a formal basis for comparison between different tales. The fundamental element of narrative in this model is the *function*, which is identified according to actions performed irrespective of the performing characters (similar to observing Aristotle's mythos irrespective of the ethos). Of critical importance was also the ordering of these functions, which Propp claimed while not always present would appear in identical order, the validity of which is challenged by capabilities of modern digital media. This approach of mapping units of narrative as functions into sequences with constraints is a syntagmatic analysis represented along a horizontal axis (just as words in a sentence are horizontally constrained to their neighbors) [33].

French Structuralists instead emphasized a paradigmatic approach, focusing on features embedded within the grammar of narrative. Greimas, for instance, built upon Propp's work in his *modèle actantiel* [91] (that he later refined [92]) which can be used to analyze actions within a narrative. The model is made up of six different 'actants', each of which represents a different component within Greimas' abstraction of a narrative event. Actantial analysis is performed by populating each of these actants with corresponding material from the narrative action in question. The actants and their relationship to one another are displayed in Figure 2.1a. The *subject* actant is the narrative entity that is being directed toward the *object* actant with a defined relationship state between them that is desired. The *helper* and *opposer* actants aid or hinder, respectively, the *subject* in achieving the desired relationship state with the *object*. The *sender* actant is the instigator of the action, requesting the satisfaction of the relationship between the *subject* and *object*. The *receiver* is the entity for which the relationship is being satisfied, which is often equal to the *sender*. As a complete example with all actants present, shown in Figure 2.1b: "The Jarl (sender, receiver) asks the player (subject) to retrieve (relationship) an ancient scroll (object) but warns of a guarding dragon (opposer) which he must defeat with his sword (helper)". The dotted line between *subject* and *receiver* indicates that often these two elements are equal in that an entity is working for benefit of itself, such as if the Jarl in the above example let the player keep the scroll, then both the *subject* and *receiver* would be the player.

Roland Barthes similarly analyzed the semantics of text, distinguishing between what he termed the 'readerly' text and the 'writerly' text [16], which he later refined in his analytical book *S/Z* [15]. Readerly text refers to literary works that place no burden upon the reader to produce meanings of their own. Barthes describes this kind of reader as



(a) A visualization of Greimas' actantial model highlighting each of the elements and the relationships between them.



(b) A populated actantial model of a fictional in-game quest where the player must retrieve a scroll guarded by a dragon with their sword at the request of a Jarl.

Fig. 2.1 A visualization of Greimas' actantial model (a) and a demonstrative population of a fictional in-game quest granted to the player by a Jarl (b).

passive, as they receive fixed and predetermined meanings from the text. This is achieved by adhering to the status quo in both style, content, and structure, and attempting to hide any elements that could open up the text to more than one meaning, referred to as the 'plurality' of a text. This loosely refers to traditional novels and other classical works, and as Barthes claims, "make[s] up the enormous mass of our literature" [15]. On the other hand, writerly text reveals those elements which readerly texts attempt to hide. Barthes describes the goal of this form as "[making] the reader no longer a consumer, but a producer of the text" [15], where the reader now takes an active role in the construction of meaning within a text rather than passively receiving predetermined meanings. These writerly texts achieve this by using techniques counter to those found in readerly, namely the deviation from the status quo, introducing instability in meanings, and a potential disregard for traditional narrative structures. Barthes goes on to determine what he considers the ideal form of text, blurring the distinction between reader and writer.

> "In this ideal text, the networks are many and interact, without any one of them being able to surpass the rest; the text is a galaxy of signifiers, not a structure of signifieds; it has no beginning; it is reversible; we gain access to it by several entrances, none of which can be authoritatively declared to be the main one; the codes it mobilizes extend *as far as the eye can reach*, they are indeterminable . . . the systems of meaning can take over this absolutely plural text, but their number is never closed, based as it is on the infinity of language." [15]

This description of Barthes' ideal text has strong links to hypertext fiction due to its typically non-linearity, support for arbitrary access to content, the inherent mystery in meaning brought about by these non-linear structures, and the active participatory role that the reader plays in the text. Hypertext fiction is discussed in more detail in §2.2.2.

2.2 Narrative Systems

A narrative system refers to any framework or system that predominantly concerns itself with aspects of narrative. Of particular interest to this thesis is interactive narrative systems, which can be said to have four characteristics [197] — potential use of multiple modalities (sound, text, video, graphics, animation, etc.) in their content delivery, integration of complex structures to control individual elements, strong interactions between the user and the system, and that they are based upon narrative structures that regulate how the constraints and use of the system develop temporally. The first two of these characteristics are said to be consistent with typical multimedia systems and the latter two distinguish narrative systems by means of storytelling and user interactivity. The vague classification of interactive narrative systems reflects the broadness of the field. Hypertext systems are those that associate themselves with modeling and delivery of hypertextual content to users in an interactive and dynamic way. These systems act as a foundation upon which interactive fiction systems are built, being an intersection of narratology and hypertext theory. As we will see later in this thesis, there is an overlap between contemporary video games and traditional interactive fiction, making them of interest to this thesis. Another type of system that is of tangential interest to this thesis is those that are generative, associated with automated creation of narratives based on constraints and author inputs. It is important to acknowledge these generative systems, but they are of less relevance as this thesis does not approach authoring of generative content but instead focuses on manually authored stories.

2.2.1 Hypertext Foundations

The first insight into what we today call hypertext can be traced to the theoretical proto-hypertext system *Memex* from 1945 by Vannevar Bush [35]. In his article describing the system, Bush speculates harnessing of computational power to build a physical device that can store, manage, and display great volumes of information such as books or other documents from libraries. The system would then be able to create and follow associative trails of links between pieces of content and allow for markup of the content, much in the same way that hypertext allows, albeit simplified. While *Memex* itself has remained a thought experiment, the concepts that it described have gone on to influence many fully realized systems [48]. The term 'hypertext' was introduced in 1965 by Ted Nelson to mean "a body of written or pictorial material interconnected in such a complex way that it could not be conveniently presented or represented on paper" [160]. This concept

was brought about in part due to Nelson's vision of a system, inspired by *Memex*, of a "digital repository scheme for world-wide electronic publishing" [50] that was set apart in how it embraced digital concepts for enhancing the reading and writing experience rather than mimicking real-world counterparts directly. This hypothetical system, named *Xanadu* [161], is considered the first true hypertext system having started development in 1960. However, to this day it has unfortunately yet to be fully realized as a software package and lacks widespread adoption despite the spatial and connective advantages it has over what is now the Web and various paradigms found in standard document processing software. Nelson later went on to clarify that "the 'hyper' in 'hypertext' is used in the mathematical sense of extension and generality . . . rather than the medical sense of 'excessive'. There is no implication about size . . . 'Hyper-' refers to structure and not size" [162].

Elements of Hypertext

A more recent description of hypertext from Jacob Nielsen [165] explains how hypertext is made up of fundamental elements which work together to create an experience for the user both in terms of reading and writing.

> "Hypertext is non-sequential writing: a directed graph, where each node contains some amount of text or other information. In most hypertext systems, a node may have several [outgoing] links, each of which is then associated with some smaller part of the node called an *anchor*. When users activate an anchor, they follow the associated link to its destination node, thus *navigating* the hypertext network. Users backtrack by following the links they have used in navigation in the reverse direction. *Landmarks* are nodes which are especially prominent in the network, for example by being directly accessible from many (or all) other nodes."

Nodes. A node in its basic form is a unit of information that is a part of a larger network of interconnected nodes, with the granularity at which it resides being defined by the system within which it exists. As many early hypertext systems were presented on low-resolution terminals, it was common to think of an individual node consisting of "a single concept or idea" such that it could "be displayed on one computer screen" [66] due to the limited screen space available at the time. On the other hand, even though contemporary to low-resolution terminals, Nelson's *Xanadu* [161] doesn't place restrictions on the size of a single node, allowing for theoretically infinite workspace for each node. The hypertext system *ZOG* [178] referred to its nodes as 'frames' with each containing various attributes (such as a title) alongside the textual content within. These frames were similarly limited by the available screen space and as such were often specified to represent a "structured screenful" of information, although later in *ZOG's* life this was no longer the case as it was able to render multiple frames to the window at the same

time, which added flexibility in the granularity of the frames created by users [143]. The later *HyperTIES* system improved upon the concept of frames by allowing for arbitrary multimedia to be embedded within, which increased the flexibility for users, but the underlying granularity remained comparable [193]. The *World Wide Web*, pioneered by Tim Berners-Lee, chose a simpler model by representing nodes of content as an individual page that could be explored with a browser. As with *Xanadu*, these individual pages are theoretically unrestricted in size, but also allow for rich multimedia experiences much alike *HyperTIES*. Although the node is a truly fundamental element of hypertext, these differences in what a node is demonstrates its ambiguity beyond a concept that is open to interpretation.

Landmarks. It is common to have a specialization of a node that acts as a landmark within a hypertext as it helps to ground the user within what could otherwise be a complex and confusing navigational experience [164]. Just as prominent structures in reality help to ground location, landmark nodes in hypertext help to frame the user's current location within a web of interconnected nodes. There isn't anything particularly special about a landmark node, but they are typically directly accessible from all locations within the hypertext [164]. Examples of landmark nodes include homepages, sitemaps, indexes, tables of contents, and so on.

Anchors. Nodes by themselves do not constitute an interactive hypertext, but instead must be interconnected to form networks that can be traversed by the user. Anchors provide the endpoints to make such connectivity and interactivity possible. In practice, as touched upon by Nielsen, anchors refer to a subsection of a node that represents some kind of meaningful element or region. Anchors can be defined at different levels of granularity dependent upon what the node allows as content. For instance, if a node allows only textual content, then an anchor could be present at paragraphs, sentences, or even individual words or characters. If a node allows for multimedia assets to be contained, then anchors could instead map to the individual assets or even regions within the assets in the case of images. The granularity of anchors inherently defines the level of interactivity that a user has with nodes. Anchors alone, however, do not constitute interactivity between nodes, nor do they connect them. Instead, anchors are used by links to tie together whole nodes or parts of nodes.

Links. Links, in their simplest form, are a linear connection between anchors either within the same node (internal) or between separate nodes (external). However, links are typically accompanied by varying attributes that increase the functionality and complexity of the connection. For example, a link between two anchors may have an associated conditional statement that determines whether or not, upon evaluation, the link can be activated. Nelson's *Xanadu* has a complex linking system that maintains bidirectional linkage between the origin and destination anchors, with additional attributes for links that handle authorship permissions and copyright among other things. On the other hand, the *Hypertext Editing System (HES)* [38] by Nelson and Andries van Dam used Nelson's

concept of a link but were considered 'flattened' as they were without as many attributes and were unidirectional which restricted the relationship between the nodes [219].

When links are made up of an origin anchor and at least one destination anchor, they are typically activated by interacting with the origin anchor. The granularity of the origin anchor, as mentioned earlier, therefore determines the level of interactivity that a user has with a system. The granularity of a destination anchor instead determines the scale that relationships can be defined at. In *Xanadu*, for example, all anchors can be applied at a variety of levels which enables it to support fine levels of navigation but also to support transclusion, transdelivery, and transquotation[3] of content between nodes. *HES* represented links as simple relationships between two pieces of arbitrary content, but also included a specialization which allowed for multiple links to be resolved as a branching user choice. It is also claimed that *HES* contained the first occurrence of a backward navigation button to traverse links in a reverse stack, which came about as a result of simplifying the link structure between content [219]. van Dam's later *File Retrieval and Editing SyStem (FRESS)*, built upon *HES* and greatly inspired by Douglas Engelbart's *oN-Line System (NLS)* [63], implemented links as either 'tags' or 'jumps'. The former were simple links between two anchors within a node for content that was aside from the main body such as references or footnotes, and the latter joined together arbitrary anchors (both internal and external) bidirectionally, similar to *Xanadu*. *HyperTIES*, allowing its nodes to contain rich multimedia content, allowed users to create origin anchors as either highlighted text links (referred to as 'embedded links') or as state-based regions on image maps. Due to the enhanced interactivity of the system, text-based links were often decorated on radial pie menus or as buttons. Destination anchors were typically whole nodes which could be placed within a split window panel. These embedded links became the inspiration for 'hyperlinks' found within Berners-Lee's *World Wide Web* manifesto[4], although they were implemented instead as simple unidirectional links between a fine-grained anchor and a complete node (a webpage). Some links are dynamically generated based on properties of the content, calculating potential origin and destination links in runtime typically upon interaction. For example, the *Intermedia* [148] system has generated links in its glossary lookup system which, upon user request for any word (what would otherwise be an authored origin anchor), dynamically establishes a destination anchor to that word in a dictionary. A similar example can be seen in *Microcosm* [49] which featured origin anchor generation based on presence of keywords with predetermined destination anchors.

Advantages and Disadvantages of Hypertext

As part of his survey of at the time contemporary hypertext systems, Jeff Conklin formulated and discussed nine advantages and two disadvantages of using hypertext systems [44]. The advantages of using hypertext systems can be summarized as:

1. **Ease of Tracing References.** Support for link tracing means it is equally easy to follow forward to a referent or backward to a reference.
2. **Ease of Creating New References.** Users can make their own network or annotate others without modifying the source documents.
3. **Information Structuring.** Both hierarchical and nonhierarchical organizations can be imposed on unstructured information.
4. **Global Views.** Inherent support for global overviews of document structure which supports easier reconstruction of large or complex documents.
5. **Customized Documents.** Text segments can be threaded together in many ways, allowing the same document to serve multiple functions.
6. **Modularity of Information.** Since the same text segment can be referenced from several places, ideas can be expressed with less overlap and duplication.
7. **Consistency of Information.** References are embedded in their text and if the text is moved, even to another document, the link information is still retained.
8. **Task Stacking.** The user can have several paths of inquiry displayed on the screen at the same time, such that any given path can be unwound to the original task.
9. **Collaboration.** Several authors can collaborate, with the document and comments about the document being tightly interwoven.

The first disadvantage that Conklin raises is *disorientation*, what he referred to as "the tendency to lose one's sense of location and direction in a nonlinear document". This problem arises from having to know where you are in the network and how to get to another place within the network. In linear texts, the reader is only able to go forwards or backwards, but in a hypertext network of many interconnected nodes, there is a greater potential for information to become hard to find or simply forgotten. The main reason for this is a lack of contextual clues as to location and lack of navigation aids for better traversal throughout the network. One of Conklin's suggestions to remedy this is to plot the nodes and links in either 2D or 3D space with use of visual properties (color, size, shape, texture) to differentiate between nodes, within which users can then orient themselves using visual cues just as when walking through a familiar city. However, he goes on to note the lack of a natural topography for hypertexts and describes the challenges of maintaining and understanding graphs as networks grow in scale. Existing systems have tried to resolve this by better enhancing navigation capabilities and increasing contextual awareness of both current location and the ability to navigate. For example, *HES*, as mentioned earlier, introduced the first occurrence of backward navigation of traversal in response to a discussion of how to keep the user oriented within the hypertext after they blindly jump between content [219]. Similarly, *FRESS* introduced backward traversal of otherwise destructive changes by maintaining a traversable stack of mutable steps taken. While debated, it is widely believed that *FRESS* is the first system to have implemented this ability reminiscent of modern undo [47, 52, 219]. We can also see examples of enhanced context within content itself, serving as navigational cues to

further assist in recognizing how and where traversal is possible. Examples of this can be seen in *HyperTIES* in which anchored text (i.e., the source of a link) was highlighted in a color different from regular text to indicate interactivity, and due to enhanced visual quality over previous systems, was able to present a radial menu upon activating the link. Additionally, some links had state-based colored backgrounds acting much in the same way that buttons do today. In the Web, we can also see this behavior with hyperlinks being highlighted in a different color and buttons indicating action. However, in the case of hypertext fictions, we may actually favor disorientation for the user as a narrative device. As Mark Bernstein explains, "writers of hypertext fiction adopt textual devices to disrupt reader expectations in order to shape the narrative experience, to confront the reader with challenges and puzzles, or to advance the narrative" [22], further listing such devices as deceptive orientation cues, violating expectations, fragmentation, disruptive novelty, defeating spatial analogy, and use of recursion.

The second problem that Conklin discusses is *cognitive overhead* which he summarized as "the additional effort and concentration necessary to maintain several tasks or trails at one time". In the context of authoring hypertext, an example is given where an author is writing about X and thought Y comes up; the challenge in hypertext authoring is then how one notes down Y without disrupting the process of writing X. Hypertext authoring systems should aim to alleviate this overhead by minimizing the friction of complex, hierarchical, or parallel tasks. A suggested solution is to support immediate recording of the substance of Y but defer creation and labeling of the node and necessary linking until later. In a node-based system, this could be realized in various ways such as by allowing the user, while editing node X, to make an arbitrary note (containing the premise of Y) which is automatically converted into an accessible node in the background. Alternatively, a node-based system could support multimodal editing where more than one node can be edited at the same time, in which the author could, again while editing node X, create a new node Y and edit its contents without losing context or disrupting the original editing process. Conklin also raises similar concerns from the perspective of a reader, a concept he terms "informational myopia", where the reader experiences distractions in regard to following links and the "cognitive loading" that appears when pausing to consider alternative pathways. He suggests three solutions, one being improving the response time, another being providing a synopsis of link destinations to ease the cognitive burden of choice by providing additional context, and finally by plotting the network's connectivity in a graph to show the reader where links lead, again providing more context to choices.

2.2.2 Hypertext Fiction & Interactive Fiction

Hypertext Fiction

In his seminal book *Cybertext: Perspectives on Ergodic Literature* [4], Espen Aarseth outlines the concept of *ergodic literature* and how it differs from its counterpart *nonergodic*

literature. It is said that "in ergodic literature, nontrivial effort is required to allow the reader to traverse the text", and with its counterpart, "the effort to traverse the text is trivial, with no extranoematic responsibilities placed on the reader except (for example) eye movement and the periodic or arbitrary turning of pages". As part of this work, Aarseth describes how a reader of typical literature has little to no power over the experience and is more of an observer than an active participant. Throughout the book, he raises the concept of *cybertext* within which the reader is no longer passive:

> "The cybertext puts its would-be reader at risk: the risk of rejection. The effort and energy demanded by the cybertext of its reader raise the stakes of interpretation to those of intervention. Trying to know a cybertext is an investment of personal improvisation that can result in either intimacy or failure. The tensions at work in a cybertext, while not incompatible with those of narrative desire, are also something more: a struggle not merely for interpretative insight but also for narrative control: 'I want this text to tell my story; the story that could not be without me.' In some cases this, is literally true. In other cases, perhaps most, the sense of individual outcome is illusory, but nevertheless the aspect of coercion and manipulation is real." [4]

While a full definition of cybertext is beyond the scope of this chapter, we can still draw parallels between its concept and the process of reading fictional literature in hypertext systems, commonly referred to as *hypertext fiction*. In both cybertext and hypertext fiction, the reader has to traverse the underlying content in a nontrivial manner requiring conscious choice versus passive observation. Additionally, the reader in both is also an active participant in shaping the realization of the story from its underlying structured content, given varying degrees of freedom as to how they traverse the space.

Hypertext fiction is characterized, just as regular hypertext, by nodes of content that are interconnected in such a way that they form a traversable network often (but not always) difficult to represent in a paper-based counterpart. However, in this context, the content is appropriate for the story that is being told, whether it be text, images, or other media. In hypertext works such as nonfiction books or encyclopedias, links are typically used to signify retrieval of supplementary information such as footnotes, references, or other lookups, and in some cases for navigation as found in table of contents, index pages, or general links to other nodes. In hypertext fiction, however, links carry a different weight, often representing travel between content that goes beyond the semantics of the text and instead works as a function of the content itself. For example, two connected nodes may be linked chronologically with respect to the story, perhaps representing traversal between locations described within the content of the nodes, or they could represent further exploring a conversation by attaching to a part of character speech.

The linkage between content within a hypertext fiction, as well as the methods and freedom of traversal provided to the reader, are integral to and defining of the hypertext fiction genre. The most straightforward is an *axial* layout, where content is connected in

a linear fashion from a defined beginning to a known end. They are easy for readers to follow as they are often chronological, but they offer no control to the reader in terms of story influence and result in the same reading each time. Sometimes these linear layouts are embedded within more complex structures. For example, in Michael Joyce's *afternoon, a story* — one of the oldest works of hypertext fiction, authored in *Storyspace* — the reader is instructed that they can, at each node, press the return key to take a predetermined path through the network which results in a fairly traditional narrative experience from start to end. By doing so, however, the reader misses out on additional information of the story, and arguably the attraction that sets hypertext fiction apart. For instance, if the reader follows the default path, the narrator remains nameless, but if they deviate from the path, they are likely to find out his name, Peter.

Perhaps the most notable structure is a *networked* hypertext, where nodes are interconnected yet with no dominant axis upon which one navigates. In this format, there is usually no fixed beginning (beyond a title page or introduction) nor designated ending, leaving the reader free to explore and experiment with the structure at their own leisure, building an individual reading based on their own navigational choices. Looking again at *afternoon, a story*, while the reader can follow a default path, they are expected to deviate from this and explore the world on their own by clicking on words within the text to navigate to otherwise missed out nodes. However, this can result in disorientation as the reader can quickly become lost, as discussed in §2.2.1.

Another example of a networked hypertext fiction is *Patchwork Girl* by Shelly Jackson, within which the reader is presented with a title screen and can then navigate around the network by clicking on appropriate links in an order that they wish. However, Jackson herself stated that she encourages readers and personally prefers to explore and navigate the hypertext using the map view of *Storyspace*[5]. A part of this networked map structure is displayed in Figure 2.2. The first two nodes from the top are the introduction and title page, after which the reader is presented with five links (highlighted in red in the bottom half of the graph) accessible from the title page's content. If instead the reader chooses to navigate using the map, then they have complete freedom not only to the visible nodes, but to content nested within.

John McDaid's *Uncle Buddy's Phantom Funhouse* also has a similar networked behavior, although the method of navigation differs slightly. As this hypertext fiction is made in Apple's *HyperCard*, it doesn't have a node view as found in *Patchwork Girl*, and instead separates out sections of the story into individual *HyperCard stacks*, which act as a metaphor for interconnected Rolodex-like collections. In the manual that comes with the story, it is explained that "there is no 'right' way to read it", and although it suggests starting with the *READ ME FIRST* stack, it notes that it "is only a suggestion" and that readers should "feel free to skip around, follow links, and build your own connections" of the story. If the reader does indeed follow the (suggested) instructions, they are presented with a 'home' page of sorts but are free to, at any time, browse another stack instead of

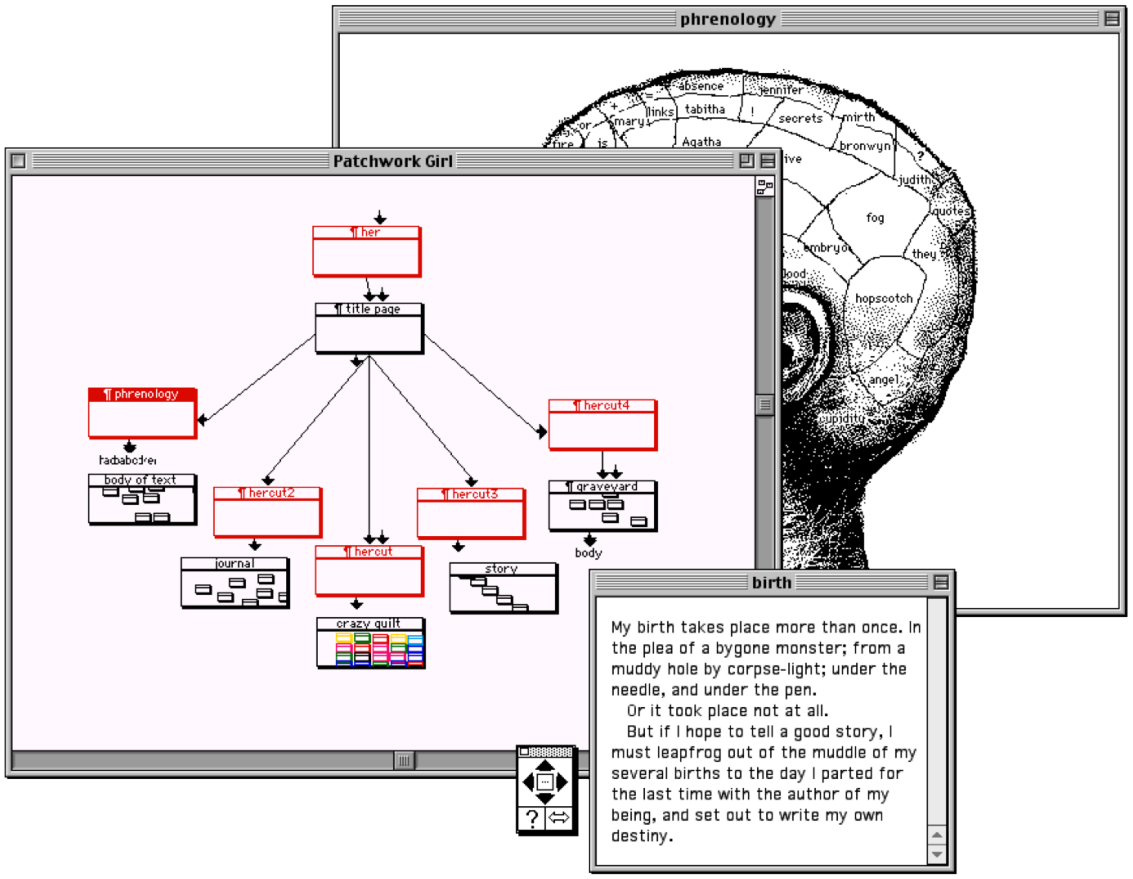


Fig. 2.2 High-level map view of the networked structure of Shelly Jackson's *Patchwork Girl* with windows for reading the hypertext fiction in the background and foreground.

following on-screen links, as shown in Figure 2.3, just as they are free to choose nodes in graph-based visualizations such as those discussed earlier. *Uncle Buddy's Phantom Funhouse* also made notable use of conditioned links within the network, with one particular stack accessible from the home page labeled 'Egypt' being password protected. If the reader attempts to access this content at any time, they are prompted for a password, which upon failure will reject them from viewing the content. As part of the narrative, the reader must first access a separate stack named 'Riddle' upon which being solved will reveal the password and enable the reader to progress the previously blocked content. This blocked content is in fact a mirror world of the original home page, serving as a narrative device to enhance the story, but only to readers who are willing to explore beyond conventional reading techniques.

Interactive Fiction

The actual term 'interactive fiction' was introduced by Niesz *et al.* in their aptly named article *Interactive Fiction* [168], within which they consolidate through discussion and example related forms of computerized fiction ('compunovel', 'participatory novel', and 'participa-stories') where the players (as opposed to readers) have levels of control over the story itself into a new single genre. However, there is critique of the term in regard to its vagueness and repeated widespread use without an agreed upon definition [4]. The

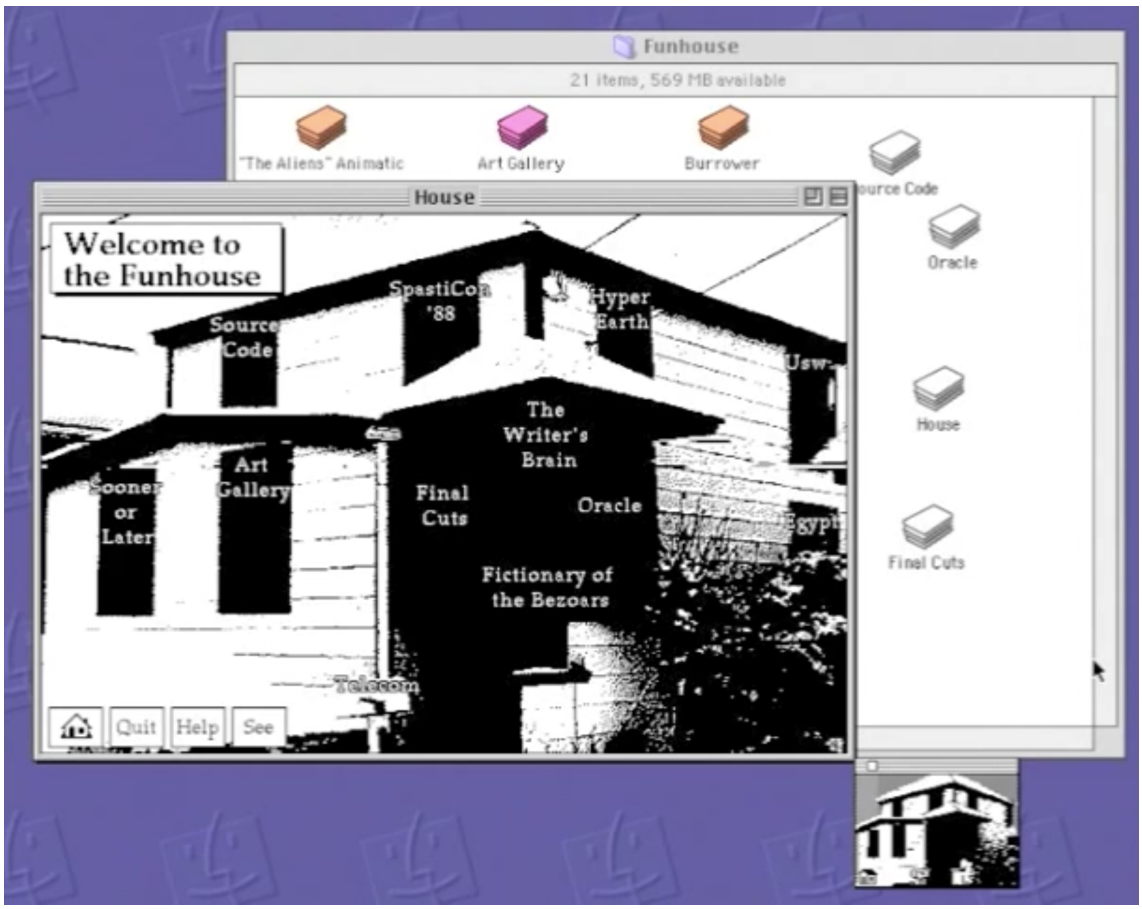


Fig. 2.3 A 'home' page in John McDaid's *Uncle Buddy's Phantom Funhouse*.

term has also been used in different contexts to mean different things, creating a confusion between some works being considered hypertext fiction or interactive fiction [155]. This is likely due to the apparent surface-level similarities between the two genres in that they both digitize and enhance the novel, they both cover a broad range of narrative content within which there is overlap, and both are described well by the concept of cybertext (or perhaps could be considered related subsets of cybertext). However, we can determine some characteristics of interactive fiction that differentiate it from hypertext fiction.

One defining feature of interactive fiction is that they maintain, to some capacity, a dynamic state of the game world that the player is exploring, which allows for significantly greater interaction beyond that typically found in hypertext fiction works. In the text adventure game *Zork I*, for instance, the player has an inventory which is tracked along with other various dynamic properties of the game world, which in turn creates the illusion of a living and breathing environment that responds to previous acts the player has (or has not) carried out. An example of this is the player's encounters with an infamous monster — a *Grue* — which if the player encounters without a source of light (such as a lamp) in their inventory, are likely to perish, thus ending the game.

Another differentiating factor is the tendency for interactive fiction — most often text adventures — to actively receive textual input from the player in the form of natural language and to translate that into actions within the game world. While this is not

impossible to add into a hypertext fiction, it is not something that is encountered often if at all, whereas it is a common trope of text adventures. The foundations of this interaction paradigm came from early natural language processing research such as *ELIZA* [216] and *SHRDLU* [220], the former simulating discussion with a psychotherapist and the latter allowing a user to interact with an artificial intelligence to query and build up a 3D world using natural language statements. This research would go on to influence early text adventures such as *Colossal Cave Adventure* and the aforementioned *Zork I*, among others, to make natural language processing the primary way of interacting with and traversing through the underlying network of nodes. Most of these interactions, however, were limited to simple verb-noun combinations, which inherently limited the amount and granularity of interactivity that players could express. If we wanted to take a sword and helmet then open a door, we'd have to issue three separate commands as `take sword`, `take helmet`, `open door`. A major enhancement for parser complexity came with the development of the *Z-machine* virtual machine by *Infocom*. The built-in parser introduced support for players issuing chained commands and more complex world-querying sentence structures to specify particular objects, much alike that found in *SHRDLU*. Rewriting the same example from earlier, we could instead say `take the greatsword and leather helmet and then open the back door`, which enabled the player to interact in a more natural sounding way. Beyond the improved parser, the virtual machine, which is still in active use today, enabled games to be written in a high-level language and to be easily ported to multiple platforms.

Works of interactive fiction, just like hypertext fiction, are capable of linear axial layouts, but by doing so do not utilize the capabilities of the genre and limit interactivity. Both the gamebook and visual novel subgenre of interactive fiction are commonly coupled with an *arborescent* structure. In these formats, there is one beginning and typically more than one ending, with the path in between representing a tree-like structure. Divergence along separate paths within the structure happens in the form of implicit choices as a result of world state (such as based on player inventory or previous actions) or explicit choices (typically in the form of dialogue options or a list of responses to an action). For example, in the visual novel *Everlasting Summer*, the player unintentionally travels back in time to a Soviet-era pioneer camp and must, through explicit choices, unravel the mystery as to how they got there and how to get back to the present. The choices that are made throughout the novel project the player along a selection of pathways, each resulting in a different ending. In some cases, accessing a pathway (and thus its ending) requires the game to be first beaten and then replayed before becoming accessible.

Text adventures are typically structured as a variation on networked hypertext structures, the difference being that players are given a dedicated starting point, usually have multiple defined end points (and even undefined dynamic end points such as by player death due to missing items or inadequate health), and rely upon parser-based user input for traversal. This is in contrast to the hypertext fiction networked layout, which as

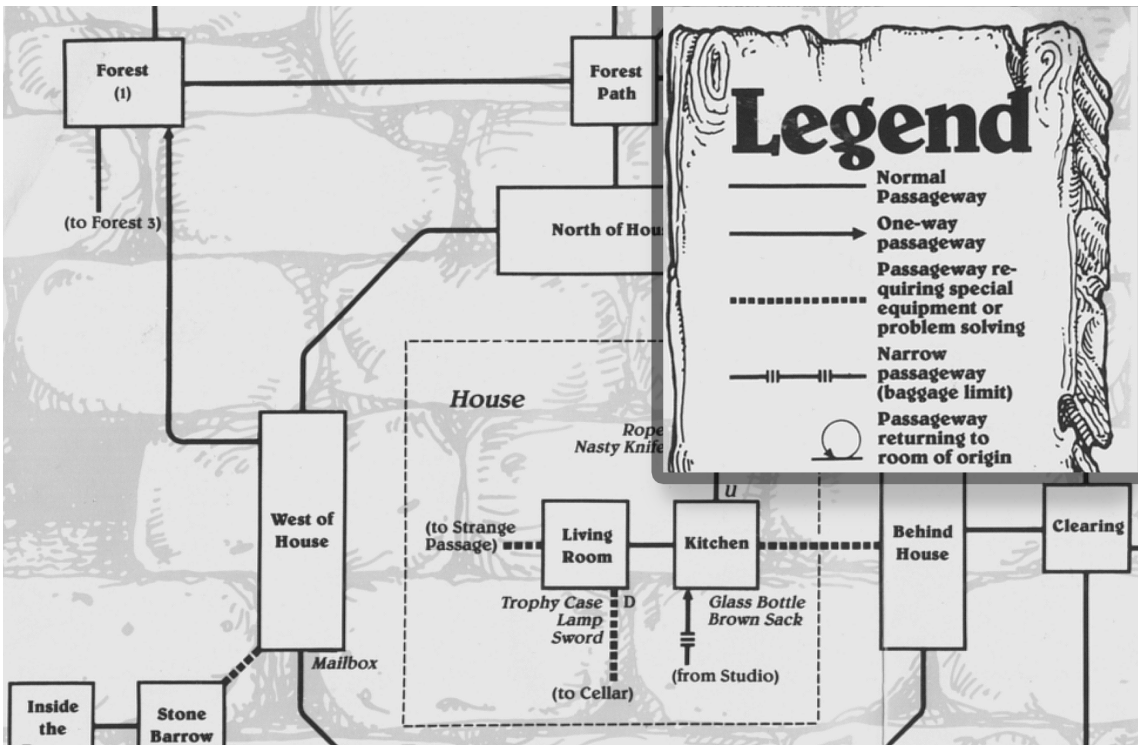


Fig. 2.4 A sample of the world map from *Zork I* in its *InvisiClues* book.

discussed earlier are frequently absent of a start (beyond a title page) and often encourage arbitrary traversal throughout the network at the reader's command. Additionally, traversal of network-like structures in interactive fiction works are often more dynamic due to the active simulation of a virtual world rather than interlinking of a novel's pathways. This can be demonstrated by analyzing a section of *Zork I's* map as shown in Figure 2.4 as an excerpt from its *InvisiClues*[6] book. This map shows that the hypertext nodes actually represent physical locations and that traversal between nodes is analogous to physical movement between the locations in the game world. Consequently, the rules for traversal throughout the network are complex as to gate progression as would be in reality. In the case of the *Zork I* map, pathways are bidirectional by default (such as between *Living Room* and *Kitchen*) with unidirectional being used to constrain movement (such as being unable to return when going from *West of House* to *Forest*). Certain pathways are guarded by dynamic in-game checks such as when traversing from *West of House* to *Stone Barrow*, which requires the player to first collect and correctly place all treasure from around the world within a trophy cabinet to receive instructions that provide access to the location. Some other checks are based on in-game stats such as coming up from *Studio* to *Kitchen*, which requires the player's baggage (inventory) to be within a certain threshold or to contain limited items, simulating a narrow passageway. It is not that hypertext fiction is incapable of such mechanics, but that these tropes are found chiefly in interactive fictions and therefore help to distinguish them from one another.

2.2.3 Computational Narrative Systems

The process of authoring a digital narrative experience involves cooperation between computer systems and human authors. This is referred to as a *mixed-initiative* approach with the balance between human-centered authoring and system-based generation varying in degree. It has been found that mixed-initiative approaches are typically most successful when the human author retains control and the system provides intelligent and automated services for the author rather than replacing them [135, 158]. A survey of computational narrative systems by Kybartas *et al.* [120] proposed mapping the degree of automation in human-computer authorship of *space* and *plot* onto a set of orthogonal axes. In this context, *space* refers to "characters, settings, props, and anything which is present either physically or abstractly in the space of the narrative" and *plot* refers to "a set of events with an overall structure which represents both the temporal ordering and the causal relations between the events". This mapping can be seen in Figure 2.5, where the four colored quadrants demonstrate broad categorizations of the types of cooperative balance between human authoring and computer systems' generation.

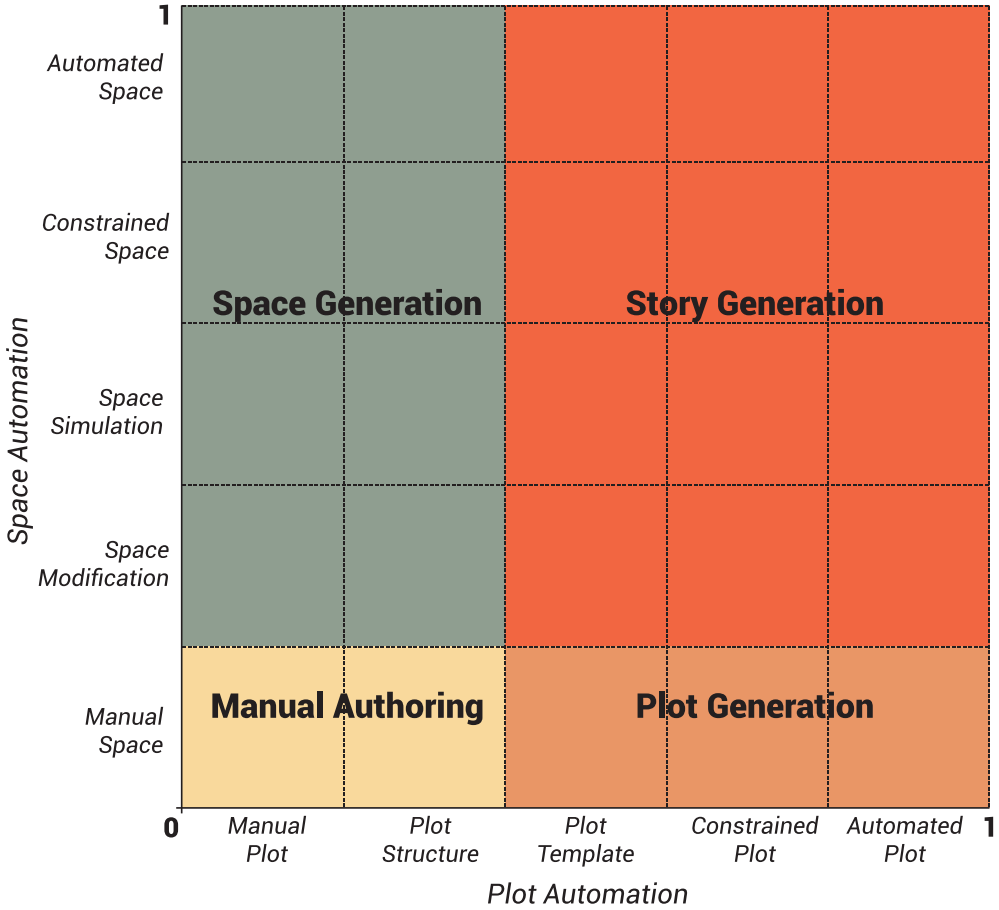


Fig. 2.5 Kybartas *et al.*'s story automation matrix with broad categories [120].

The horizontal axis of the graph represents the degree of automation of plot ranging from zero (maximizing human authorship with minimal system involvement) to one (minimizing human authorship with maximum system involvement). This is further divided into five categories that represent approaches to plot automation along this scale.

Manual Plot refers to the passive involvement of systems where the human remains entirely in control of the plot and the system may enhance the authoring experience but will not generate anything. *Plot Structure* systems provide the structure of the plot to the author but do not detail events or orderings that make it up, constraining the author to a predefined structure but freeing them to specify what actually happens. *Plot Template* refers to systems are an extension of the previous category where the events themselves are also generated, but the human author is left to determine what characters and objects are involved in the events. *Constrained Plots* are fully generated by the system with human input being reserved for the underlying rules and constraints that regulate and determine what is generated. *Automated Plot* refers to systems that minimize human authorship as much as possible to completely generate the plot structure, contained events and ordering, and the constituents of the events themselves.

The vertical axis represents the degree of automation of space, also ranging from zero to one, and similarly subdivided into five categories. A *Manual Space* is equivalent to *Manual Plot* in that the system is passive and that the author is expected to create the space themselves without generative aid from this particular system. *Space Modification*, on the other hand, expects an entirely authored space but allows the system to modify elements of the space, typically to improve plot generation. *Space Simulation* again expects an authored space, but the system is able to generate new content for the space by simulating interactions between elements within the space. *Constrained Space*, similar to its plot counterpart, expects rules and constraints to be provided by the human author which it then uses to generate either in part or in full a space, typically including the elements with the space too. *Automated Space* again refers to systems that attempt to minimize human authorship to completely generate a space and its inhabiting elements.

Any point on the graph describes and classifies a system that has a certain degree of automation for both plot and space. I will briefly outline the prominent combinations below from each section, but I refer the reader to the original survey [120] for a more in-depth discussion as that is beyond the scope of this thesis.

Manual Authoring

Manual authoring systems are those that, for the purposes of creating digital narrative experiences, give the author complete control over the plot and space with minimal intrusion from the software system (typically in the form of assistance rather than generation). A large number of hypertext fiction and interactive fiction authoring tools fall into this category having the attributes of manual space and manual plot. In these systems, many of which are discussed in §2.4, authors are unrestricted in their authoring of plot and space, which allows for experimental structures and a wide variety of works to take place. Some manual systems, while expecting the author to create the space and plot, do enforce a general structure that the created narrative must fit within. Examples of this can be seen in screenplay authoring tools such as *Scrivener* and *Fade In*, the former

providing templates for the user to adhere to (for short stories, screenplays, novels, etc.) and the latter restricting the author to tropes of screenplay structure (scenes, actions, shots, etc.). Less restrictive yet similar behavior can be found in *INSCAPE* [12], whose interface consists of a graphical 'Stage Editor' for defining the narrative's space and a 'Story Editor' used for plot and ordering of the stages, which together inherently apply a visual scene-based structure to the narrative. *StoryTec* [79], which took inspiration from *INSCAPE*, also features editors for stage and story, likewise imposing a specific structure.

Plot Generation

Systems that fall into the plot generation category are those where the author is responsible for the space within which the narrative takes place, but the plot is automated to varying degrees. In the case of *Plot Templates*, sequences of events are generated but without any content, typically appearing in the form of a plot grammar. Propp's Morphology [174] has a notable history in this space, with Propp himself suggesting its use for generation by stringing together variations of his plot functions.

> "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↑, 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." [174, 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 [183]. 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. Further work has been done using Propp's Morphology as a grammar for generation [76]. 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 [37] and Lakoff's Structure of Fairy Tales [121].

A *Constrained Plot* in this context is one that is fully generated and populated, but the human author still provides constraints to the generation such as character/object selection rules, rules for causal or temporal relations of events, desired states that must be true at a given time, or general goals for inhabitants. An example of this is *TALE-SPIN* [146], which is considered one of the first story generation systems. The author provides characters within the simulated space, each with unique personalities and goals. The system then simulates events surrounding each character's attempts to achieve their own goals, resulting in a highly emergent narrative derived from author input. While the space itself is authored, the system can modifiy the space to ensure that characters can achieve their goals. Another related approach is planning-based plot generation which aims to yield a coherent story given goals and constraints. Goals vary but are typically desired states of one or more inhabitants of the plot. It's not uncommon to have multiple goals, subgoals, or even dynamic goals. Goals and other constraints typically come from the user to indicate their intentions to the generative system. The *Story Canvas* [196] authoring tool, for example, lets the author specify goals and rules for plot generation at a high level using hierarchical task networks, which are then converted and generated for each of the inhabitants of the story into behavior scripts. Furthermore, we can find a type of constrained plot generation in generative interactive narratives. Take *Façade*[7], for instance, which uses authored plot constraints to ensure that certain dramatic situations occur despite taking place within an authored space and being dynamic based on player input [142]. *IDtension* [206, 207] demonstrates a similar premise where authors specify attributes such as moral values between characters which then aid the generative plot to ensure dramatic tension occurs as readers progress.

Space Generation

Space generation systems are for when the author knows the events that make up the plot but not the space within which they take place. Within the context of narrative generation, this area is less explored as the focus tends to be on the generation of events and their relationships. Despite this, there are examples of systems that generate constrained spaces with authored plot structures. The *ScriptViz* [133] system, for instance, constrains the author to typical screenplay formats, and then based upon an appropriate input can generate a full 3D visualization of the space that is being described in the plot. Another example can be found in the *Radiant Quest System*[8] of *The Elder Scrolls V: Skyrim* where authors can manually write plots in the form of quests, but only roles are specified as opposed to concrete instances of individuals within the plot. The actual in-game realization of the quest is generated dynamically, filling in roles with characters and items that already exist, or creating and instantiating new ones where this cannot be done. These quests may be repeatable, and each time would vary due to the generative nature of their realization, including altering the location, often favoring places that the player is yet to discover to encourage traversal of the game world. Charles *et al.* [39] also

used space generation to increase the accessibility of patient education documents in a medical setting by processing said documents and realizing them in a generated 3D visualization using a game engine. The space in this system is generated based on the actions and interactions the patient will be able to perform, ultimately decided by the input patient education documents including locations, medical procedures, objects that populate the environment, and present characters.

Story Generation

Some systems attempt to further reduce human authoring input, at its extremes to an absolute minimum, by automating partial or even complete generation of both the plot and the space within which it takes place. Sometimes the generated plot, such as a *constrained plot* where the author contributes rules and goals, may benefit from modifying the space within which the generated plot takes place, particularly if it benefits the quality of the generated plot. For instance, the *Virtual Storyteller* [205, 208], which poses itself as an 'automatic story generator', generates plots based on authored scripts (which contain information about settings, goals, and constraints). The system introduced a concept termed 'late commitment' which allows modification of the world space in real-time to enhance the plot [204]. An example is given where a pirate ship is attacked by another ship, stating that based on a variety of goals, the system may determine that the plot would be more interesting if the ship was loaded with gold, and as such manipulates the world space accordingly. While procedural generation of space is not new to video games, we can see automated story generation appear in some circumstances. *Dwarf Fortress*, for instance, generates its world space procedurally by taking constrained input from the player (world size, length of history, rarity of minerals, etc.) and then generating and simulating the world space over long periods of (theoretical in-game) time, resulting in complete worlds populated with inhabitants with unique history. Certain characters may also become famous and the heroes who defeat them in turn become famous also. From this perspective, the game generates a world space through constraints provided by the player, but as a part of that process also generates the vast history of the inhabitants of the world, resulting in a plot prior to the player joining.

2.3 Video Games Telling Stories

There has been much debate as to the capacity or even capability of video games to act as a narrative medium. This section describes this debate and then briefly outlines the various ways in which video games can in fact deliver narrative experiences.

2.3.1 Ludology & Narratology

Despite being a relatively new field, the area of video game studies has been a source of heated debates surrounding its legitimacy as a field and its relation to other fields. As Murray explains [157], the problem arose from the claims that video games are a discreet field explicitly disconnected from narrative, drama, poetry, and other forms, and that attempts to impose such fields onto the study of video games were considered 'colonialist' in nature [5]. This resulted in a separation between the study of video games and other cultural genres in the methods used for study, which gives the false impression that games are explicitly disconnected from other cultural genres [157]. Brand *et al.* [30] recite how as video games became an object of academic study, the discourse surrounding them was constructed as an opposition between ludologists and narratologists (a phrase popularized by Frasca [72]), chiefly on the idea that games are optimally a form of narrative or play-based media.

Ludologists held a position that seemingly dismisses the narrative dimension of games as secondary, arguing that they should be analyzed from a ludic perspective. Ernest Adams claimed that interactivity in games is almost the opposite to narrative in that narrative flow is reliant upon the author while interactivity depends upon the player, and that many game worlds remain static and dead until the player arrives[9]. A similar rhetoric is delivered by Costikyan who explains that there is an immediate conflict between the demands of stories and games, focusing chiefly upon divergence from predetermined story paths resulting in a less satisfying story, yet restricting player freedom to this path resulting in a less satisfying game [46]. A more extreme position was taken by Juul who claimed that games are "simply not a narrative medium" [105], although he has since retreated from this, studying games from both sides while still maintaining a critical dissonance between them [106]. Narratologists, on the other hand, favored the application of existing narratological frameworks (such as theories of hypertext, interactive cinema, and narratology) onto video games, treating them as another medium capable of portraying narrative, or even as a direct form of narrative. That said, Jenkins highlights that attempting to map traditional narrative structures onto games does so at the expense of attention to what makes them different from other types of media [103].

Reflections on this matter [26, 71, 157, 198] demonstrate that the debate was built upon a foundation of misunderstandings between both parties. It is now widely accepted that video games are not a subset of narrative, but instead that they contain elements that have qualities suitable for analysis from both the perspective of game studies and more traditional narrative theory. This is demonstrated by Aarseth's work where he explains how video games are complex software capable of emulating other mediums, including those that are considered narratological, such as film, graphic novels, and text novels [3]. In the end, it is not for one group nor another to determine what is and is not appropriate for the study of video games, and with these kinds of debates, it is likely that there is truth in both sides [157].

2.3.2 The Game Narrative Triangle

There are many ways that narrative can find itself interwoven into video games, but typically they are placed into one of two high-level categories: those authored by the developers and those created by the players. In his 2000 Game Developers Conference talk[10], Marc Leblanc termed these concepts as 'embedded narrative' and 'emergent narrative'.

Embedded narrative refers to the systems in a game that are intentionally authored by developers working together as intended to build a narrative experience for the player. Some elements of video games clearly present narrative in a purely embedded form such as cutscenes, scripted in-game sequences, and authored dialogue. In all of these embedded examples, the player has no control over how the narrative sequences play out.

Emergent narrative is a topic that has received significant attention in the academic space [24, 41, 116, 134, 184, 186], particularly in the classification, theory, design, and implementation of emergent narrative in video games, details of which are beyond the scope of this chapter. For the purposes of this discussion, emergent narrative comes from the player's interactions with various subsystems of the game, such that the player recognizes, interprets, and even creates events that are part of a larger narrative not directly intended by the developers. Purely emergent systems are uncommon, as some form of authored aspect almost always comes into play. For example, a player may direct their avatar and have their corresponding actions contribute to an emergent narrative, but the player may act within a predefined set of gameplay rules, assets (animations, sounds effects, and so on), and narrative context that is authored in an embedded way.

Others have purposed further dividing emergent narrative into two subcategories — that forming as a result of the player, and that resulting from the seemingly autonomous complex interactions of the game's subsystems[11]. This division was originally labeled as 'player' and 'computer', but I will instead refer to them as 'Direct Emergence' and 'Indirect Emergence' respectively, as I believe it better captures the involvement or lack thereof of the player in the instigation of the emergent process of the narrative sequences.

This now three-dimensional classification can be organized into a triangle as shown in Figure 2.6. As earlier alluded to, it is uncommon for a narrative sequence or aspect to be entirely at one pole of the triangle. Instead, we can plot points on the triangle using barycentric coordinates to represent narrative sequences that weight between all three poles individually. Plotting a narrative sequence on this diagram allows for a high-level overview of the amount of authoring effort by developers (*Embedded*), the degree of inclusion the player has in the sequence (*Direct Emergence*), and the degree of autonomous behavior that caused the sequence (*Indirect Emergence*). Additionally, plotting multiple narrative sequences on the triangle allows for a rough classification of similarities, or as a design tool to help fine-tune the experience.

To better understand the differences between these poles which represent modes of narrative delivery in video games, I will briefly discuss aspects of *The Elder Scrolls V: Skyrim*. If we were to describe the story of the game, we would discuss the player's

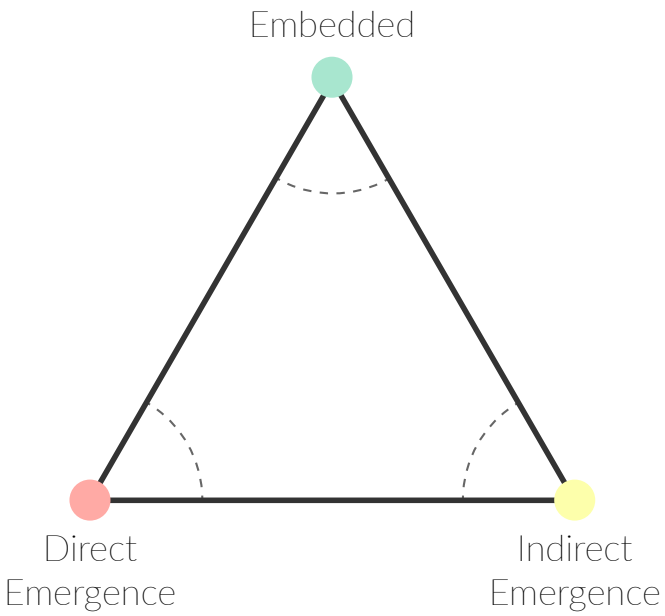


Fig. 2.6 A triangle of *Embedded*, *Direct Emergence*, and *Indirect Emergence* narrative methods in video games. The barycentric coordinates of a point on the triangle can describe a weighting of all three types of narrative for a given use case.

character rising from death's door to discover they are the fabled 'Dragonborn', engaging in an epic quest to ultimately stop the return of dragons. We would also discuss the various guilds that the player can join, the schools they can attend, all of the quests that they get sent on, and all of the deeply intertwined lore found in books and scrolls. These are examples of *Embedded* narrative, as they are directly created and intended by the game's developers. If we were to instead look at what the player does during the game, we may mention them encountering a village and talking to the locals to learn about the area or stumbling upon ancient ruins of past civilizations and investigating what was left behind. These are examples of primarily *Direct Emergence* mixed with *Embedded*, as the player instigates the narrative that comes about, but within the framework of actions set up by the developers. Finally, if we were to discuss what happens in the game's world irrespective of the player's presence, we would bring up the active villages with dynamic and interacting non-playable characters, we may see guards chasing down a criminal, or the player may randomly stumble upon hunters tracking animals. These are examples of primarily *Indirect Emergence* mixed with *Embedded*, as they are the result of the subsystems within the game interacting as intended, but without those specific outcomes being authored by the game's developers, all without direct player intervention. Some narrative sequences can take advantage of all three poles, such as the player, while exploring the wilderness, being approached and attacked by a group of bandits due to a bounty on the player's head. This has aspects of *Embedded* (the framework within which the actions take place, the assets and resources, the scripted behaviors), *Direct Emergence* (direct involvement of the player in partially instigating the narrative sequence, and the player's own interpretation of such a sequence taking place), and *Indirect Emergence* (the

simulated systems of the game working to transition from roaming to identifying and attacking the player, partially causing the sequence to take place).

2.3.3 Complexities of Video Game Narrative

This section provides a set of demonstrated examples that highlight the complexities that working with video game narrative structures brings. While many of these aspects may not be unique to video game narratives, their presence remains a key factor of what video games are and consequently must be considered when developing models.

As described earlier, video games are capable of emulating a wide variety of other narrative forms and consequently they integrate or otherwise overlap with techniques from other fields. Take, for instance, the mobile game *Pokémon Go*, which sees players use their mobile phone cameras and GPS systems to actively locate, capture, battle, and train Pokémon in a mixed reality setting, overlapping with several aspects of locative narrative experiences such as *Shelley's Heart*[12] created in *StoryPlaces* [98, 149]. For example, both rely upon location-aware content delivery, allow the player to exert limited agency over the narrative flow by interacting with such content, and provide mechanisms through its locative nature and enhanced agency to experience the narrative in multiple ways even when replaying. Further comparisons can also be drawn between the fictional *Choose Your Own Adventure* [43] book series and the early video game *Dragon's Lair*, which both present the narrative as a mostly uninterruptible sequence but then use the narrative device of reader or player choice to divert the experienced narrative. In the former, the reader progresses through segments of linear narrative that cannot be altered, being periodically given a set of choices that typically represent actions or outcomes which allow the reader to exert basic agency over the narrative such that two unique readings are unlikely to be the same. In the latter, the player is presented with sequences of linear movies as the player's avatar navigates the game world, periodically being interrupted by series of quick-time events[13] which appropriately direct the player along alternative narrative threads based on their response, some even terminating the narrative with a premature character death, at which point the narrative rewinds and the player can attempt the interaction over for a more desirable outcome.

Narrative Presentation Methods

As established earlier, video games are often said to contain elements that have narrative qualities, either being authored as embedded narrative, or resulting from direct or indirect emergence within the game world. Moreover, they can demonstrably emulate a wide range of other narrative forms. It is then not surprising to learn that video games use a vast range of narrative devices for the portrayal and presentation of narrative content to players. It is critical that we acknowledge these practices as they are ultimately affordances of video game narrative that must be considered when creating models.

Cinematics. Games often deliver pre-rendered sequences of narrative through streamed full-motion video (FMV) playback. During playback, player control is almost entirely restricted. Cinematics are often used to highlight specific moments of narrative significance such as introducing the game to provide context for the gameplay proper. In some cases, real-life video footage is delivered as FMV sequences to fill the narrative. For instance, in *Wing Commander III: Heart of the Tiger*, the space combat gameplay was intertwined with live-action footage. This behavior can also be found in the aforementioned *Shelley's Heart* which integrates real-life video footage when discovering a locative trigger. As the technical cost of real-time rendering continues to diminish, FMV cinematics are being increasingly replaced with in-engine sequences that fulfill the same purpose, but still remain a utilized narrative device.

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. In some cases, cutscenes directly suspend player control to show an in-game narrative sequence, much alike cinematics, with the difference being that they are self-contained within the game engine. An example of this can be found in the original *Pac-Man* where gameplay was periodically suspended to show brief comical interludes involving the characters. In other cases, the player's controls or the playable area are restricted while a narrative sequence is delivered. For example, at the start of *Call of Duty: Modern Warfare 2*, the player is faced with an impassible destroyed bridge and is forced to wait with restricted controls in a limited play area for an alternative path to be created, during which the player and their squad observe a danger-close airstrike over the city they are about to progress into, which adds believability and increases contextual depth as to the militaristic sense that the player has. A third cutscene variant introduces player agency over an otherwise restricted form of narrative delivery through the introduction of, at key points, the ability for the player to make decisions that alter how the cutscene progresses. The effects of such interactions typically have no longstanding impact but do give a greater sense of agency in the steering of the narrative being displayed. For example, during a cutscene at the end of the first mission of *Mass Effect 2*, the player needs to grab the attention of a spooked character by either doing nothing and letting the cutscene proceed, or by using a paragon (good) or renegade (bad) interrupt during the cutscene to gain the character's attention by either shutting off screens or shooting them respectively. This interrupt provides limited agency over how the spooked character is made aware of the character's presence, subtly altering the narrative but without greater consequence. In other cases, however, the interrupts may have a longer lasting significance.

Loading Screens. Due to the inherent complexity and size of video games as a software, they frequently have to load content. While some games simply inform the player that the game is loading, others opt to make use of the loading time to present narrative to the player. For example, in *BioShock*, loading screens are styled according to the game's art deco theme with a rotating set of contextual quotes from characters involved in the

story that serves to enrich the game world. Another practical example is the integration of cinematics into loading screens such as how *Call of Duty: Modern Warfare 2* uses its mission briefings, which serve as a narrative prelude to the gameplay, to disguise loading screens with the presentation of narrative content. In other cases, the loading screen and narrative delivery are more implicit such as in *Mass Effect* where they are often disguised as elevator rides, during which a radio can update the player on the happenings within the game's universe, sometimes in response to a recent player activity.

Optional Content. 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 narrative content may or may not be found by players, and consequently as the certainty that a player will experience it is reduced, it is usually reserved to provide additional background narrative for those willing to seek it out. This kind of narrative delivery is popular in open-world games by hiding books, letters, and other similar items, each of which provides some kind of further insight into the game world. For example, in *The Talos Principle*, the player can find painted QR codes painted around the world, which when either highlighted in the game or scanned with a QR decoding application present the encoded text, typically enriching the backstory of the characters that were present prior to the player's arrival.

Environmental & Indexical Storytelling. The concept of environmental storytelling was popularized by Don Carson[14] where he proposed translating lessons learned from theme park and ride design into game environment design. Carson explains that "one of the secrets [of designing] entertaining themed environments is that the story element is infused into the physical space". Many of the design strategies that he outlines focus on aiding navigation and understanding of the physical space, altering the park's structure and design to enhance the experience of the visitor and ultimately the narrative that they experience while in the park. One such strategy is what Carson describes as *Cause and Effect* — a way of generating interpretative narrative with mise-en-scène. Environments can be purposefully staged to invite the player to build their own conclusions of what happened prior to their arrival, or to unravel potential warnings of dangers ahead. Building upon this concept, Henry Jenkins enumerates several methods for how game design and gameplay can become a narrative experience [103], with particular attention given to *evocative spaces* and *micronarratives*. The former refers to evoking narrative based on a player's existing knowledge rather than exclusively through the environment alone, such as how environments in games based on a book series may evoke specific narrative interpretations different for those who have read the book to those who have not. The latter refers to short scripted vignettes of narrative that are integrated within the game, typically in response to player actions, which serve as 'narrative hooks' and work together to shape the overall narrative experience. Environmental storytelling as a concept has received attention within the games industry[15] and has since become a crucial skill within a game designer's toolbox for creating enticing narrative experiences.

More recently, the concept of environmental storytelling has been further analyzed and incorporated into the theory of *Indexical Storytelling* by Clara Fernández-Vara [65], which describes construction of narrative experiences through 'indices' both on part of the game designer and the player. It is described as a refinement of environmental storytelling as it "specifies the ways in which story and game are integrated together based on leaving traces in the gameworld". Indices are derived from Charles Peirce's philosophy of language [173] in that an idea is physically connected with signage, such as how smoke indicates fire, or how a signpost's orientation aids navigation. In the context of game narrative, an index can act as an indication to the player about what could or should be done in the game, such as strategic placement of environmental props or textures that provide a sense of directionality or capability to the player. Alternatively, an index can instead serve as indication of what happened within the game world that the player currently inhabits prior to their arrival, such as bullet holes and blood splatters indicating combat and danger. Fernández-Vara outlines two prominent forms of storytelling in game narrative that *Indexical Storytelling* captures well — history of the game world and history of the player.

History of the game world refers to indices that are authored by developers, representing previous events which have transpired prior to the player's arrival, leaving a trace in the physical space that the player navigates, encounters, and eventually interprets. For example, the adventure game *Myst* is designed around the concept of authored indices to lead the player into uncovering the narrative of the game world as they play. The player is placed on an island with little to no prior information, and as they traverse around the island, they must figure out the various scenes and contraptions that they encounter, gradually piecing together and solving the overarching plot of two brothers being trapped. In this sense, the authored indices left for the player to find define the history of the world which ultimately drives the player's narrative experience.

A common implementation of environmental narrative takes this concept and focuses it into smaller narrative vignettes. That is, authored indices are setup to specifically invoke a sense that other agents were present in the environment before the player arrived, which the player can use to build their own interpretation of the intended narrative. While these indices often stand alone as micronarratives (in the Jenkins sense), they can also work together to help shape the overall feel of a location or otherwise contributing to the overall narrative experience. The narrative of *BioShock* is largely conveyed through this form of *Indexical Storytelling* by decorating the environment with a multitude of micronarratives such as blood-stained walls, mise-en-scène, and discoverable audio logs left by those present in the game world before the player's arrival. Sometimes these micronarratives can work together to contribute to the greater narrative or some underlying theme throughout the game world. For example, when the player first enters the underwater city of Rapture, they are greeted with a gigantic figurehead bearing the text "No gods or kings. Only man.", which is a mantra repeated throughout the game,

providing a sense of dismissal of religion in all forms within the game's narrative. Much later in the game, the player encounters a bloodied corpse hanging on a wall imitating crucifixion with "Smuggler" painted above. Shortly after, the player discovers that the smugglers were bringing in crucifixes and bibles which are forbidden in the city. While these individual narrative vignettes are physically separate, the player is able to interpret them as related and build their own understanding of the events that had transpired.

History of the player is less frequent, referring to devices that allow the player to construct and leave their own narrative indices rather than only have the story dictated to them. For this to occur, their impact on the game must be persistent either within their own game world or one shared with other players. For instance, in text adventure game *Colossal Cave Adventure*, the player encounters a complex maze where all rooms have an equally vague description. In order to differentiate between the paths trodden, the player can drop items in each location, leaving behind a trail to help identify where they have been. These persistent objects aid in solving the puzzle, but also create a player-driven narrative through indices reflecting the player's previous actions within the game world. As described in *Optional Content* above, *The Talos Principle* allows the player to discover QR codes painted around the game world, displaying a message from characters present before the player had arrived. If the player discovers paint buckets, they can leave their own messages in the game world which people on their online friends list can discover in their own games. In this case, the player's indices are merged with the authored indices to create a richer narrative background and a sense of importance of player actions.

Narrative Interaction Methods

As with presentation methods, video games involve a wide range of interaction methods which often give the player some level of agency over directing the narrative flow. In *Fundamentals of Game Design* [6, p. 170], Adams describes the concept of immediate, deferred, and cumulative influences that the player can have on the narrative as in-game events take place, usually as the result of player choice or interaction. Immediate choices have a direct and instant alteration to the narrative, typically as an irrevocable decision that branches the narrative along a distinct path. 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 over time, that ultimately result in some divergence of narrative.

Narrative Mechanics & Mechanics as Metaphor. The term 'game mechanics' is an elusive concept that remains without a standardized definition. There have been attempts, particularly by Miguel Sicart, to form consensus through extensive review of existing (and conflicting) definitions [195]. Below I am using Sicart's definition of game mechanics — methods invoked by agents for interacting with the game world. That is, the ways in which the player or other in-game agents (e.g., non-playable characters) can interact with

the game world at varying levels of granularity (in a first-person shooter, a fine-grained mechanic could be weapon control, whereas a coarse mechanic could be a cover system).

Game mechanics in this sense have been used for quite some time as a game design tool to evoke meaning through emotion and narrative rather than serving purely as interaction methods. For example, in the original *Missile Command*, the player is given the impossible task of defending several cities from relentless missile attacks from an unknown enemy by taking control of missile defenses with limited resources. Using the simple mechanics to defend cities and bases against an unending wave of violence raises several moral choices, and through it produces a compelling narrative and emotional experience[16]. The player will have to decide whether to prioritize saving populated cities or defense bases, whether they value certain cities more than others based on tactical advantage, and ultimately who to sacrifice and when, eventually losing the game regardless of their efforts.

Perhaps more nuanced, game mechanics themselves can serve as a metaphor for a conceptual or thematic narrative aspect that goes beyond the interactions that the player (or agent) has with the game world. This has come to be known as *mechanics as metaphor*, although alike defining game mechanics, its history is unclear and remains without a standardized definition. The earliest occurrence that I've been able to uncover of a developer explicitly describing their own game's mechanics in terms of a narrative metaphor (but not using the term 'mechanics as metaphor') is in a reflective article from 2007 by Jason Rohrer[17], the designer and developer of the memento mori game *Passage* which depicts "an entire life from young adulthood through old age and death" in all but five minutes of gameplay. Most of the game's mechanics are explicitly designed to be

Fig. 2.7 Three snapshots from *Passage* showing gradual progress through the game.

a metaphor for different aspects of life, death, and the journey in between. Figure 2.7 shows three snapshots from the start, middle, and end of a single game playthrough. The

background within the game is dynamic and alters its behavior and visual presentation throughout the duration of the game based on the player's progression. Rohrer writes that "at the beginning of the game, you can see your entire life out in front of you, albeit in rather hazy form, but you can't see anything that's behind you, because you have no past to speak of", which can be seen in the first snapshot where left of the player is minimal, but the far right is crowded yet difficult to comprehend. Rohrer continues, "as you approach middle age, you can still see quite a bit out in front of you, but you can also see what you've left behind — a kind of store of memories that builds up", which can be seen in the second snapshot where left of the player maintains reference to the past but hazy, and right of the player is clear yet not too far in the future. Rohrer concludes by stating "toward the end of life, there really is no future left, so life is more about the past, and you can see a lifetime of memories behind you", which can be seen in the third snapshot where left of the player is vast yet hazy, as past memories tend to be as age progresses, and the future is minimal both in terms of sight and time. The combined mechanics of the player's horizontal traversal and the responsive background act as a metaphor for ageing, with particular detail on the typical stages of life and memories. Several aspects of the player's movement also act as metaphors. For example, if you choose to join with a spouse at the beginning of the game, they will always perish before you toward the end of the game, at which point the player's avatar hunches over and considerably loses speed, serving as a metaphor for grief at the loss of a significant other.

The earliest direct reference to the concept of mechanics as metaphor I've been able to discover is in an interview conducted in 2007 by Jason Rohrer with Rod Humble in regard to his contemporary game *The Marriage*[18]. In this interview, Rohrer cites Raph Koster's *Theory of Fun for Game Design* [115], explaining that Koster suggests that artistic games can carry a deeper meaning than demonstrated by their gameplay alone, specifically in regard to game mechanics and playstyle which he compares to a viewer's choice of interpretation when watching artistic films. Rohrer adds that in *The Marriage*, Humble demonstrated how "we can also make art by using game mechanics as a metaphor directly", noting that it pulls away from Koster's idea of playstyles and more toward the analogy of interpretation, further directly questioning Humble "why did you chose[sic] the 'mechanics as metaphor' approach?". In response, Humble states that he believes the use of rules and structure (game mechanics, in this context) for communicating meaning are strong tools that designers have available to them. While the concept of mechanics as metaphor was undoubtedly being used prior to its popularity as an actual term, we do see a gradual increase in both its usage and discussion of the topic through the late 2000s[19]. In 2012, *Extra Credits* produced two educational videos on the subject[20], which on *YouTube* alone have amassed over 478,000 views and over 1,600 comments as of 23 March 2021. Undoubtedly, this accessible presentation of mechanics as metaphor has reached an audience that other aforementioned sources were unable to, possibly serving as a motivator for the term to become commonplace. In the videos,

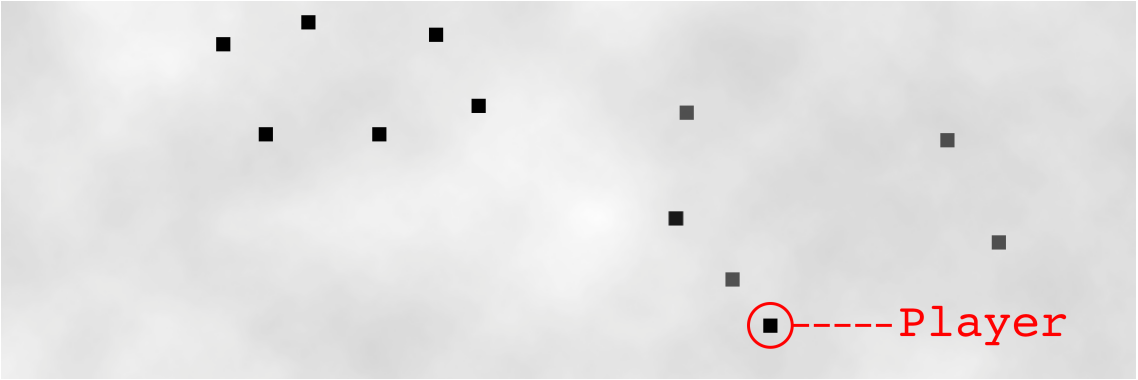


Fig. 2.8 The game *Loneliness* demonstrating mechanics as metaphor through isolation.

they provide several examples of mechanics as metaphor in contemporary games, but prominently focus *Loneliness* by Jordon Magnuson, as shown in Figure 2.8. In this game, the player can move their square in all cardinal directions, progressing forward. When they come into close proximity of another group of squares, they scatter as if avoiding the player's square. The metaphors that the mechanics of this game portray depend upon how the player approaches (or does not approach) the other squares within the environment, as well as their own interpretation of what the responses to their actions mean. For example, if the player had begun by approaching groups, but after repeatedly having them move away decided to instead traverse the rest of the game intentionally avoiding groups, this could be interpreted as a metaphor for rejection when trying to fit in, ultimately leading to self-inflicted isolation.

Of course, we continue to see mechanics as metaphor further refined and used in games today. A great example of this is *Hellblade: Senua's Sacrifice*, a dark action-adventure game inspired by Norse mythology which follows Senua on a journey to Helheim to rescue the soul of her dead lover from the goddess Hela. The unique twist is that the game's main character suffers from psychosis, and both the game's overall narrative and its mechanics are intentionally designed to portray, through metaphor, the various aspects that those who suffer from psychosis can encounter. During one segment, for instance, Senua's progress is hindered by the great beast Garm who tends to the shadows, meaning that the player, in a dimly lit environment with extended areas with little to no light, must carefully navigate the environment to solve the puzzles at hand while taking into account the amount of light they are receiving at any given point. As the player remains in darkened parts of the environment, the music grows louder and more intense, the environment becomes a bloodstained red with strobing lights, Senua's inner voices become more aggressive, and control over Senua appears less precise as she stumbles in fear. The mechanics at play in this part of the game serve not only gameplay, but also as a metaphor for the overstimulated senses, shutting down of oneself, and feelings of intense anxiety associated with particular stages of psychosis.

Quick-Time Events. Quick-time events are short-lived sequences presented to a player in a game in which they must respond to some given prompt requesting player input.

The result is typically binary in that the player either achieves the requested prompt or does not. Based on the outcome, the narrative is able to branch. One of the earliest games to feature quick-time events was *Dragon's Lair*, where amongst interactive movie playback, the player was frequently presented with numerous input actions which they had to respond to within a given timeframe. The success or failure to do so redirected the narrative, either allowing the player to continue the narrative or terminating the game early with a premature death respectively. Another example can be seen in *Shenmue*, where on the fifth day working at the harbor, the player engages in a chase scene following two characters, which is interacted with exclusively through quick-time events. Of that chase sequence, one particular quick-time event allows the player to diverge either left or right, each direction chasing a different character and ultimately impacting both the quick-time events that follow and the overall narrative for that segment before it rejoins. *Shenmue II* presents an interesting case where a quick-time event itself is used as a metaphor for an in-game narrative action, as well as a form of agency for progressing along a specific narrative pathway. The main character, who is present in a barbershop while being lectured about philosophical concepts of trust, has a cut-throat razor placed against them, followed by an extended quick-time event. If the button is pressed, as is the instinct of players, the sequence fails. Only if the player goes against the quick-time request and purposefully ignores the sequence does the game continue. This mechanic is used as a metaphor for trust, tranquility, and remaining calm under pressure.

Triggered Events. As video games are an interactive form, they are able to take advantage of dynamic events based on the state of the world that otherwise may not occur. These can occur when the player moves into a specific space or when other state-dependent conditions are met. The player is usually unaware of how these events are triggered, as they are typically associated with authored narrative segments, such as triggering a cutscene. These triggers serve as launching points upon which further narrative can be delivered. The player therefore has no real agency over these events other than to avoid, if possible, the conditions that raise the event. In action games such as those from the *Tomb Raider* and *Uncharted* series, triggers are commonly used to cause irreversible damage to structures that the player interacts with while traversing, usually being accompanied by a short cutscene or in-game event sequence to show the destruction, and sometimes forcing the player to progress forward by having the damage block the trodden path.

Dialogue Trees. Dialogue trees are, at a high level, sequences of conversational dialogue that a player can traverse through interactively by repeatedly selecting responses from a list until reaching the end or being otherwise ejected from the sequence. These interactive conversations can help the player feel more involved in the story and are often a primary means of narrative delivery in role-playing games, where interaction with other characters is essential. Due to the dynamic nature of games, the dialogue choices available to players can be chosen or altered based on, for example, the player's morality or previous activities.

A limited sense of agency can be instilled by providing temporary branches that quickly reconvene, which allows for narratives to provide variation in response but maintain a reasonable scope. Some dialogue choices, however, may have more of a permanent impact upon the narrative. For example, in *Mass Effect*, the player enters a heated debate with a character named Wrex and is eventually provided several dialogue options to resolve the situation, two being available depending upon the player's morality score. Choosing an unsatisfactory option in this dialogue will result in Wrex being killed, meaning he is no longer a part of the narrative, which has dramatic impacts upon future narrative sequences not only in this game, but in the two games that followed.

Explicit Player Choice. Mason introduced a vocabulary of *diegetic* and *extra-diegetic* choices to describe types of choices within games [141]. Diegetic choices represent those that players can make as a character or presence within the story world that affect the story, whereas extra-diegetic choices are those 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 gameplay for either a limited or indefinite about of time while the choices are being made. The use of diegesis in video games is not a new concept[21] 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*. In one scene, the player confronts a friend-turn-foe trapped under an overturned tanker with an explosion imminent. The trapped character begs for forgiveness and requests the player to put them out of their misery. However, the player is able to, during gameplay, either optionally eliminate the character or alternatively walk away and leave them be. The choice to comply with the character's requests is completely at the will of the player, and the choice to do so is made as an active participant of the game world.

Extra-diegetic choices are commonly used in dialogue-centric games, where the story world essentially halts as the player makes a decision, usually with no ramifications for time spent making a decision. In *Life is Strange*, for example, when major decisions are presented to the player that will significantly alter the course of the narrative, the game world halts, taking the player out of control of their avatar and providing indefinite time to make a choice.

Player Traits. The interactions that players have with their game world can also be altered based on traits associated with their avatar. These traits alter the narrative in a variety of ways, often personalizing the player's own narrative experience by tailoring it to their playstyle, which can increase immersion and believability of the world. For example, in *The Elder Scrolls V: Skyrim*, guards react differently based on the player's race, bounty, and other factors. Players of the Nord race may experience a guard passively announce "How can I help a brother Nord?", whereas players of the Khajiit race may experience discrimination along the lines of "Stay out of trouble, Khajiit". A comparable setup can be found in *Mass Effect*, where the player's morality, which is based on their

actions within the game world, directly impacts which dialogue options are available during conversation. The meaner a player acts, the more mean options become available, and vice versa. A more drastic example of player traits affecting the narrative outcome is in *BioShock*, where the player's morality, which is based upon saving or harvesting Little Sisters throughout the entirety of the game, ultimately determines which of two endings the player receives upon completing the game.

2.4 Authoring Tools

What is an authoring tool in the context of IDN? This question could be answered simply by stating that authoring tools are software programs that help authors in the development of interactive stories. However, authoring tools are much more than that, and remain as a pillar of discussion for the academic community as evidenced by the existence of the *Authoring for Interactive Storytelling Workshop*[22] that runs alongside the *International Conference on Interactive Digital Storytelling*, as well as the wide array of authoring tools that have been and continue to be proposed [86, 189].

This section provides an overview of the evolution of authoring tools in the IDN space, in particular regarding the complexity of authoring, their interfaces, and their accessibility. Select authoring tools are used to demonstrate the trends but are not intended as a taxonomy. This is then followed by an examination of a range of authoring tools to identify common features of the tools, to identify any potential clusters of genres of tools, and to find the spread of varying approaches for supporting authoring of IDN. The section then concludes by looking at the position of select authoring tools in a wider context beyond their authoring process alone to provide insight into how authoring tools are used in a pipeline.

2.4.1 A Brief Evolution of Authoring Tools

The way in which an IDN is created has evolved alongside the growth of the discipline itself. Throughout this, we have seen changes in authoring methodologies, usability and accessibility improvements to allow a wider audience to create rather than only consume, and greater complexity of affordances offered to authors. At a high level, the progression of IDN authoring tools has transitioned through three stages — terminal-based creation, appropriation of generic hypertext tools for IDN, and creation of specific tools for IDN. We are still advancing in this last stage, particularly in the domain of user experience. These stages are temporal by nature but are not necessarily a chronological progression in that they are expected to overlap as new approaches were explored, and that using new approaches does not mean forfeiting existing approaches.

Terminal-Based Authoring

Early IDNs were implemented by technical authors, generally programmers, using rudimentary authoring systems, usually by means of directly programming the story within a terminal. The early IDN games *Colossal Cave Adventure* and *Zork I*, for example, were both written for the PDP-10 in FORTRAN and MDL respectively. Figure 2.9 shows the former being executed on a VT100 terminal. The authoring experience for these systems involved using basic text editors such as *EDT* or *KED* at the terminal. It must be noted that the hardware with which the terminals communicated were often the size of rooms and only affordable or even accessible by universities or research labs. This meant that authoring such stories was limited to those that were able to afford the equipment and had the technical know-how for working with complex languages or were able to partner with somebody who did. This became less of a concern as home computing began to

Fig. 2.9 *Colossal Cave Adventure* running on a *PDP11/34* with a *VT100* terminal, courtesy of Trammell Hudson, available at https://trmm.net/Advent as of 4 February 2022.

be affordable and popular with the introduction of hardware such as the Apple II, BBC Micro, Amstrad CPC, IBM PC, and others. While the introduction of these computers didn't necessarily change the methods of authoring, they did make acquiring the required hardware more accessible. Many games, including the aforementioned *Zork I*, were brought over to these systems along with a newer wave of IDN works.

The barrier to entry was slightly improved with the popularity of printing complete game source code in magazines, also known as 'type-it-in-yourself games', but this

still required knowledge of how to operate the machines and in some cases set up a development environment [159]. Although this did not change the method of authoring or its tools, it did expose people to the contemporary authoring process for the creation of IDNs. Readers were often given instructions alongside the code and were expected to type it in by hand. This mimicked the authoring process bar the responsibilities of a programmer having to initially figure out how to write the story and debugging along the way. This process served as an introduction of sorts to the authoring of IDNs, and since source code was provided over closed binaries, readers could step into authoring by modifying existing stories that they had previously typed out or learn from them and create new stories. While this did improve accessibility, the actual experience for authoring remained challenging due to the inherent technical nature of operating, let alone entering in code on a terminal, even with assistance.

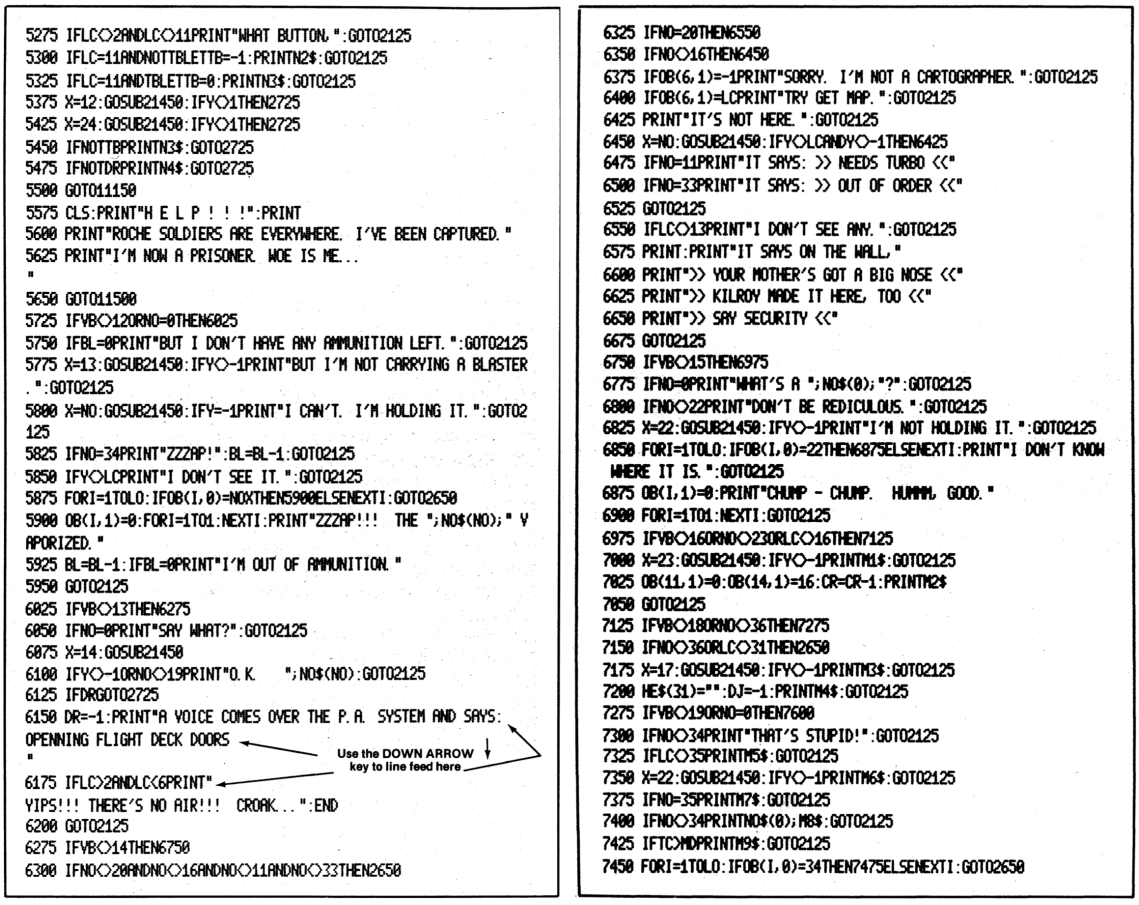


Fig. 2.10 An excerpt of *Dog Star Adventure* in the *SoftSide Magazine* of May 1979.

An early example of this is the publication of Lance Micklus' *Dog Star Adventure*, an excerpt from which is shown in Figure 2.10, listed in the May 1979 issue of *SoftSide Magazine*. This feature printed its complete BASIC source code along with instructions to the reader aside and within the code which they then typed into a terminal. As adventure games became more popular, dedicated material was released to help those who wanted to get into authoring stories. For example, the book *Creating Adventures on Your Commodore 64* [77] released in 1984 targeted users of home computers, introducing them to the concept

of adventure games and guiding them through the development in BASIC starting with simple worlds and moving to multiple complete games. Being produced later than other guides, they were able to leverage subsequent advancements, and in the case of this particular book, guided the reader through implementation and inclusion of graphical sprites, title pages, sound effects, and soundtracks. Unlike magazines, the books were able to expand upon what the code did and served much like a follow-along tutorial.

Appropriation of General Tools

Another category of authoring tools involves the appropriation of other non-narrative, usually hypertextual, authoring environments or technologies for use within IDN production and deployment. Unlike terminal-based authoring methods, these tools presented a much greater variety in authoring experience due to both the rise in computing ability allowing for more affordances and visuals, and also because of the inherent broadness of the category including general tools not particular to IDN.

Apple's *HyperCard* enabled a wave of creatives to bring to life their interactive stories without requiring significant technical knowledge, all despite it being a generalized hypertext authoring tool. The authoring experience of this software was unique, particularly for its time, as it offered a complete graphical editor to manage and create interactive content. The interface made extensive use of skeuomorphism, abstracting the development of interlinked content as 'cards' within a 'stack', much alike a deck of playing cards or a Rolodex filing device. This was accompanied by a built-in extensible scripting language named *HyperTalk* which presented itself as something much closer to the English language than lower-level approaches. For example, we could prompt the reader for their name and display the result in an appropriately named textbox like so:

```
ask "What is your name?"
put it into card field "player_name"
```

This is in contrast to lower-level languages that we looked at earlier such as BASIC or FORTRAN where this kind of task would be exponentially more challenging. Consequently, the authoring experience was much more accessible as *HyperTalk* could be used to implement advanced interactivity without requiring the technical skills for solutions found in terminal environments. Similarly, the use of a relatable, abstracted, and simplified drag-drop authoring interface likewise lowered the knowledge requirements to begin creating. An example of editing within the *HyperCard* environment can be seen in Figure 2.11. The popularity of *HyperCard* resulted in a large number of IDN works being created, some even becoming household titles. For example, John McDaid, a science fiction writer, developed and published his novel *Uncle Buddy's Phantom Funhouse* in *HyperCard* which went on to be a finalist for the *New Media Invision Award* in 1993. McDaid explained that the work's multimedia approach was made possible by the affordances of *HyperCard* and that *HyperTalk* was used to inject randomness into the story[23]. Success with *HyperCard* was also found in the budding video games industry.

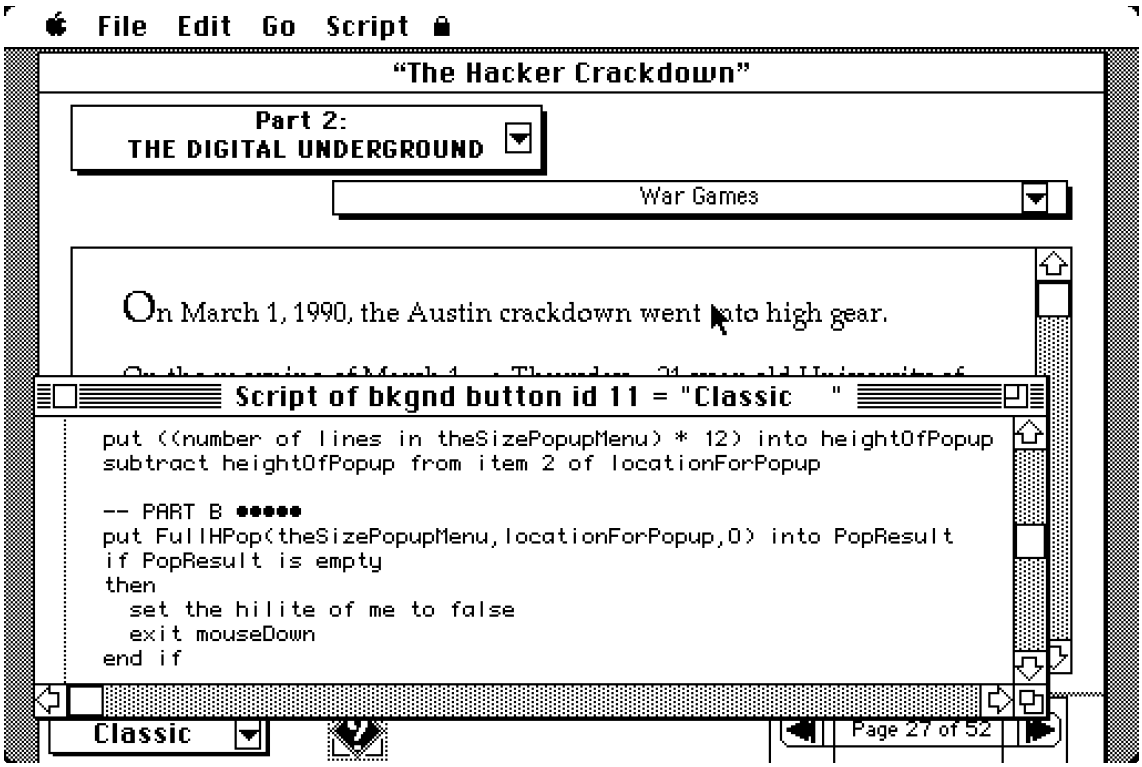


Fig. 2.11 A HyperTalk script in the *HyperCard* stack *"The Hacker Crackdown"*.

The original 1993 version of *Myst* was implemented by creatives rather than programmers and relied heavily upon the *HyperTalk* scripting language. This game was in many ways an evolution of text-based adventure games, taking advantage of the new visualization and interaction paradigms provided by *HyperCard*. As of the year 2000, the game had sold more than 6.3 million units worldwide[24].

This appropriation of general hypertextual software for the creation of IDN can also be found elsewhere. For example, Asymetrix's *ToolBook*, an e-learning content authoring tool, was used to create M. D. Coverley's *Califia* in 2000. This tool was much alike *HyperCard* but for the Windows platform, sharing a similar metaphor of books made up of pages (versus *HyperCard's* stacks of cards), featured a scripting engine named *OpenScript* which was similar to *HyperTalk*, and also featured a drag-drop visual interface for the creation of page content.

Products from the Macromedia (later by Adobe) suite were also commonplace for creation and publication of narratives even though they were general hypermedia tools. *Director*, an authoring platform for the creation of Shockwave applications, allowed publishing to the Web and allowed any computer with a Shockwave Player installed to run the application. This was used to create works such as *Code X* by W. Mark Sutherland, *Vniverse* by Stephanie Strickl and Cynthia Lawson Jaramillo, *Looppool* by Bastian Böttcher, and *Lair of the Marrow Monkey* by Erik Loyer, among many other pieces. *Flash* was another software from the same suite, able to publish interactive vector-based graphical applications into many formats including desktop, mobile, and embedded within websites (providing an appropriate player was installed). Examples of works

created include *pianographique* by Jean-Luc Lamarque *et al.* and *London Eye* by Diane Greco, among a wide range of others.

The Web itself — use of HTML, CSS, and often JavaScript — while not considered an authoring tool in the traditional sense, has also enabled a variety of IDN works. Sometimes these are created with Web authoring tools such as *FrontPage* or *Dreamweaver* like *Fibonacci's Daughter* by M. D. Coverley. Other times, they are authored manually using code editors. Some well-known works have found their home on the Web, such as *Sunshine '69* by Bobby Rabyd (the first Web-based interactive fiction) and *Uncle Roger* by Judy Malloy (which was adapted from a previous command line version for the Web).

Specific Interactive Digital Narrative Tools

Some authoring tools were released that specifically targeted creation of IDN which allowed them to focus their affordances as they were no longer for general purpose authoring. These tools began in the latter period of when terminal-based authoring was popular, yet new tools within this category continue to be released, even today.

An early example of this is Gilsoft's *The Quill*, initially published for the ZX Spectrum in 1983, which presented a terminal-based authoring system for the creation of text-based adventures. The software was also extensible, later allowing for use of simple graphics. Its selling point was that as an author you could "write your own machine code adventures without any knowledge of machine code whatsoever", as shown in an advertisement in Figure 2.12. The advantage of this software was that it provided abstracted functionality for tropes specific to text-adventure games, such as a verb-noun parser, creation and linking of locations, creation and management of objects and inventory, an event manager complete with preconditions and actions, and both automatic variables that track world state (e.g., an integer that decreases each time a command is entered) and manual variables that the user can configure. All of this and more was displayed to the author using simple tables with commands being used for manipulating them accordingly. This software resulted in over 532 IDNs being created, both as commercial products and as freeware[25]. The later *Professional Adventure Writer* also by Gilsoft released in 1987 extended the functionality to add more affordances for authors based on text-adventure tropes. The major upgrades over its predecessor include an improved parser for more advanced interactions, extended character sets, and objects specifically for non-playable characters. The result of this was that authors had to worry even less about technical implementation details and could instead focus on creation. There were at least 487 works created for this software both as commercial products and as freeware[26]. Another notable contemporary of these systems was *The Graphic Adventure Creator* by Incentive Software, which further abstracted command parsing by featuring a more advanced yet high-level system that allowed authors to handle multiple sequential commands rather than have individual commands be issued. There was a vast array of other terminal-based software packages for authoring of IDN, a detailed discussion of which is beyond the scope of this section.

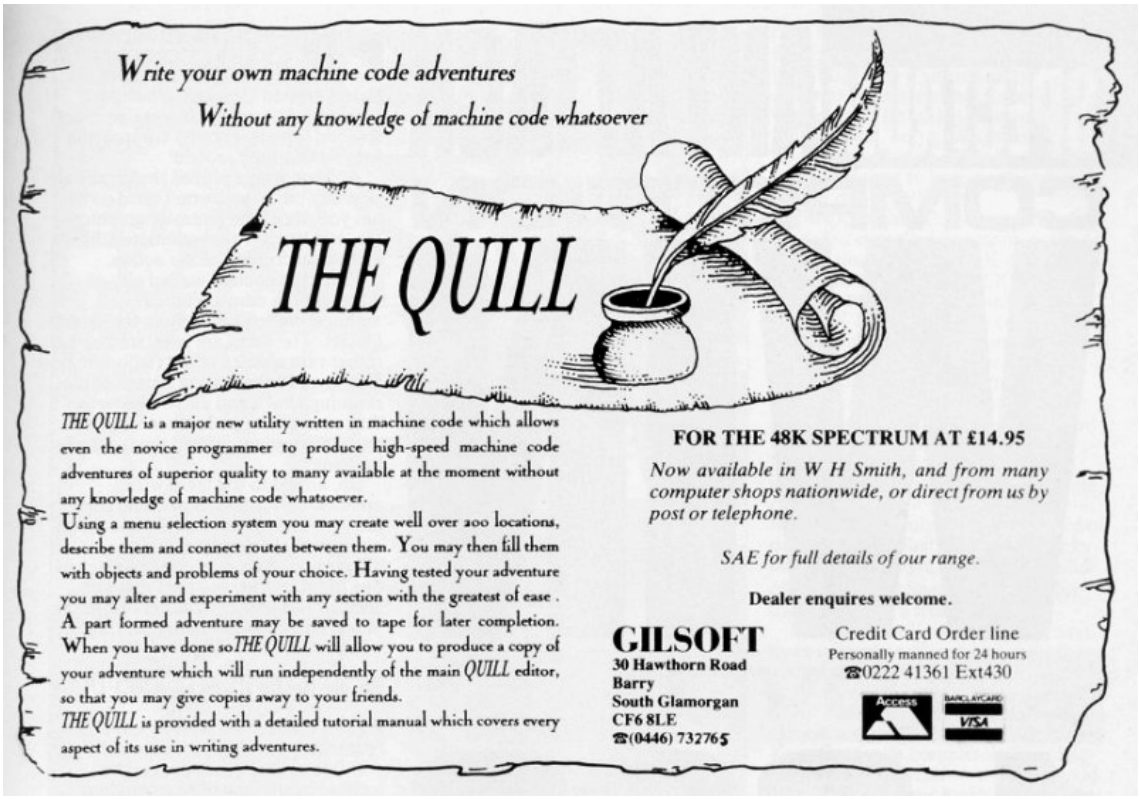


Fig. 2.12 An advert for *The Quill* courtesy of *World of Spectrum*.

Examples include *Adventure Builder* and *Adventurer!* by Sinclair User, *Adventure Builder System* by Tartan Software, and more recently *IF Creator* by n-Discovery to name a few.

The *Text Adventure Development System* (TADS) took a different approach by providing a complete domain-specific language resemblant of C++ and Java for the creation of IDN. While this language did abstract complexities of tropes found in IDN, it was still a technical language and therefore retained a technical barrier at the expense of allowing greater complexity. The software underwent three major revisions eventually resulting in an authoring process of taking source files and compiling them with a command line make tool. Those authoring on a Windows platform were also given the option of using *TADS Workbench*, a fully featured integrated development environment (IDE) including an intelligent source editor, automated compiling, advanced debugging capabilities, and built-in documentation. Despite this technical approach, the *TADS* authoring system was widely successful with 951 titles authored in *TADS 1*, *TADS 2*, or *TADS 3* present on the *Interactive Fiction Database* at the time of writing.

A similar approach was taken by the *Inform* IDN authoring system by Graham Nelson, specifically the seventh revision. In an interview[27], Nelson explains how the first five iterations of *Inform* were short-lived and that the sixth revision became the first truly stable release. This version was, much alike *TADS*, a low-level language, but abstracted many commonalities found in IDN like world building and input parsing. The seventh revision, commonly referred to as just *Inform*, rebuilt the entire language from scratch, instead opting for a natural language approach complemented by an IDE for authoring.

This change came about as Nelson was working on the *Inform Designer's Manual* [159], noting that the descriptive text used to explain the underlying world model was much more eloquent than the corresponding *Inform 6* code (although in actuality, the natural language code of *Inform 7* actually compiles down to *Inform 6* code, but this is transparent to the author). The change to a natural language model accompanied by a visual authoring interface as opposed to command line compiling has been greatly successful. While difficult to ascertain due to varying tags, as of 10 January 2021, searching for the 'Inform' system on the *Interactive Fiction Database* reveals at least 2331 unique works, and 'Inform 7' shows that at least 1156 are written specifically in the seventh revision. Figure 2.13 shows the *Inform* IDE with the natural language code editor on the left, one of the editor's assistive panels for visualizing the story on the right, and the toolbar at the top for easy compilation and packaging of the story.

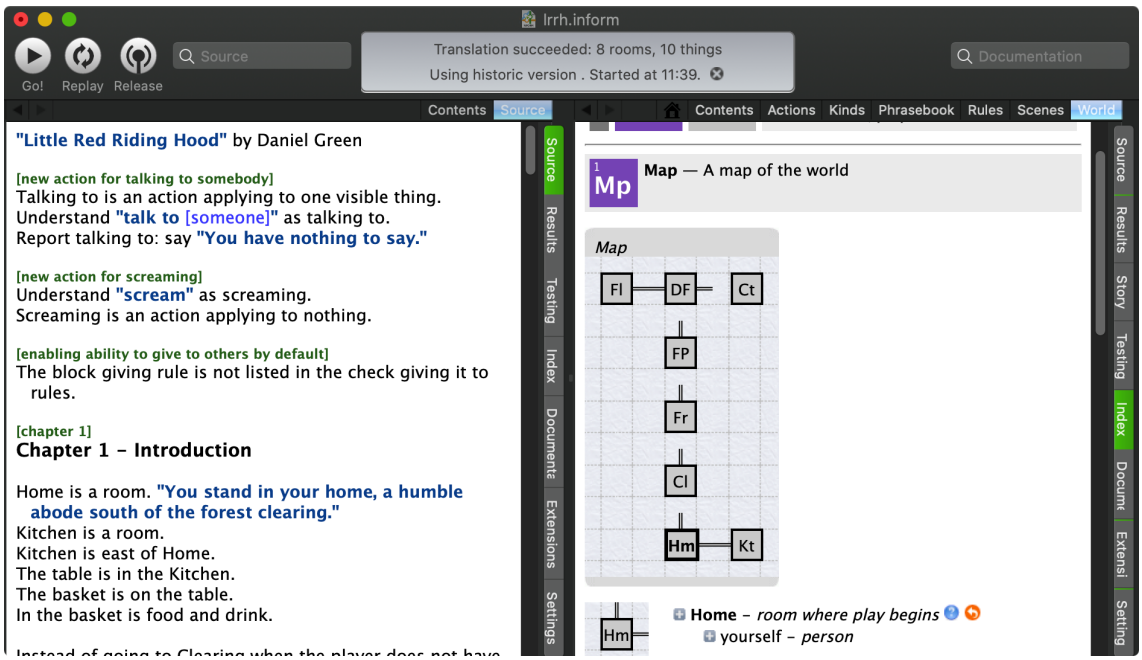


Fig. 2.13 An interactive adaptation of Little Red Riding Hood written in *Inform 7*.

The broader concept of creating a domain-specific language for abstracted authoring of IDN can also be seen in software such as *Hugo* by Kent Tessman and *Ren'Py* by Tom Rothamel. The former, a much older project contemporary to *TADS 2* and *Inform 6*, was a C-like language that provided abstracted syntax for commonalities found in adventure games like locations, objects, and parsing of commands issued by players. This came without an authoring tool, requiring development to be done in an editor and compiled with command line[28]. The latter, while newer, takes a similar approach, offering a syntax based on Python that simplifies many aspects of writing visual novels such as speech, animating and transitioning characters, playing background music, and so on.

Domain-specific languages have also been incorporated as an underlying logical layer within a more graphical authoring interface. This can be seen in *Twine*, where the primary method for visualizing and editing content is through a spatial hypertext graph

on a canvas made up of nodes and connective lines, but the underlying connectivity and interactivity of the content is determined by a contained script within each node. Indeed, *Twine* supports several domain-specific languages that advanced users can switch between, each with different affordances and styles of writing. Figure 2.14 shows a subset of a story in *Twine* presented to the author as a spatial hypertext graph, chiefly used for visualization and organization of story content. Each node determines story content, connectivity, and logical behavior using, in this case, the *Harlowe* domain-specific language. In the figure, the contents of the *wrongname* node demonstrates the combined use of code that is hidden from the reader, text that is displayed to the reader, and special commands reserved for connectivity.

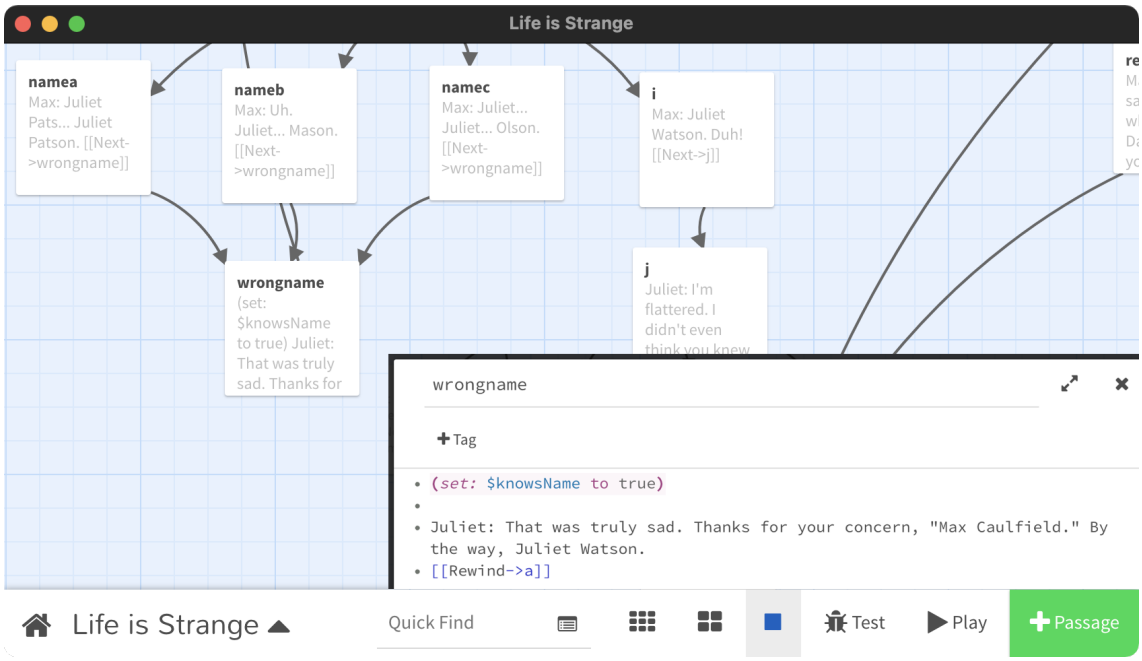


Fig. 2.14 The *Twine* authoring tool showing a combination of a spatial hypertext graph for organization and a domain-specific language for content, connectivity, and story logic.

Certain distinct fields within IDN have specialized needs that cannot be readily accommodated by most general authoring tools and therefore often result in specific authoring tools being created. For example, locative digital narrative is a form of IDN that utilizes additional geospatial or otherwise positional information such as QR codes or GPS data to enhance and contextualize a narrative. Support for location-aware behavior can be found in general hypertext authoring systems, but typically this comes as an additional feature rather than being a focus of the tool. The *ToolBook* software that we looked at earlier, for instance, added geolocation support to its API in 2010, but this is not a typical use case of the software, and as such authoring a primarily locative narrative experience would be challenging. An example of an authoring tool that is built specifically for locative forms of narrative is the authoring tool from the *StoryPlaces* [98, 149] platform as shown in Figure 2.15. The concept of narrative being driven by location is inherent in this authoring tool and its underlying narrative model. As such, the interface and authoring experience of this tool reflect the domain-specific tropes by providing a

geographic map with visual feedback for locations and radii around points of interest. In contrast to a general hypertext authoring tool that may support locative contexts as a secondary feature, authoring tools that are specific in their affordances based on their domain pose a clear advantage to the authoring experience.

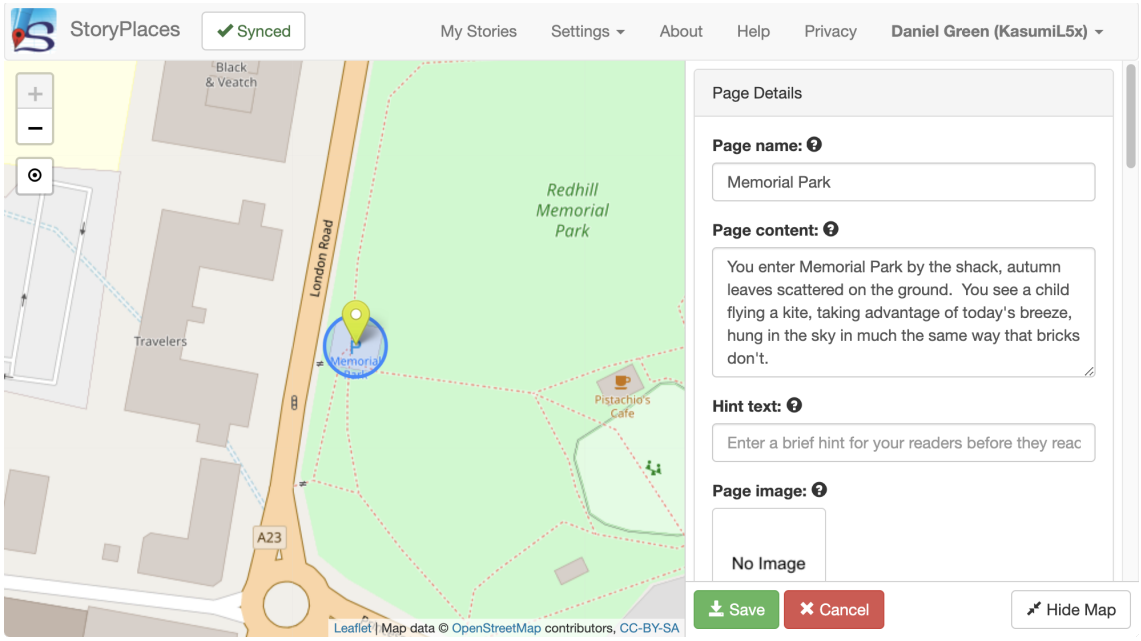


Fig. 2.15 The *StoryPlaces* authoring tool with a map component for locative authoring.

A Note on Game Engines and Game Makers

It is worth briefly exploring 'engines' and 'makers' for games as perfectly valid ways of authoring IDN, addressing their absence in this discussion, and determining the boundary between them and the already discussed IDN tools. In this context, a game engine (*Unity*, *Unreal Engine 4*, *GameMaker*, etc.) refers to a software development package specifically designed to build video games. They are typically made up of a set of components that facilitate abstraction of low-level game development tasks such as rendering, input handling, core logic, physics, networking, world building, and so on. These game engines are accompanied by a visual editor interface for creating the game world and laying out any elements. A game maker (*Bitsy*, *RPG Maker*, *GDevelop*, etc.) is similarly a game engine and editor interface, but it targets a specific subset of genres, adding additional abstraction for the tropes found within those genres, much alike how tools built specifically for IDN abstract its own commonalities.

The boundary between IDN tools and game engines or game makers is blurry at best and is likely to change from person to person. It is difficult, if not impractical, to determine a clear decisive line between the categories, so I instead propose separation based on the generality of the tool and the prominence of IDN abstraction within the tool. Although game engines are perfectly capable of creating IDN, they are designed to be as general as possible to facilitate development of a wide variety of games, and therefore

they do not need to concern themselves with the development or abstraction of IDN concepts and should be labeled as such. In the case where plugins to the engines target authoring of IDN (such as the *Fungus* plugin for *Unity*), the plugins themselves should be considered rather than the engine as a whole. If the game engine incorporates a 'story engine' as proposed by Adams [6] — a dedicated system that communicates with other game systems to track and manage narrative events — then it is plausible to consider the engine an IDN authoring tool. However, story engines are typically bespoke creations for specific games and are not generally integrated into game engines. Similarly, while game makers are perfectly capable of creating IDN also, their abstracted affordances typically focus on the mechanics and systems specific to the subset of genres that they represent rather than IDN itself. In the situation where these systems do support abstracted features of IDN, I propose labeling them based on the relative prominence; if IDN features are surmounted by the rest of the abstractions, then labeling it as an IDN tool would be underrepresenting its intended purpose. Game engines and game makers differ from the appropriation of existing hypertextual tools for creation of IDN in that those tools were never intended for IDN, whereas engines and makers primarily support creation of a narrative medium, but do not necessarily associate themselves with the narrative aspect.

2.4.2 Delivery Methods & Interface Paradigms

The design of IDN authoring tools is an emergent process from the underlying narrative model that the tool supports. However, by building an interface based on a model, it is possible that we overlook the impact on user experience of a variety of design decisions and interface paradigms. Without a well-designed user experience, the accessibility to these authoring tools is reduced, and systems can once more restrict their users to those with the technical know-how or can become intimidating to use due to their apparent complexity. This can result in a frustrating experience for the user and can contribute to a reduced rate of adoption by interactive fiction communities. With the vast range of tools available, some of which were explored above, it is possible that there may be repeated user interface trends in the design of these tools that may have an unforeseen impact on the resulting creative works. Surveying the current state-of-the-art in authoring tools will help to better understand similarities between them.

My survey included 42 IDN authoring tools that cover a variety of disciplines. 17 are sourced from academic research, 10 are developed and sold as commercial products, and the remaining 15 are from other non-commercial and non-academic sources such as open-source or otherwise free projects. This distinction is important as the purpose of these tools can differ; commercial products have a monetary incentive to be accessible and are created with a different goal from academic tools which are typically created for exploratory research, which in turn can alter the design quality and methods of delivery.

The inclusion criteria for tools in this survey was that they directly target IDN, or in the case of general hypertextual tool appropriation, have strong evidence for IDN works

Table 2.1 A collection of authoring tools sorted by their method(s) of delivery.

Standalone	Integrated	Web
Academic		
ASAPS [110], DraMachina [60], EmoEmma [40], FearNot! [116], GAIA [109], HypeDyn [153], INSCAPE [12], PaSSAGE [209], Scenejo [78], StoryTec [79], SVC Editor [222], Villanelle [139], Virtual Human [80], NM2 [213]	GHOST [93], Story World Builder [172]	StoryPlaces [98, 149]
Commercial		
articy:draft 3, Director, Dreamweaver, Flash, FrontPage, HyperCard, Storyspace [21], ToolBook, Celtx Game	None	Arcweave
Other		
Inform, Inky, Quest, Ren'py, Squiffy, TADS, Twine, ADRIFT, Adventure Game Studio, Chronicler	Fungus	Genarrator, Inklewriter, Playfic, Quest, Squiffy, TextureWriter, Twine

being created with them. The authoring tools must also contain some form of graphical user interface that is used in the authoring process, excluding code-only solutions (typically standalone domain-specific languages) such as *ChoiceScript*, *Dedalus*, *Undum*, and *Monogatari*. This is because we are interested in understanding the user interface paradigms of authoring tools, and consequently those without an inherent authoring interface are not included. Similarly, terminal-based authoring tools are disregarded due to their similarity and limited paradigms. These rules together create a selection criterion that considers authoring tools of interest to this thesis — those used for IDN with a graphical user interface — and filters out those that are of lesser interest.

These authoring tools can be broadly categorized by their delivery methods (*standalone*, *web*, *integrated*) and their high-level interface design paradigms (*form*, *graph*, *text*). Table 2.1 shows all of the included authoring tools classified by their methods of delivery. In the case where an authoring tool has more than one delivery method (for example, the tool primarily relies on a web service but offers a standalone offline alternative), then they are included in both categories.

Delivery Methods

Standalone authoring tools offer a dedicated application that is independent of additional software other than the host platform and runtime dependencies. A drawback of such

approaches is the difficulty of cross-platform support, although this is becoming less problematic as the need for industries to have a cross-platform presence is increasingly important and therefore development tools and frameworks are likewise improved to better support this. The *EmoEmma* [40] authoring tool, for example, provided a Windows binary, but does not support other operating systems due to relying upon the platform-dependent WinForms API. Some standalone applications are able to export to interchange formats which increases the space within which the tool can be used. *articy:draft*, for instance, is able to export its contents to an intermediate file compatible with the *Unity* game engine, such as JSON, XML, XLSX, DOCX, and more. This means that while the tool may operate in isolation from other software, it is well-connected through its ability to export to both specific and generic interchange formats.

Web authoring tools provide a browser-based solution that comes with the advantage of being mostly independent of platform (cross-browser support can be challenging and in some cases not possible) and is widely accessible. Authoring tools such as *StoryPlaces* [98, 149] and *Inklewriter* present Web interfaces for authoring, testing, and usually publishing of IDN works. However, with the increase of Web frameworks being deployable on desktop environments (*Electron*, *Cordova*, etc.), some tools are able to deploy as both *web* and *standalone*. An example of this is *Twine*, built on the *Electron* Web framework, which provides a native browser interface alongside a standalone multi-platform desktop application, which increases availability. This is particularly true if the desktop deployment of a Web-based tool is available as an offline service.

Integrated authoring tools situate themselves directly into host software, such as game engines, having a distinct advantage of being tailored to the host and having direct access to the game itself, which other deployment methods cannot easily achieve. For instance, the *Fungus* authoring tool is built directly into *Unity* as a third-party plugin. As it is integrated, it is able to take advantage of *Unity's* native objects directly and deeply intertwine itself with the engine. The plugin is also able to utilize existing authoring features from *Unity* such as the user interface controls, which makes it integrate into the existing development environment more fluently than would an external solution. However, this tightly coupled approach comes with the caveat that the authoring tool relies upon the host which inherently limits its scope.

It is important to consider the way in which accessibility can be affected when choosing a delivery method. Standalone distributions empower the possibility of native performance and can deliver a homogenous experience within the host platform, but do so at the expense of difficulties found within cross-platform native development. Supporting only a single platform can reduce the accessibility of a system. Web-based systems, on the other hand, have increased accessibility due to their platform independence, although to an extent are hindered by cross-browser complications and lesser performance than native standalone solutions. As technologies advance, however, the cross-platform difficulties and performance gaps are shrinking, resulting in more freedom of choice with regards to

authoring tool delivery methods. With that said, while integrated solutions are able to tie themselves closely to the target platform, they still restrict themselves in scope.

Interface Paradigms

Authoring tools can also be broadly grouped by the paradigms that are used in their interfaces. These paradigms are not necessarily mutually exclusive and many tools will make use of one or more within their design.

Form-based interface design refers to the use of atomic user interface controls that are often mapped closely or directly to an underlying data model. For instance, an interface for editing character attributes using a form-based approach would present controls such as text fields and checkboxes that map to the character's attributes in data. These kinds of fields are ideal where an authoring tool requires data input. An example of a form-based interface can be seen in the *StoryPlaces* authoring tool shown in Figure 2.15. While the map on the left is used for abstraction of geolocation input and to provide a visualization of such data, the actual content displayed to the user is edited on the right in a form-like interface present, in this case, with labels, text boxes, buttons, and image selection fields. The benefit of these kinds of fields is that they are readily understandable by most users due to their ubiquity and standardization across all major platforms. However, their danger lies within overuse, as creating an interface of purely atomic controls may not always be the best way to author nuances of IDN, and overuse inherently limits the level of abstraction that an authoring tool can provide, especially in contrast to the other interface paradigms that follow.

Graph-based authoring tools abstract the structure of hypertextual data into visual graphs, the most common being a node-line graph. They come in both static and dynamic variations. Static graphs are for the purposes of visualization only and do not allow content or structural editing to take place, although they may still be interactive. An example of a static graph visualization can be found in *Inklewriter* as shown in Figure 2.16. In this case, the map feature uses annotated boxes to represent the pages of content that the author has created, connected together with lines to show structural relationships. While the author can navigate the graph and click on the boxes to visualize paths, they cannot perform any editing. Dynamic graphs expand upon static maps by allowing for structural editing and often content editing to take place. When present, dynamic graphs are typically the primary method for structure creation within the authoring tool. This abstraction of connected data into a more visual form provides tangible benefits that are not possible with simpler form-based approaches, such as being able to organize content visually according to an author's personal preferences and being able to see an overview of the structure of the story. These graphs come in many varying formats but tend to draw upon techniques from graph theory (e.g., interconnected nodes with lines communicating relationships) and aspects of flowcharts (e.g., sequential or otherwise structures of interconnected nodes using various methods of connectivity).

Consequently, a benefit of using graph-based systems is that they are typically familiar and understandable to users due to the ubiquity elsewhere of other graphs and flowcharts upon which they are built. Moreover, they have been widely adopted within the spatial hypertext paradigm [18, 136, 138, 190] which is utilized in a variety of authoring tools. This paradigm is reminiscent of Conklin's suggestion of nodes and links in 2D space with the use of visual properties (color, size, shape, texture) to aid in understanding a hypertext network and to provide visual cues for orientation within the network [44].

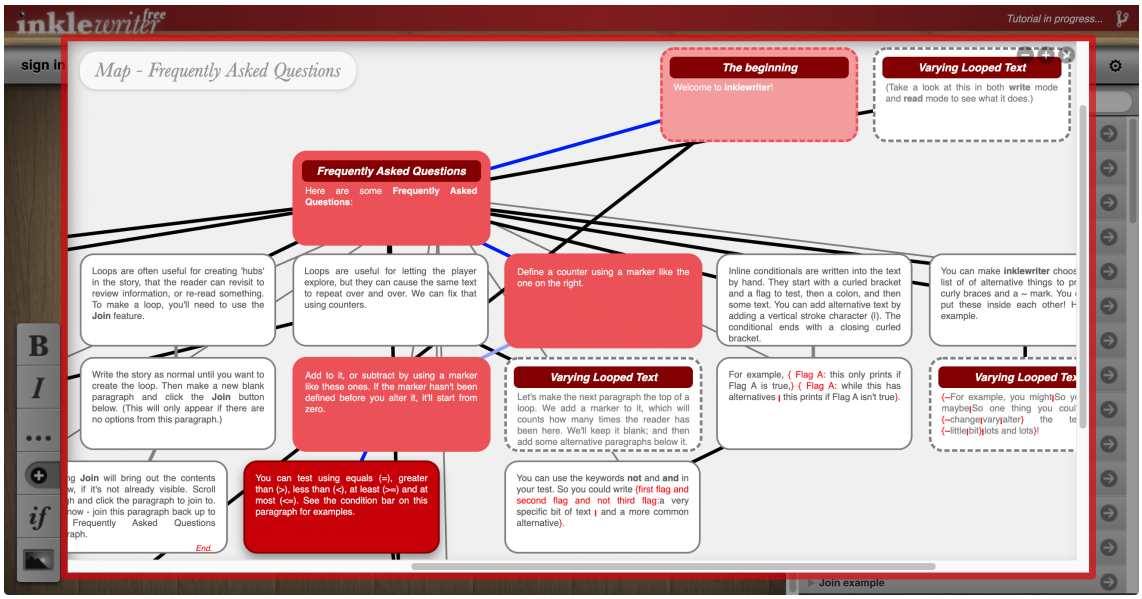


Fig. 2.16 A static graph for story visualization in the tutorial for *Inklewriter*.

Text-based authoring tools feature an augmented text editor, providing additional features such as autocomplete and syntax highlighting of context-sensitive keywords and other markup. This approach is commonly found partnered with domain-specific languages or other forms of scripting. For instance, *Inform*, shown in Figure 2.13, is centered around a plain text editor (visible on the left) with integration of its own domain-specific language and presents, within the text editor, syntax highlighting markup and simple debugging support. While a plain text editor would be technically suitable for writing such code, the presence of an augmented text editor enhances the authoring experience and aids the author in creating correct code through its assistive features.

Within this analysis, the concept of a primary paradigm and shown paradigms is made. A primary paradigm for an authoring tool is derived from the way in which the narrative content is presented to the author for visualization and the way in which authors create and edit their story content and its structural connectivity through this presentation. In the case where two competing paradigms are at play, the paradigm with the most prominence in the authoring tool is used. Shown paradigms are the presence of paradigms that may or may not be core to the authoring experience but are demonstrated within the authoring tool interface. For example, *Inform* is primarily a text-based authoring tool as it relies upon an augmented text editor for all of the content creation and editing, but

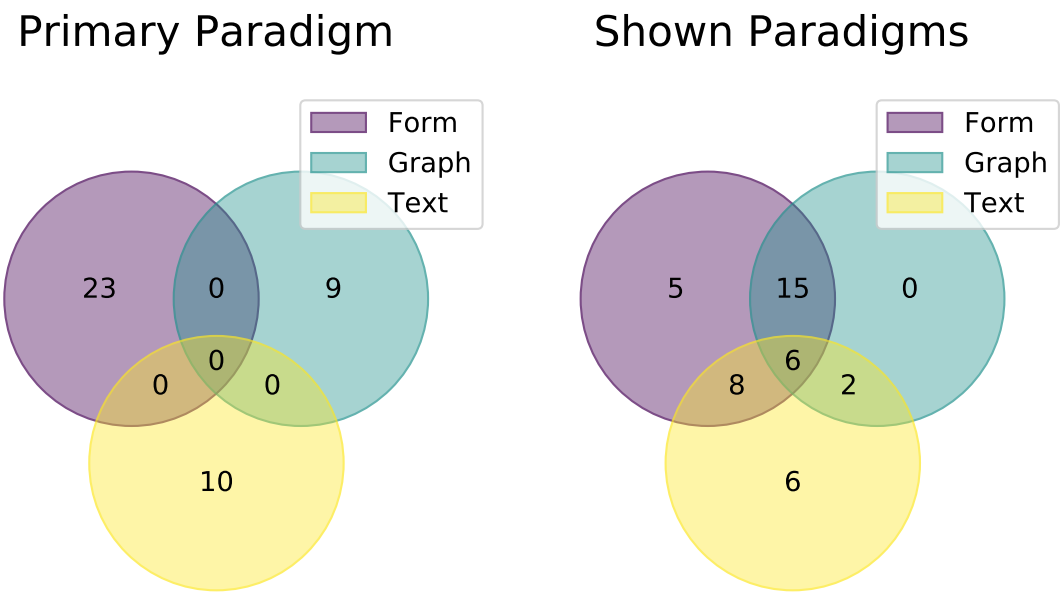
it also demonstrates a static graph for visualization of location connectivity. Therefore, its primary paradigm is *text*, but it shows both *text* and *graph*.

Analysis & Discussion

Figure 2.17a shows a Venn diagram for the primary interface paradigm across all included tools. 23 of the included authoring tools use a form-based interface paradigm as their primary way of editing with a distribution of 14 academic, 5 other, and 4 commercial. This initially suggests that form-based interface paradigms are the preferred design choice across authoring tools, but we must consider the origin and possible motivation behind the constituents of this group. As the atomic elements of form-based designs are ubiquitous across most computing devices, there is an overwhelming amount of support for creating applications with form-based designs including standardized APIs and drag-drop interfaces like *WinForms* or *Qt*. Consequently, creating a form-based design is both quicker and simpler than writing a graph system or an augmented text editor. The included academic authoring tools overwhelmingly favor form-based design with only one being text-based and two being graph-based. The motivation behind authoring tools that stem from academic projects is typically to answer a specific research question rather than to facilitate themselves with the nuances of a carefully crafted user experience that would be vital to a commercial product and do so often without a dedicated software development team. It therefore makes sense that academic authoring tools would favor form-based designs to rapidly serve their purpose under their given limitations. It is likely that the number of form-based authoring tools is somewhat artificially inflated by this. With this in mind, all three interface paradigms are of a similar magnitude with a slight tendency toward form-based systems, but less so than at face value.

Figure 2.17b shows a Venn diagram for all shown interface paradigms across all included tools. That is, for all authoring tools, the presence of all three interface paradigms are considered individually rather than only the primary paradigm. Interestingly, there are no authoring tools that rely exclusively upon graphs, which suggests that, while a powerful paradigm, it is unable to stand on its own as an authoring method. One reason for this could be that graphs do not lend themselves well to data entry, and IDNs are typically driven by content as well as structure. This is reflected in that graphs are most often combined with form-based interfaces. Ten of the 15 authoring tools in this combined category used form-based interfaces as their primary paradigm with static graphs being used for visualization. The remaining five utilize dynamic graphs for content creation and structural organization but delegate actual content editing to form-based entry fields. This combination of paradigms allows for rich abstraction of structural relations between pieces of content, but also presents familiar and more direct editing of content in places where abstraction may not be necessary, such as entering character names or dialogue.

Authoring tools that demonstrate all three paradigms in tandem are uncommon, likely due to the effort required to create such solutions, or the practicality of including all



(a) Primary interface paradigms.

(b) Shown interface paradigms.

Fig. 2.17 Primary (a) and shown (b) interface paradigms for all delivery methods.

paradigms based on the underlying data model and intended user experience. In four of the six cases where all three paradigms are exercised, graphs are the primary paradigm, with form-based fields being used for content entry, and augmented text editors being used for entering scripts. The remaining two cases primarily use form-based paradigms, with one featuring a static graph for visualization and the other heavily relying on graphs almost to equal weight of the form-based editing functionality.

It is also common for form and text paradigms to be combined, in this case totaling eight instances. Six of the eight tools in this denomination prioritize form-based methods accompanied by drag-drop interfaces with augmented text editors for script entry.

Each paradigm presents its own advantages and disadvantages to the authoring process. Using a given paradigm does not necessarily mean that the UX will predictably be a certain way as this is largely dependent upon the details of the implementation. We can, however, conclude general observations of trends regarding accessibility when these paradigms are used. Based on this survey, *Text* interfaces provide the most power to the user as they are typically associated with a narrative grammar but do so at the expense of usability. Authors must first learn the grammar and often have little to no assistance beyond syntax highlighting and documentation, which can result in a tedious authoring process, particularly to new or non-technical authors. On the other hand, *Graph* interfaces are more accessible due to their familiar flowchart-like design used to abstract and visualize complex intertwining relationships of the underlying narrative data. The abstractions that this method can achieve cannot be easily matched with other paradigms but do so at the loss of control provided by text-based systems. *Graph* interfaces are also typically unable to stand alone and are combined with another paradigm, typically *Form*, to supplement the drawbacks such as actual data entry for the narrative content.

Form interfaces are reliant upon the visual presentation of the involved atomic controls, benefiting from familiarity due to their ubiquity in almost every computing platform. If done well, they can provide a basic abstraction of the underlying data model that is usually closely mapped to the atomic user interface controls, making them ideal for manipulation of the basic data of the underlying data model. Caution must be taken with this approach to ensure that the data is presented meaningfully as to not overload the author with too much information at once, risking the authoring tool becoming a data entry tool rather than an IDN authoring system. This is often found in applications that have not considered the user experience and instead choose to simply expose the data model directly through controls rather than abstracting the authoring processes.

2.4.3 Authoring Tools in Context

For creators of IDN, authoring tools are an environment for the creation of various materials that will be composited into an interactive experience. However, the actual receivers of IDN experiences typically do not interact with the authoring tool itself, instead using some form of playback method that is the result of the authoring tool's output. If we focus exclusively upon the authoring tools themselves, then we risk studying them in isolation, disregarding the greater context within which they exist. While a complete investigation is beyond the scope of this chapter, it is worth briefly exploring the ways in which works created in IDN authoring tools become a finished product that a user can experience. Below is a listing, based on the tools included in the above survey, of various high-level paradigms found between an authoring tool and the realization of its output into something users can interact with.

Custom Playback Engine

Fig. 2.18 High-level process for exporting to a custom playback engine.

Figure 2.18 demonstrates a common paradigm where an authoring tool is accompanied by a specialized dedicated playback software. Typically in this setup, the authoring tool outputs an intermediate file that can be opened for playback in the standalone playback software and is usually free for users even if the authoring platform is not. A benefit of this approach is that if the target platform that users will experience the IDN through is known and guaranteed, then the affordances of the authoring tool can be carefully crafted to mirror the known abilities found within the playback software. However, the tradeoff is that the authoring tool inherently locks its feature set to that of the playback software's capabilities, and the playback is limited to platforms that the custom software supports, which may or may not be a problem depending on the goal of the software.

Where alternative methods of exporting are allowed, the software then risks disparity between versions as to which features are available based on the capabilities of the targets. An example of this is the *ADRIFT* authoring tool that exports its stories to a custom format that can be loaded and interpreted by the standalone *ADRIFT Runner* software or played using a plugin in a browser. A similar approach can also be seen in *HyperCard*, where exported stacks were playable in the free *HyperCard Player*.

Wrapped Custom Playback Engine

Fig. 2.19 High-level process for exporting to a wrapped custom playback engine.

A variation of custom playback engines, shown in Figure 2.19, is to wrap a custom playback engine in a platform-independent package. The *Ren'Py* authoring tool is an example of this, which uses a build process to create a custom binary for each target platform that wraps the playback engine accordingly. This reduces platform dependence and consequently increases feature parity between targets. However, the approach is technically demanding to implement and is therefore less common.

Direct Integration

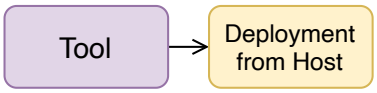


Fig. 2.20 High-level process for direct integration from within a tool.

Authoring tools that are integrated into a host platform, such as a plugin for a game engine, have a distinct advantage of becoming playable as they are being authored, not requiring explicit exporting or converting into an external system. Figure 2.20 shows this procedure, where the only step between the authored content in the tool and the deployment is the host runtime's own build system (such as packaging a game in *Unity*). This has a clear advantage of iteration speed, and these integrated tools can leverage the host's capabilities such as debugging features which can further speed up development and deployment. An example of this is the *Fungus* plugin for *Unity* which inherently builds a playable story within *Unity* as it is being authored and benefits from many of the capabilities of its host like being able to directly reference *Unity* objects.

Export to Web Technologies

The ubiquity of web browsers and the interactivity that they provide make them an ideal target for deployment of IDN experiences. Some authoring tools choose to leverage this

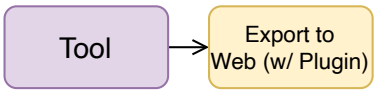


Fig. 2.21 High-level process for Web deployment (optionally with a plugin).

by enabling export to Web technologies like HTML, CSS, and JS, the process of which is shown in Figure 2.21. This is particularly common in authoring tools that themselves are based on Web technologies. Examples of this include *Squiffy* and *Twine*, which both are based on Web technologies and primarily export to Web formats. The advantage of this format is that users can readily open and experience IDNs without any additional technology beyond their web browser. The downside is that the IDN is limited to the technologies supported by the Web, although this is becoming increasingly complex over time. A more restrictive variation of this is exporting to the Web but requiring the user to have installed a proprietary plugin, such as the *Shockwave* plugin required for content created in *Director*, or the *Flash Player* for content created in *Flash*. While this does have the benefit of allowing techniques not possible with vanilla browser-based deployments, it requires an extra step for the users to be able to experience the IDN.

Publish to Web Directly from Tool

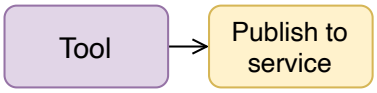


Fig. 2.22 High-level process for publishing to a Web service.

Another Web-based variation, shown in Figure 2.22, allows for publishing and hosting of the IDN creations as a service from within the tool itself. This differs in that there is no export process required and that the steps for viewing the published material are streamlined for the user as they simply have to visit the service online. Examples of this include *Genarrator*, *StoryPlaces*, and *inklewriter*, which all feature the ability to publish IDN works from the authoring tool interfaces to a hosted Web service where others can experience them by simply navigating to an appropriate address in their web browser.

Export to Intermediate File

Fig. 2.23 High-level process for exporting to an intermediate file.

A common method supported by a large number of authoring tools is to enable exporting of the IDN created within the authoring tool to an intermediate common format such as JSON or XML, the process of which can be seen in Figure 2.23. The nature of intermediate

files means that the steps following export will vary. Sometimes the exporting of an intermediate file is the endpoint, where the authoring tool then defers the playback of the exported IDN to others. An example of this is *Inform*, which exports primarily to intermediate formats that are able to be played back by common IDN interpreters. Alternatively, some intermediate exports are intended for use with an API for integration into other systems. In these cases, the authoring tool is separated from the integration of the IDN into the runtime system, which places a burden upon the authoring tool to enable representative playback within itself to aid testing during development. An example of this is *articy:draft* which exports to an intermediate file that can be read by their plugin for *Unity* or be used by their general API by programmers to integrate into a custom runtime. Similarly, the *Inky* authoring tool exports to JSON and provides an API for programmers to implement exported IDNs into *Unity*.

2.5 User Experience for Authoring Tools

We can trace practices reminiscent of modern UX to the *Human Factors Engineering Department* of Bell Labs in 1947, founded by renowned psychologist John Karlin, which according to their own recounting was formalized around user-oriented design and marks the rise of dedicated behavioral analysis with systems [97]. The actual phrase 'User Experience' was coined by Don Norman at *Apple* in their *User Experience Architect Office* [169, 170]. Norman explains that this term was introduced as he believed that 'usability' alone was not representative of the interactions that humans have with computers. This new phrase intended for UX to be present throughout development and to cover *"all aspects of a person's experience with a system, including industrial design, graphics, the interface, the physical interaction, and the manual"*[29]. Around the same time, Lauralee Alben was similarly working to define what qualities make up an 'experience', focusing on a user's sensations, understandings of how things work, feelings during usage, and the achievement of their goals with the system, although this was chiefly on experiences within interaction design, whereas Norman's work covered a greater spectrum [7].

Despite these early works and more recent admission into ISO standards (*ISO 9241–210:2010*), there is active discussion of what makes up UX [14, 99, 122, 124, 125, 152], and more recently how UX is positioned relative to emergent neighboring fields such as customer experience and brand experience [73]. This lack of an agreed upon definition can be explained by the fact that UX is an umbrella term associated with a variety of fuzzy and evolving concepts [125]. An important step towards identifying a common understanding of UX was a roundtable of 30 experts from both academia and industry working together to bring clarity to the concept of UX [181]. Researchers and practitioners agreed that UX is a subset of the general concept of experience, that it includes encounters with systems (both actively using and passively observing), that it is individual and influenced by prior experiences and expectations, and that it is rooted in both social and cultural context.

Moreover, they highlight that UX should *not* be driven by technology, instead focusing on humans first. It is also noted that while users may perceive UX as usability, it is but a subset of the larger picture, and objective usability measures (time to complete tasks, number of clicks and errors, etc.) alone are poor representations of UX quality as they do not capture the perceived experience from the user's point of view.

A model that demonstrates this particularly well is Elizabeth Sanders' work on product development research, a process she refers to as *converging perspectives* [187]. This model builds upon Abraham Maslow's *Hierarchy of Needs* [140], a seminal work in psychology that outlines various human needs that must be met before others are fulfilled, such as ensuring physiological needs (food, shelter, etc.) prior to approaching self-actualization needs (partner acquisition, pursuing goals, etc.). This concept also applies to a user's experience. Sanders explains that basic needs and expectations of people using a product or service must be met before considering more complex functionality. Figure 2.24 demonstrates both of these models depicted as a hierarchical pyramid. Here, 'useful' refers to the need that the product is fulfilling, 'usable' refers to the usability and understandability of the product, and 'desirable' refers to why and how much people want to use the product. In the case of authoring tools, they should be first useful before they can be usable, and should be usable before being desirable. Not satisfying this order can result in a poor overall user experience. For example, if a tool is attractive in its design (*desirable*) but ultimately fails to satisfy the user's needs (*useful*) or is difficult to use (*usable*), then the quality of the user's experience with the tool will suffer. Whether or not a tool is considered useful is a matter of ensuring that there is a problem that needs to be solved, knowing who the users are, and identifying requirements for the tool to successfully solve the problem. On this note, Sanders exclaims "If the product is not useful, if consumers don't need it, then why bother to make it usable? Why bother to make it at all?", which highlights the importance of ensuring that there is a problem that needs solving. In the context of authoring interactive stories, the problem is the complexities involved with authoring such stories, and the authoring tools are trying to reduce this challenge. Determining if a tool is usable is about examining and refining the usability of the software. This can be done in a variety of ways at different stages of development, as discussed below, such as by utilizing participatory methodologies [156] during ideation,

Fig. 2.24 Maslow's original *Hierarchy of Needs* (left) and Sanders' *converging perspectives* (right) which is based on the concept of the former applied to user experience.

following established user experience design paradigms during development (established laws, good practices, considering accessibility), and analyzing usage after deployment (heuristics analysis). The desirable attributes of an authoring tool refer to aesthetics and appeal. Functionally, this can be considered the least important part of a user's experience as long as the core behavior is implemented despite its visual appearance. However, a phenomenon known as the *aesthetic-usability effect* [119] states that users are strongly influenced by the aesthetics of a user interface, even when trying to evaluate the underlying functionality. This effect means that attractive designs are often perceived as more usable as people tend to believe that if things look better, they function better, even if they are functionally identical or even inferior to other less visually appealing designs. Moreover, the positive emotional response that is generated by aesthetically pleasing designs makes users more tolerant of minor usability issues. These sentiments are echoed throughout Aaron Walter's *Designing for Emotion* [215], in which he chiefly discusses how designing beyond the realm of functionality into that which users resonate with aesthetically can enhance the overall user experience even if the software doesn't functionally change. In the context of authoring tool design, this means that we should strive to not only make the user's experience functional but also pleasing so that their overall satisfaction and confidence with our authoring tools increases.

2.5.1 The Importance of UX in Authoring Tools

The importance of refined UX for interactive narrative authoring tools cannot be understated. Poor usability of software can increase cognitive friction [45], resulting in a frustrating experience. It has also been shown that designing systems without considering how users process information and solve problems using the system can alienate the users [75]. In the case of authoring tools, this can also create technical barriers which act as gatekeepers preventing creatives from fully utilizing the medium [149, 201]. Additionally, if an authoring tool is inefficient or unpleasant to use, then people will look to replace it with something more convenient when possible. This means that authoring tools with poor usability may suffer from a reduced rate of adoption by interactive fiction communities, although it is worth noting that the popularity of software is far more involved than just its usability alone and is beyond the scope of this thesis.

The user experience surrounding the authoring of interactive stories has been a point of interest and discussion in the academic space for quite some time [199, 200]. A discussion by Spierling and Szilas [201] raises several important user experience issues within authoring tools. It was found that unclear design, both in terms of visual clarity and interaction design, caused confusion for authors and ultimately resulted in a slowdown in the authoring process, which despite previous works tackling the issue remains troubling. Similarly, a lack of functionality and not meeting author needs caused further slowdowns, such as slow iteration speeds between writing and previewing the content, and not supporting sufficient visualizations of the content's structure resulting

in authors creating maps outside of the authoring tools respectively. Authors were also found to not follow expected methods of authoring, often projecting their own methods onto the tools rather than using the methods required by the tools. It was claimed that abstractions of underlying narrative systems were often in conflict with typical methods of creative writing. While this may be true, I find it more plausible that these systems, built during a key time of user experience reflection within the community, suffered from unrefined user experiences rather than the authors being at fault. Evidence of this can be seen in the transition from systems that did not have authoring tools to those that began to include them, and the reduced error rate that came along with this process. For example, it is reported that creative writers working with *IDtension* [206, 207] who had the manual authoring process explained to them, involving the use of spreadsheets, produced resulting documents that were nonfunctional and out of scope with the narrative engine, or when functional used a bare minimum of features available to them. The creator of *IDtension* has also expressed difficulty in the same authoring process. It is later reported that the authoring tool provided for *Scenejo* [202], while not without problems, forced authors to write in the way that the narrative system expects, and did not result in invalid formats, and encounters with authors using the system in the 'wrong' way, while still occurring, were not as extreme. Therefore, with further refinement to the user experience and general clarity of expectations within the authoring tools, it is likely that this error rate would be further reduced.

The quality of the user experience within authoring tools also has implications where they are used in business environments such as by game development studios. In his 2010 Game Developers Conference talk[30], Jim Brown calculated that the increases in efficiency brought about by user experience improvements, when summed for all users of the tools within a company, could result in significant savings of both time and money. As a simplified example, if a tool's usability is improved to save 10 minutes per user per workday and we have 20 concurrent users, then we save 3hr 20m per day. Assuming the typical number of workdays in a year (≈256), then we save just over 35 days and 12 hours. The benefit for a company is that the users have a significant amount of time freed up to instead spend on other things that would otherwise have been taken up by an inefficient or otherwise unfriendly process. This ultimately reduces development time or at least allows for more efficient allocation of work time. The inverse holds true in that if an authoring tool, or even a process within it, is tedious or otherwise hinders authors, then this can result in an increase of frustration and overall time spent with the tool.

2.5.2 Usability Techniques for Authoring Tools

The tools and techniques available to us as designers and developers of authoring tools are vast, and a complete review is beyond the scope of this thesis. However, it is worth understanding the different approaches that can enable us to better the usability and overall experience that users have with our authoring tools.

Following Established Principles

One of the methods by which we can improve the user experience of our authoring tools is to, when designing the interfaces and interactions, consider and follow established principles of good design, with these practices often being rooted in theories from psychology or other behavioral sciences. The *Gestalt Principles* — a body of work about the psychology of perception — are a great example of this. This work was initiated by Wertheimer [217] and was further developed by Köhler [113], Koffka [112], and Metzger [147], collectively referred to as Gestalt Psychologists. This work has had a great influence upon the understanding of user experience, particularly regarding interface design, due to its description of how users subconsciously organize and make sense of pieces of information in respect to a greater context. A subset of this work referred to as *prägnanz*, or alternatively the *principles of grouping*, outlined several principles that explain the mind's inherent preference to perceive patterns and from them identify groups, which can be used to determine the likely visual interpretation of a design that a user may have. The graph-based authoring tools that we saw in §2.4.2, more accurately referred to as *spatial hypertext graphs*, make extensive use of these laws in their design to ensure that authors perceive collections of information as a whole rather than as disconnected or unrelated, which if not achieved could result in an unsatisfactory authoring experience. Spatial hypertext systems achieve this by presenting interconnected hypertextual content as a visual network of links and nodes, typically referred to as a map, making use of both the spatial relationships between nodes and the properties of the nodes themselves [18, 136, 138, 190]. For example, Figure 2.25 shows a snapshot of a story made in *Articy* which utilizes the principles of grouping. The shape of the nodes is used to differentiate base function (from left to right, a filled thin rectangle is a *Hub*, large squares with images and text are *Dialogues*, and the semi-transparent box with a contained node is a *Jump*), and the author has applied a fixed color scheme to further differentiate between collections of related nodes in an idiosyncratic way (all dialogues colored yellow are for one character, and those colored red are for another). This demonstrates an implementation of the *similarity principle* which describes the tendency to perceive objects as related to one another when they share visual attributes that are closely related. Moreover, if this principle was not used in the design, such as by making all nodes appear the same despite function or contents, it would be a lot more challenging to visually read the project, especially as scale increased. *Articy* also makes extensive use of the *common region principle* which declares how items within a shared boundary are perceived as a group that are in some way related. For example, each dialogue node is actually composed of at least six components — the title, an icon indicating the node type, an image of the speaker, preview text used in playback, stage directions used in playback, and the actual text spoken. Within the interface, these items are collectively identified as a single entity, as they are all placed within a distinct shaded region that is not geometrically connected

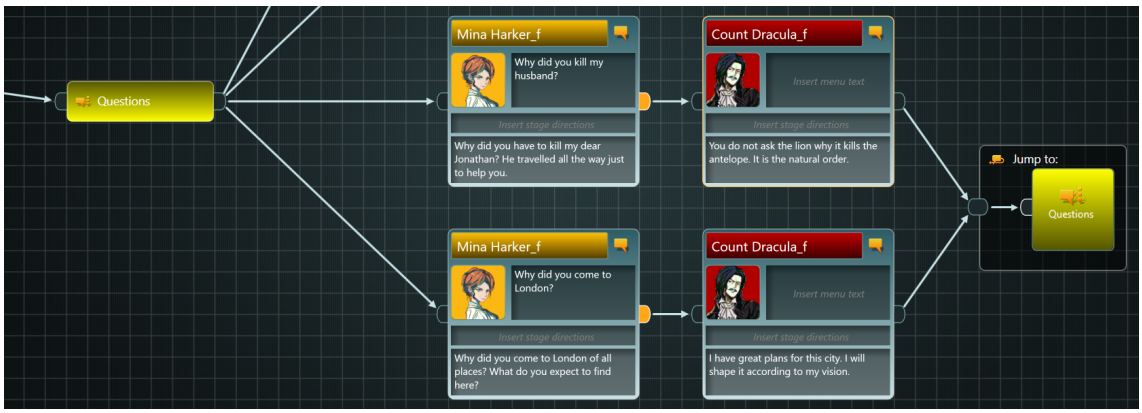


Fig. 2.25 A subset of the sample "Dracula" story for *Articy* showing use of several Gestalt Principles within its spatial hypertext graph design.

to anything else within the interface. This segmentation and containment results in an interpretation of the technically distinct elements as a related whole.

In addition to the *Gestalt Principles* and the *aesthetic-usability effect* described above, the field of applied psychology has discovered a wide range of links between visual design and behavioral response, many of which have become axioms within contemporary user experience design. Several prominent examples are demonstrated below.

Doherty Threshold. In their 1982 journal article *The Economic Value of Rapid Response Time*[31] [59], Doherty and Thadani experimentally demonstrated that there is a threshold for the time an operation takes to provide complete feedback, determining that interactions that surpass 400ms result in a loss of attention and can hinder productivity. For authoring tool design, this means that any interactions with the user interface, especially those that involve animations, should fall within the threshold to ensure that the author's focus is not diverted elsewhere or their concentration is not disturbed.

Fitts' Law. In his 1954 study [69], Fitts demonstrated that the time required to move to a target is a function of the distance and is inversely related to the target's size. Simplified from his mathematical model, the takeaway is that fast movements and small targets result in greater rates of error when attempting to interact with something. This is widely applied in contemporary user experience design by ensuring that interactable user interface elements such as buttons are appropriately positioned and scaled to minimize the chance of error, which could become frustrating to users if not considered. In the context of authoring tool design, this means that the core interactions that authors take that involve user interface element interaction should be suitably scaled and positioned to ensure successful activation. For example, Figure 2.26 shows a node in *Twine* with its interactions toolbar immediately below, activated by hovering over the node. The toolbar itself is close by to the node and each of the buttons within the toolbar has a large activation region. This combination of nearby proximity from the user's cursor and a large activation region means that the rate of error is likely to be minimal.

Hick-Hyman Law. In their 1952 study [101, 102], Hick and Hyman determined a relationship between the number of stimuli a user is presented with and their correspond-

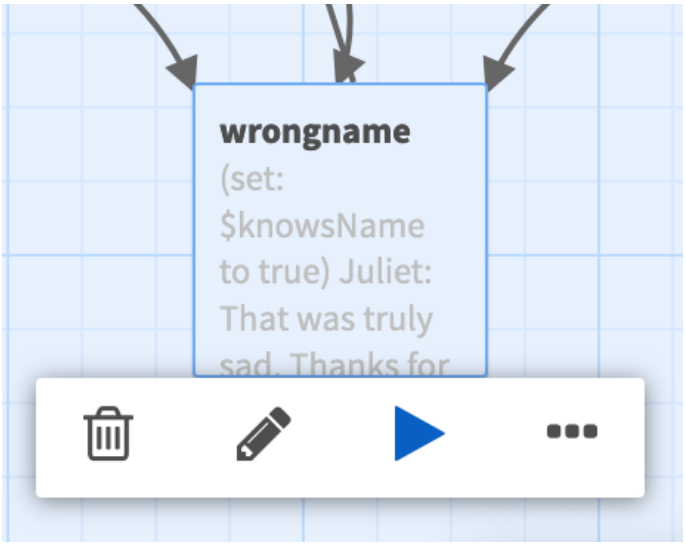


Fig. 2.26 A node in *Twine* with its interaction toolbar below, demonstrating use of *Fitts' Law* by making the toolbar close by and button activation regions large.

ing reaction time to a given stimulus. Naturally, they discovered that the greater the number of stimuli, the longer it takes a user to decide which to interact with. In terms of user experience design, this concept is used by intentionally simplifying complex tasks into smaller steps, reducing the number of overall tasks presented at one time, or by highlighting specific tasks to provide layered stimuli rather than all being of equal visual importance. When it comes to authoring tool design, we must consider the available interactions for an author to complete at any given moment and ensure that they do not overwhelm the author with choice, or if fundamentally unavoidable, implement some form of differentiation between types of interaction to subdivide the stimuli. By not considering this, we risk increasing the perceived complexity of the authoring tool which could negatively impact the author's experience.

Jakob's Law. In his 2000 article *End of Web Design*[32], Jakob Nielsen described the tendency for users to develop expectations of designs and interactions based on their cumulative experiences with other websites. This concept has since been applied to broader user experience beyond just website design, encouraging designers to follow existing design conventions where possible in an effort to reduce frustration and confusion of users due to the user interface and interactions not matching up with their mental model of how things *should* work based on their prior experiences. As many interactive narrative authoring tools are bespoke due to their unique underlying models, they may require the development of unconventional interfaces. However, this law could be applied in all other areas, ensuring that commonalities of user interface and interaction design are followed where possible, and considering whether the bespoke interfaces are truly required or if they may cause more trouble than assistance in the authoring process. Moreover, by understanding the mental models that users have built up with existing authoring tools (both for interactive narrative and those outside of the field), we can create more informed designs that cater to how users expect interfaces and interactions

to be, meaning that they can focus on their tasks and the content rather than having to create new mental models of our designs and their affordances.

Miller's Law. In his 1956 study [151], Miller observed that the average short-term memory span was limited to approximately seven chunks[33], plus or minus two, regardless of stimuli consisting of different amounts of information. A common misinterpretation of this law is that people can only hold around seven pieces of information at one time in short-term memory, but this is in fact an incomplete understanding. Miller tested *chunks* of different information types and densities, concluding that memory span is not limited by the individual bits of information that make up chunks, but rather the chunks themselves. This distinction between several items and several chunks is important, as chunks can differ between individuals. Consider a single word made up of ten letters. To a native speaker, these ten pieces of information are likely to be a single chunk, whereas to a new learner of the language, each letter may be its own chunk. When applied to authoring tool design, chunking of information makes the interface easier to comprehend, and consequently easier to scan the content to identify what aligns with their task or to consume the information more quickly. Figure 2.27 shows the same segment of an interactive story written in *Inform* (left) and copied into plain text (right). Both versions divide the text into six distinct chunks separated by whitespace, but the authoring tool, with its augmented text editor, can provide additional mechanisms for segmenting chunks at a different granularity than can be achieved in plain text. For example, we can see that all text strings are emboldened and are shaded blue, comments are in a smaller font size and are shaded green, and chapter headings are in a larger font size, bold, and are shaded a deeper black. These additional features supported by the interface allow for the user to mentally map chunks at different levels and enhance the ability to find specific types of content despite the underlying data being simply text. In contrast, finding all instances of quoted text in the plain text version at a glance is a lot more challenging.

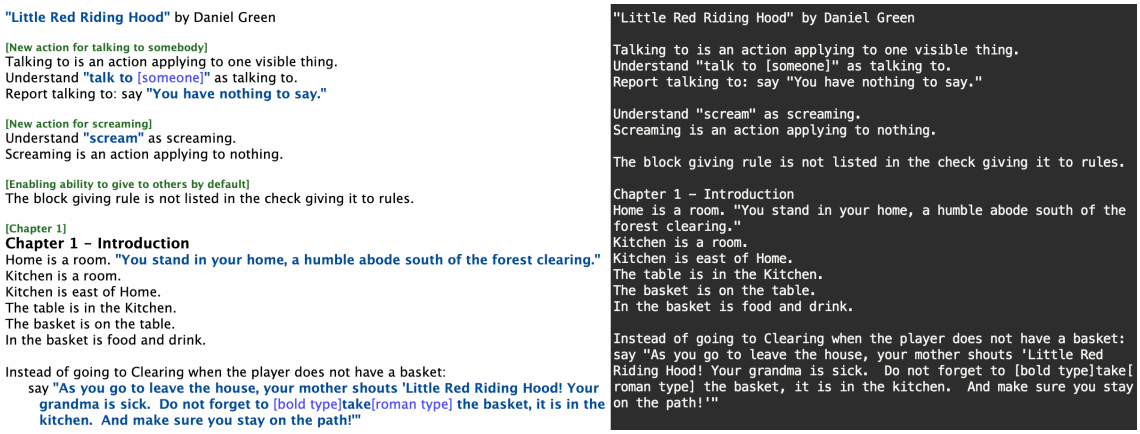


Fig. 2.27 The same block of text from an interactive narrative in *Inform* (left) and in plain text (right) demonstrating visual enhancements for identifying information as a result of considering *Miller's Law*.

Serial Position Effect. In his 1913 study [61], Ebbinghaus coined the *serial position effect* which describes the tendency for people to better recall items presented at the beginning and end of sequences rather than in the middle. The consequence for user interface design is that long sequences of information can become overwhelming and require a high cognitive load to accurately recall the location of items when needed. When dealing with short sequences, this problem is minimal, but as sequences grow, short-term memory becomes problematic, as outlined in *Miller's Law*, meaning that accurate recall is impacted. Utilizing this in authoring tool design is a matter of acknowledging the ramifications of displaying long sequences and attempting to mitigate the associated natural mental limitations. An example of this can be found in the *Genarrator* authoring tool as shown in Figure 2.28. Each page in the narrative is listed in a single horizontal sequence at the bottom of the tool and is the only way to navigate to each page within the editor. Ordering items like this in a sequential list can induce the serial position effect, particularly as the number of pages in the narrative grows. However, *Genarrator* attempts to mitigate this in part by providing customizable labels for each page that are displayed inline within the listing. While this does not entirely solve the problem, as position information in long sequences is still unclear, it does provide additional metadata that can help to contextualize positions within the list easier.

Fig. 2.28 A horizontal sequence of all pages within a narrative in the *Genarrator* authoring tool, demonstrating a mitigation of the *Serial Position Effect* by including customizable inline labels for each page.

Use of Heuristics

Another way to inform design decisions while developing and to validate the usability of authoring tools both during and after development is through the use of heuristics and heuristic analysis. These techniques have software design informed by and tested against established guidelines and best practices that either avoid or highlight potential usability problems. These applied heuristics vary but are often sourced from academic works, international standards, or otherwise widely accepted axioms within user experience, such as the various rules discussed above. When conducting a heuristic analysis of a given software, one gathers appropriate heuristics and iterates through them alongside the software in order to identify any violations that occur. As with most rules of design, they are rules of thumb rather than being absolute, and often rely upon the proper judgement and experience of the person conducting the analysis. It is therefore optimal to have a usability expert perform the heuristic analysis and indeed many user experience consultants offer such a service[34]. If developing an application that will be deployed on a popular app store, a typical part of the submission process involves professional usability

reviews using heuristic analysis to ensure that developers have adequately followed established design heuristics in their software[35]. A design violating a given heuristic does not necessarily constitute an inherent failure of the design, as in certain situations it may be necessary to not follow standard guidelines. However, if this occurs frequently or greatly violates a given heuristic, then one should be careful in the deviation from accepted methods as to avoid introducing potential usability troubles.

One of the most well-established heuristics sets is *Nielsen's Ten Usability Heuristics* [163] created in 1994 by Jakob Nielsen, which is still in wide use today despite being developed in a time when user interfaces were not as mature or interactive. The ten heuristics cover a wide variety of topics that could induce usability problems if sufficient attention is not paid, including *visibility of system status*, *match between system and the real world*, *user control and freedom*, *consistency and standards*, *error prevention*, *recognition rather than recall*, *flexibility and efficiency of use*, *aesthetic and minimalist design*, *help users recognize, diagnose, and recover from errors*, and *help and documentation*. One of the benefits of such heuristics is that they are robust. Nielsen himself highlights[36] that although the language of the definitions may have been refined, the substance of the heuristics has not changed in over 26 years, yet remain just as, if not more applicable, meaning that they are likely to apply to future generations of user interfaces also.

In some cases, heuristics may be developed for a specific domain. Naturally, this limits their applicability to a tighter scope than general heuristics, but has the significant advantage that the heuristics can be defined at a much finer granularity, targeting specific elements special to the field. An example of this can be found in Desurvire's work on HEP [53], GAP [57], PLAY [56], and VR PLAY [54], all of which expanded upon general usability heuristics (including those by Nielsen) and other established guidelines to create heuristics lists specific to game playability (HEP), game approachability (GAP), game usability (PLAY), and best practices for virtual reality experiences (VR PLAY).

Other commonly applied heuristics include *Shneiderman's Eight Golden Rules of Interface Design* [194] which outlined eight rules of thumb derived from experience spanning two decades of user interface research, *Gerhardt-Powals' Cognitive Engineering Principles* [75] which presented and demonstrated ten design heuristics for "cognitively friendly" user interface design (i.e., user interfaces which reduce cognitive burden by having strong usability), and the *Guidelines For Designing User Interface Software*[37] which specified hundreds of rules of thumb for user interface design divided into six categories (*data entry*, *data display*, *sequence control*, *user guidance*, *data transmission*, and *data protection*). Moreover, it is not uncommon to use standards documents as design and analytical heuristics, such as *Human-Centred Design for Interactive Systems (ISO 9241-210:2019)* and the *Guidelines for Designing User Interface Software (ESD-TR-86-278)*.

User Research

If at the stage in development where an artifact can be presented to users, whether that be a functioning piece of software, interactive prototypes that mimic the intended experience to a satisfactory level, or even early design sketches, then we are able to utilize the partnering of qualitative user research methodologies with established quantitative usability surveys to aid in understanding and refining the usability of the system, and even compare between revisions of a design. The involvement of users allows us to gain insight into their behaviors and opinions that would otherwise be missed.

A somewhat recent appraisal of the field identified a total of 24 questionnaire-based usability surveys, concluding that the majority of methods have very high reliability, that they target a wide range of interface types (although a few are specific to a given domain), and that a majority are based on Likert scales (with dichotomous and semantic scales being used less frequently) [9]. Some of the most popular metrics for measuring usability are the *System Usability Scale* [32] (SUS), the *Usability Metric for User Experience* [68] (UMUX), and *UMUX-LITE* [131]. SUS is a ten-item questionnaire that uses a five-point bipolar agreement scale, resulting in an easy-to-understand usability score ranging from 0–100 in increments of 2.5. Although the authors of SUS described it as a "quick and dirty" usability scale, it has excellent reliability [13] and has been used in a vast range of both academic and non-academic works [128]. Moreover, it is possible to eliminate one of the ten questions if desired whilst retaining a resulting accuracy within one point, if so desired [129]. UMUX is a four-item questionnaire that uses a seven-point unipolar agreement scale to measure the subjective assessment of the perceived usability of a system, resulting in a score ranging from 0–100, designed to be comparable to the output of SUS despite having fewer total questions. UMUX-LITE, on the other hand, is an adaptation of UMUX to further shorten the length to only two questions yet maintain a high degree of reliability. The authors demonstrate how the resulting scores are comparable to UMUX and even correlate to SUS scores despite its significantly shorter length [129]. Other researchers have investigated the regression formula used to map between the outputs of UMUX-LITE and SUS with an independent set of data, finding that the outputs remained comparable [130]. The motivation behind UMUX-LITE is not to ultimately replace UMUX or SUS, but to provide an alternative system for scenarios where limiting the overall number of questions is an important factor. SUS, UMUX, UMUX-LITE, and the other available usability surveys are not designed to be a comprehensive method of discovering usability problems, but instead to perform a quick and general usability assessment based on the perception of the users. As outlined by Nielsen in his seminal book *Usability Engineering* [166], usability surveys such as those described above are not a replacement for observation as they do not study the user experience itself, but rather the users' opinions of the experience. Moreover, he highlights how users will unintentionally yet frequently mistakenly claim about their perceived experience when filling out surveys. Nielsen provides several examples of this in various contexts, such as how 26% of users

had mistakenly claimed to not know a terminal command, but later described using the command within the same survey [179], or how 24/25 users of a telephone system stated that the instructions of the system were simple to understand, yet when performing the task, only succeeded on average 50% of the time; the users thought they had understood the usability of the system, but in fact they had not. This highlights that while usability surveys come in many varieties and are overwhelmingly popular, they do come with the bias that users are self-reporting their own perceptions of the user experience, which is well understood to often include unintentionally false claims about the actual usability of the system. The aforementioned standardized survey techniques are proven to be adept at measuring the general usability of a system. However, in practice, this general metric of overall usability may not be desirable, especially if we are trying to identify the usability of a particular aspect or to answer specific questions. In such cases, we must employ surveys with domain-specific questions, and instead follow informed survey design practices such as choosing suitable types of measurement scales [36, 118, 188], the potential inclusion of data validation questions [10], and use of appropriate language to mitigate misinterpretation, to avoid leading participants to specific responses, and to generally maintain data quality.

There exists a vast array of qualitative user research methodologies that can be used in conjunction with quantitative surveys discussed above. Choosing an appropriate methodology is a skill in its own right and requires working knowledge of the advantages and disadvantages of each, their applicability to the problem being solved, and the logistics of preparing and executing them. Moreover, as with the creation of domain-specific surveys, these existing techniques are not strictly prescriptive, and methodologies can be appropriately modified or combined based upon the study or participant needs [81, 137, 166, 182]. A complete review of user research methodologies is a thesis unto itself, so I will instead briefly discuss several relevant methodologies that are helpful within the context of authoring tool design research.

Usability Testing. A popular method for evaluating where users typically encounter problems is usability testing, also referred to as user testing [137]. With this method, a user is given tasks to perform in a product or service — in our case this would be typical interactions with authoring tools — and are observed by researchers while doing so. Tasks are usually specific to a topic of interest (e.g., targeting a particular feature) but it is also possible to use open-ended tasks (e.g., providing minimal information and allowing the user to uncover a solution themselves) which reveals how users navigate and understand your system. While this methodology works unmoderated (i.e., the researcher is not present or at least doesn't interact during the completion of tasks), the richness of the data gathered benefits greatly from being able to not only observe participants, but to also interact with them through appropriate questions and answers in the moment. Although this method is relatively quick to learn and execute, becoming a skilled moderator requires significant effort [81, 137, 182] as to avoid accidentally

introducing bias (most commonly by talking too much, explaining the design, answering detailed questions, performing mostly interviews rather than testing, and soliciting opinions and personal preferences [212]). Execution of this methodology typically has four stages [81, 137] — introducing the participant to the study, a short pre-task interview, performing the actual tasks while being observed, and a short post-task interview and debrief. During the performance of the tasks, it is common to ask the user to think aloud their actions, which provides a deeper insight into participants' mental processes while doing the tasks. However, the quality of data gathered through this differs greatly based on the individual and can introduce several biases such as downplaying struggles to maintain social status, interjecting with an opinion on how to fix problems for other users, and being slowed down by the extra cognitive load of maintaining verbalization of their thoughts [137, 166]. Ultimately, the specific configuration of this methodology is determined by the goal of the research. Regardless, usability testing allows us to gain insight into how users go about interacting with our software, and more importantly where they struggle, with additional data being gathered through think-aloud protocol and questions during task completion. Moreover, this technique is actively used and encouraged in the creation of game development tools [132].

Contextual Inquiry. Another common method of evaluating usability is contextual inquiry [25, 218], which can be seen as a hybrid of ethnography — examining a subject or phenomenon in its native environment (as opposed to artificial) — and user interviews [137]. In this method, the participant is the expert, performing typical routines in their own environment, which makes the gathered data more realistic[38]. As the participant works, they are observed by a researcher who takes appropriate notes. The researcher may also periodically interject to discuss the participant's actions. According to the original authors of this method [25, 218], there are four key principles to a successful contextual inquiry — *focus* (have a clear understanding of the purpose of observation), *context* (observe participants in their own environment), *partnership* (talk to participants about their work and engage in discussion), and *interpretation* (develop a shared understanding with participants about aspects of their work that matter to them). Moreover, they suggest that by following these key principles, you can gain an understanding of the tools that people use, the sequences in which they perform actions, their methods of organization, and what kinds of interactions they have. Typically, a contextual inquiry is run almost identically to a usability study as described above. The major differences between the two methods are that contextual inquiry takes place in the participant's natural environment and that they perform their own tasks rather than ones exclusively provided by the researcher. It is also possible to combine usability testing and contextual inquiry by having users perform predetermined tasks in addition to their own regular tasks[39]. As with usability testing, this technique is also actively used and encouraged in the creation of game development tools [132].

Content Testing. An important component of the usability of software is how suitable and understandable the actual content of the user interface is. As developers of authoring tool software, the meaning of the content that we create is clear to us, but developers are rarely the target audience of such software[40]. Content testing [137] is a methodology that aims to solve this problem by determining how suitable and understandable your content is for the intended audience, performed as either a standalone test or as part of a greater usability test. Including content testing as part of the overall design strategy can help us to better organize, design, label, and otherwise signpost the user interface to be easier to understand and comprehend at the surface level (e.g., the layout of elements), at an individual action level (e.g., is the purpose of a button clear), and at the procedure level (e.g., are the sequential steps required to achieve a given outcome clear). Content testing can be performed at any stage of the development cycle, and although it benefits from being contextualized, this is not a necessity, particularly during earlier development stages where context may not yet have been established or is not yet usable. There are several variations of this methodology, each aiming to uncover different usability information about the same content. *Comprehension Testing* involves asking open-ended questions (e.g., "What does this mean to you?") about the meaning of things, typically words or phrases, to determine how they interpret what the content means. This can be presented in context of a user interface (e.g., buttons, menus, icons) or simplified by presenting a standalone list of words without further context. It is also possible to use task-oriented questions to determine if participants have understood the content based on what they have read. For example, given the content of a document, asking if they could explain a concept or envision how they would take action to complete a task. *Cloze Testing* is similar to comprehension testing of phrases, but parts are removed and participants must attempt to fill in the removed content. You can accept only specific answers or choose to accept synonyms of the expected outcome. If enough participants are sampled, this method can become quantitative to determine the kind of language that the target audience understands, which can help to reduce reliance upon technical jargon. *Choose Words* presents content in context but with appropriate labels removed — such as an authoring tool interface with empty button labels — and has participants fill in their own labels based on their understanding of the content. As with Cloze Testing, given enough participants, quantitative inferences can be made about the language that the target audience expects. *Highlighting* takes one or more phrases, either in context or as standalone items, and has participants highlight anything they are confident about in green and anything they are uncertain about in red. As with the previous two quantitative variations, given enough participants, it is possible to analyze the frequency of colorings to gain a broad understanding of words or phrases that were difficult to understand and those that were easy to comprehend.

CHAPTER 3

Modeling Video Game Narrative for Authorship

This chapter will begin by proposing a complete high-level architecture for the authoring of video game narrative. It does so primarily to aid in answering the first research question by providing a framework within which to design and shape a theoretical model of video game narrative rather than to develop it in isolation which risks the model being disconnected from the wider context. The modeling proper will then begin, starting with an exploration of existing modeling approaches, followed by applying a variety of models to two contemporary video games that differ in narrative complexity in order to better understand the affordances and constraints of these existing models in the context of video game narrative. In response to this analysis, my own high-level descriptive concept of *Discoverable Narrative* is introduced and demonstrated. This is followed by the formulation of my own complete video game narrative model, *Novella*, which is designed in response to the analysis and is influenced by the overall proposed architecture, accompanied by multiple worked examples to demonstrate its functionality.

Much of this chapter is presented in several publications including the proposed architecture [87], the exploration of existing modeling approaches [87], a summary of the application of models [89], the descriptive concept of *Discoverable Narrative* [83, 89], and the *Novella* narrative model and its corresponding examples [87, 89].

3.1 An Architecture for Game Narrative Authorship

As discussed in §2.4.3, authoring tools for video game narrative do not exist in isolation and are in fact a component within a larger authoring pipeline that eventually realizes the creative work in a medium beyond the tool. This concept also applies to certain kinds of narrative models, chiefly those that serve as the underlying technology of runtime systems or otherwise concern themselves with practical integration beyond the tool. This is because, as with authoring tools, there is a greater context within which these kinds of models exist, and to create them in isolation risks a disconnect between the model and

its neighboring components. Developing models within context enables us to consider the greater pipeline that follows and the technologies that may be utilizing the model. In turn, the model's design can consider these otherwise external factors and improve the interoperability and functionality of the model accordingly.

For the purposes of this thesis, the proposal of a complete architecture for authoring game narrative has a number of advantages. As mentioned above, practical models do not exist in isolation and are in fact a part of a greater whole, and when being designed within that context, are able to leverage the fact and better situate themselves in a holistic manner rather than in isolation. This is particularly important when answering the first research question which asks how we may build a game narrative model, as by being situated within a game narrative authoring architecture, the model can better capture details of game narrative that may otherwise be missed. It also serves to validate and demonstrate the practicality of a developed model, as when designing or implementing a subsystem that relies upon the model, they will not function correctly if the model is not sufficient, and any problems with the model will become apparent. This is also true of theoretical worked examples of the architecture, which in some sense serve as a dry run of the model in relation to its sibling components and the overall authoring process.

Such an architecture should, at a high level, describe all layers of the creation process from authoring through integration and realization, which helps to further contextualize and influence an underlying narrative model. Additionally, the architecture should not make any assumptions about a specific realization method of the authored product. While it is impractical to satisfy all realization methods in a single architecture, striving to remain neutral such that the authored product can be deployed in various contexts ensures flexibility. Similarly, the underlying narrative model should aim to distance itself from genre-specific tropes to remain capable of representing a wide array of types of narrative. Incorporating tropes found in genres of narrative can be troublesome as it introduces assumptions that can impact the generality of the model. For instance, the *Inform* authoring tool, while positioning itself as a way of "creating works of interactive fiction"[41], introduces biases as to how the game world is handled, such as the inherent requirement of room-based layouts and interaction paradigms reminiscent of adventure games. This is not necessarily a weakness as an authoring tool may purposefully introduce limitations to better serve a particular subset of narrative genre, but by doing so inherently restricts the kinds of narrative experience that the system is able to produce.

3.1.1 Architecture Overview

Figure 3.1 depicts my proposed architecture for authoring game narrative which consists of several internal components (a narrative model, an authoring tool, an interchange format, and an API) and two external components (the engine and product) that use the internal components to complete the realization of an authored narrative.

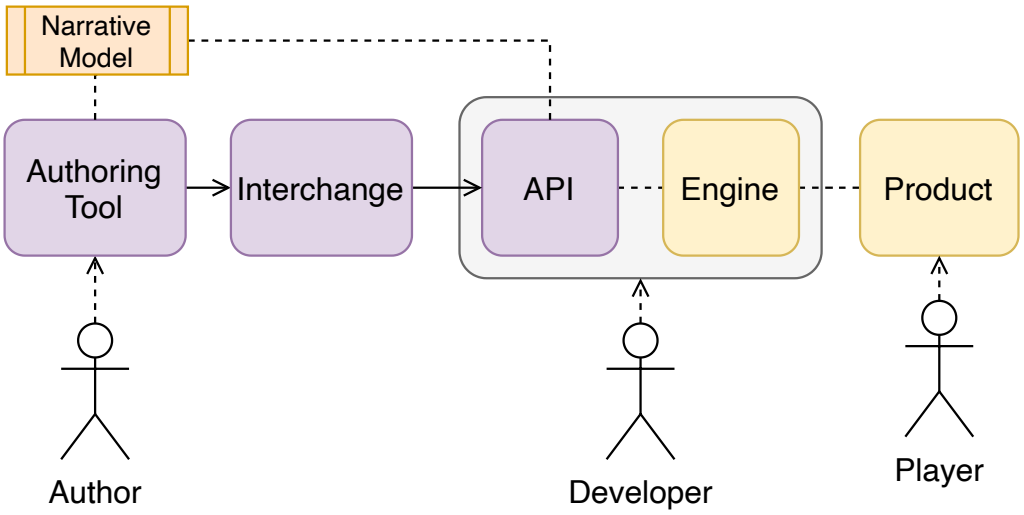


Fig. 3.1 The proposed architecture for authoring of game narrative consisting of several dependent or interacting components.

The narrative model is the underlying representation of the narrative's structure and its constituents. One of the desired factors of this architecture, as mentioned above, is to remain distant from genre-specific tropes. This is not something that can be enforced at an architectural level as it is not what informs genre, but the relationship between the model and the other components can instead influence the model's design in an actual implementation of this architecture. For example, if the API and model were being developed in tandem, the API could request genre-specific features that the model would have to respect, although in an ideal situation, the model drives the API.

The API of this architecture provides a programming interface for developers to integrate the narrative model into their product through its runtime engine. It does so by providing functionality for management (create/read/update/delete) and simulation of the narrative model. In this sense, the API is essentially a library that implements the narrative model for direct use within an existing codebase. It should be possible for a developer to solely use the API to utilize the narrative model within a product through the engine. However, this is impractical as a permanent authoring solution as it enforces hardcoded narratives or requires developers to individually create their own intermediate format which may not fully support the narrative model. The interchange component assists with this by providing a predefined intermediate format that can be loaded into the API without extra work. This step essentially automates manual use of the API, although depending upon the complexity of the implementation, it is possible that some manual work with the API may still be required. By using an API coupled with a predefined intermediate format, the architecture remains neutral to how the end product is realized. Even with multiplatform development, an adapter for the API (or even multiple distributions of the same API) could be created, which in turn equally load the platform-agnostic interchange format. The result is that a narrative can be authored

in a generalized interchange format and, with minimal developer intervention, be loaded and used in various runtimes.

The architecture does not determine the format of the interchange; it is equally viable to be represented as ASCII as it is in binary. When in an ASCII format (such as JSON or YAML), it becomes human-readable and writable, meaning that narratives can be created and edited from a text file (bar essential developer integration using the API, as discussed). While convenient, this is not a solution that is friendly to authors, particularly those that may not be technically inclined. The authoring tool software component assists with this by providing an interface to create and edit a narrative in an abstracted form of the narrative model and allowing for export of the interchange file, the benefits of which were discussed in the 'Export to Intermediate File' paradigm in §2.4.3. An authoring tool that is integrated into a given engine has the advantage of being able to directly reference in-game object instances or other engine assets without the use of an interchange format, but this inherently limits where the narrative can be deployed and imposes runtime-specific limitations (such as data formats or object reference types). On the other hand, choosing a standalone authoring tool, while requiring an interchange format due to lack of direct connectivity to a runtime, no longer accustoms itself with a specific system, therefore remaining neutral to the way in which the narrative is realized beyond the intermediate format. The advantages and disadvantages of various authoring tool delivery methods were discussed in detail in §2.4.2.

3.2 Exploration & Application of Existing Models

The very foundation of hypertext — distinct material interconnected in a complex way — makes it suitable for structural modeling, as we can describe its atomic content and the relationships that join it together. Consequently, when we observe other fields through a hypertextual lens, they too become candidates for structural modeling for their specific domain. As established throughout §2, hypertext theory is already utilized in this regard in a range of fields including interactive fiction, authoring tools, and video game narrative.

Video game narrative, which as discussed in §2.3 contains elements with qualities suitable for narrative analysis, has comparatively fewer domain-specific models and introduces a variety of complexities that could complicate verbatim application of existing models. Unlike early hypertext systems, hypertext fiction, HTML, and the World Wide Web, which each emerged directly from hypertext theory as covered in §2.2, video games did not originate from hypertext theory with early titles such as the multiplayer shooter *Spacewar!* and text-based strategy *The Sumerian Game* predating the introduction of the hypertext concept[42]. However, there is no doubt that hypertext began to influence the format and structure of video game narrative early on, particularly as text adventures became popular in the 1980s, which leveraged significant functionality from hypertext theory, as demonstrated in §2.2. It is plausible that this initial separation and nicheness of

early video games resulted in contemporary hypertextual fields receiving more attention in terms of structural modeling. Some hypertextual models, particularly those that do not target interactive narrative or that predate contemporary video game narrative techniques, will be at a clear disadvantage when tackling these unexpected and newfound elements and may require manipulation to better suit the context of video game narrative.

Structural models are typically defined as either conceptual, data, runtime, or a mixture of the three. Conceptual models often capture and define narrative elements at a high level, grouping things together to form abstractions of more granular concepts. For example, characters within a narrative may be represented as one of several archetypes (e.g., as in Aarseth's work [3], Propp's Morphology [174], or Campbell's Monomyth [37]). These models excel in the simplification of complex concepts into something that's easy to grasp, but do so at the expense of detail. Data models, on the other hand, break down narrative elements into finer structures (in the computer science sense) and define relationships between those structures. For example, individual objects within the game world and their relationship to the overall story state may be described as low-level computer-readable data structures. A strength of these models is their ability to accurately define and describe a wide variety of narrative aspects, typically readable by a computer, with great detail due to their fine granularity. Runtime models concern themselves with, in some form, execution of the model's own narrative representation and logical behavior. To some degree, they build upon data models in that they provide the framework within which the structural narrative representation and behavioral logic can execute. In some cases, runtime models may also integrate with external platforms to directly use the external representation of narrative objects and game world state, such as how the *Fungus* narrative plugin for *Unity* can directly access objects and is in fact executed within the game engine itself based on rules that the plugin defines. It is important to consider all model types as they each demonstrate a different focus when representing narrative elements. Conceptual models, for instance, better lend themselves to abstractions that could be utilized in a user interface, whereas data and runtime models can inspire the structure within new models.

This section will begin by outlining two common approaches for game narrative modeling — the adaptation of existing models and the creation of targeted models — to provide a base understanding of how others have approached this task. This is then followed by an application of four different models to two contemporary games, *Portal* and *The Stanley Parable*, to better understand fidelity with which existing models can capture video game narrative — that is, what can be represented in the models and what cannot. These games were chosen due to their opposing structural complexities and short duration; the narrative structure of *Portal* is linear, but *The Stanley Parable* is comparatively interwoven and complex. This ensures that both structurally simple and structurally complex game narratives are broadly represented in this exploratory application as opposed to choosing two simple or two complex games instead. Both games

utilize a wide array of interactional methods and presentational methods for narrative delivery as discussed in §2.3, which are used throughout many other games. For example, both games make frequent use of optional content and environmental storytelling to convey narrative, rely heavily upon triggered events for narrative progression as the player navigates and interacts with the game world, and portray narrative through cutscenes. Moreover, they are a pragmatic choice for analysis as I am already familiar with their narratives and structures. Each game was played through multiple times, keeping detailed logs of each scene, interactions taken as a player, narrative entities involved, and potential divergence of the story. This information, combined with various online resources such as walkthroughs and wiki pages, resulted in a detailed set of annotations that could be used to apply the models.

A summary of the application that follows has been published at the *Narrative and Hypertext Workshop* alongside the *Hypertext 2018* conference [89].

3.2.1 Approaches to Game Narrative Modeling

Adding Interactivity to Existing Models

A common approach to modeling video game narrative is to take existing theories that are not based on games and attempt to add support for interactivity or otherwise modify them to better suit games' representation and analysis.

One such attempt by Delmas, Champagnat, and Augeraud is to inject interactivity into Campbell's Hero's Journey [51]. The original theory by Campbell [37] specifies 17 phases of what he refers to as the 'monomyth' divided into one of three acts (departure, initiation, return). According to Campbell, works of fiction typically adhere to this three-act structure, utilizing one or more of the phases in each act. Each phase describes archetypical behaviors found in works of fiction, centered around the hero of the narrative, such that the phase can be used as a template for any hero character. This predetermined string of commonly occurring events makes it ideal for mapping linear narratives that conform to the narrative progression outlined by the model. In a linear narrative, the author has full control over the hero's choices and the resolution to these choices, whereas in a video game, the player is often given genuine choice and resolutions are conditional to the choices, which is in direct contrast with fixed linearity. Moreover, it is possible for players to fail tasks, misunderstand tasks, or outright refuse them, which is not supported in a verbatim application of Hero's Journey. In adapting Hero's Journey to support video game narrative, the authors note three considerations that must be made. Firstly, the player must remain at the center of the action and not be delegated to a passive role. Any important action of a given phase must occur by and with the player, with the exception of events that may be used to cause pressure on the player. In this sense, each phase must account for the player being at the center of the action and ensure that its resolution is dependent upon the player's actions. Secondly, each time the player makes a choice

(such as accepting or refusing help), each potential solution must be valid to allow for progression of the narrative, considering both player failures as well as successes. In the event of failure, it is not uncommon for games to make the player reiterate from a recent point or even repeat the game. In some cases, such as player death, this is unavoidable. However, the authors argue that to properly support phases in games, they must be permissive to failure by not blocking narrative progression, instead applying handicaps. Thirdly, phases must be subject to conditional release based on a player's individual abilities. For example, stages such as *Supernatural Aid* or *Rescue from Without* give the hero support in their trials, but the player's ability to succeed may determine whether these phases are needed, becoming conditionally present based on player success. With these considerations in mind, the authors alter the rules of how phases of Hero's Journey are delivered and append three new phases specific to video game narrative. The first addition is the concept of an *Optional* node which serves to dynamically branch between phases based on a conditional output (accept, refuse, success, failure, help needed) of the preceding phase. Two of the new phases, *Stubborn Refusal* and *Compelled to Adventure*, are introduced to handle cases where players make a stronger refusal. If after the *Refusal of the Call* phase the player still does not want to accept, then appropriate events are delivered in the new *Stubborn Refusal* phase in an attempt to persuade the player, and if that ultimately fails, the *Compelled to Adventure* phase forcefully draws the player's hero character to action. If the player chooses not to return in the *Refusal of Return* phase, then the newly introduced *Interference from Without* can intervene in an attempt to motivate the player to return. The player is still free to decline, at which point the player can leave the narrative sequence without yet completing the quest (analogous to delaying the completion of a quest in an RPG, for example).

Figure 3.2 shows the authors' interactive adaptation of Hero's Journey with frequent use of the *Optional* node joining together phases and the inclusion of the three new phases highlighted in red. These alterations to the structure of Hero's Journey allow the model to react to important decisions made by the player, their abilities, and their successes or failures. It also allows for skipping of phases through the *Optional* node and the change in delivery rules that it brings. However, despite its improvements for supporting player behavior and choice, it is still bound to narrative representation at a comparable granularity of the original model's phases.

A similar approach was taken in the Narratification project [198], which attempted to unite narrative and gameplay in an analytical framework. This work is a game-centric extension of Dixon's Goal, Motivation, and Conflict (GMC) [58] model. GMC is a model which aids novelists and scriptwriters in developing plots that are character-centric, focusing on the character's goals, motivations, and conflicts, listing internal (known to oneself) and external (public knowledge) factors of each. In this context, goals refer to objectives a character would like to achieve, motivations define why they want to achieve their objectives, and conflicts determine what is stopping them from reaching

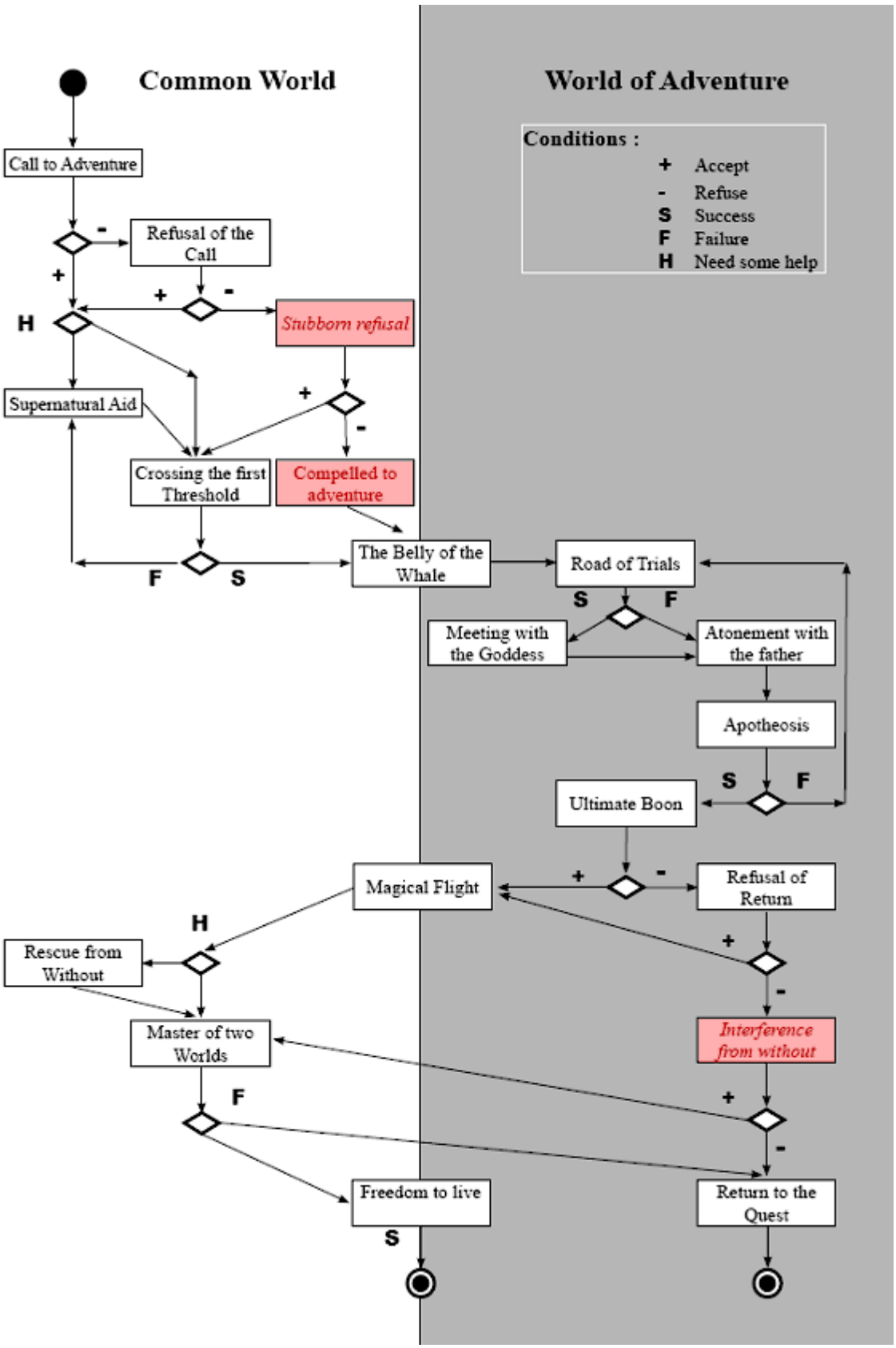


Fig. 3.2 An interactive adaptation of Hero's Journey by Delmas *et al.* [51] showing use of *Optional* nodes and the new *Stubborn Refusal*, *Compelled to Adventure*, and *Interference from Without* nodes highlighted in red.

their objectives. Table 3.1, presented as an example by the authors, demonstrates Harry Potter's goals, motivations, and conflicts in *Harry Potter and the Philosopher's Stone*, which provides a high-level summary of the plot, with external factors representing physical and visible elements of the plot relating to Harry, and internal factors representing his private considerations. The authors highlight that this model is ideal for understanding and developing meaningful plots based on a character's internal and external factors, but note that it does not capture any dimension of interactivity from the player when applying it to video games.

Table 3.1 An example of Dixon's Goal, Motivation, and Conflict model [58] applied to Harry Potter from *Harry Potter and the Philosopher's Stone* by Sørensen *et al.* [198].

Goals	**Internal** Investigate, find out who he is and why Voldemort tried to kill him; make his parents proud. **External** Survive; become a wizard; pass exams; do good.
Motivations	**Internal** Feel at home and like he's part of something; get the family he never had; figure out who he is. **External** Escape from his evil stepfamily; be at Hogwarts; be with his friends; discover mystical and personal secrets.
Conflicts	**Internal** Ghosts from his past; insecurity; fear of discovering dark secrets about himself. **External** Voldemort; dangers; enemies; exams.

To remedy this, the authors propose the Interactive Goal, Motivation, and Conflict (IGMC) model as an extension of GMC, pitching it as a tool for developing plot and interaction between a human player and their in-game character to reach a symbiosis between gameplay and narrative. An extra column is added to the GMC matrix representing the player themselves, separate from the avatar that the player controls. When using IGMC, it is possible to gain an additional holistic view of the game experience beyond that which GMC provides. Moreover, the new player column demonstrates the winning conditions of the game through the goal and conflict entries, and motivation describes the reasons that players choose to play the game. It is noted, however, that despite the player and character being separated, they should be appropriately linked, such as the motivation of the player being related to why the character is motivated in the game itself. Table 3.2 shows the IGMC model applied to *Max Payne*, again provided by the authors, which demonstrates the relationships between player experience and the in-game character's goals, motivations, and conflicts.

Propp's Morphology has also been indirectly applied to video game narrative. In work by Brusentsev *et al.* [34], Propp's Morphology is used as the basis of a game analysis

Table 3.2 An example of the Interactive Goal, Motivation, and Conflict model [198] applied to *Max Payne* showing relationships between player experience and the main character.

	Player	Max Payne
Goals	Beat the gangster mob responsible for Max's agony. Do everything guns blazing (win condition).	**Internal** Inner peace; redeem memory of family after their death. **External** Revenge; clear his name; find those responsible.
Motivations	*Investigating* the truth behind killing of Max's family. *Empathetic identification* with Max. *Fun* and *challenging* gameplay makes win condition neutral; the game invites the player to go on a killing spree. *Atmosphere* has a noir feel, monologue, and experience.	**Internal** Feels responsible and guilty; couldn't protect his family; has nothing left to lose. **External** Justice; he is a police officer; his search for justice and revenge is his prior motivation and justification for killing.
Conflicts	Many hostiles. As Max, the player is being hunted by both police and gangsters.	**Internal** Guilt weighs him down; he is not a killer but has nothing left to lose; accepts what's left of his humanity. **External** He is a one man army; police are hunting him as a suspect; he in on the run and simultaneously infiltrating the mafia.

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 are 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 *et al.* [29]. 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 & Factorization

Another approach is the use 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.

One such practice 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 Björk *et al.* [27], further refined in later contributions [123]. 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 [221] 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 [42]. This concept has also been taken further to determine the effect of these kinds of components on the narrative [30] by surveying 80 games and identifying relationships that emerge between components and various narrative forms.

A similar approach was taken by Bizzochi [26] 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 [3]. Aarseth

breaks up games into the game world, objects, agents, and events. The world represents the physical structure of the game's level layout in a way reminiscent of Adams' structural layouts [6] and differentiates between ludic (playable) and extra-ludic (non-playable) spaces. 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. The concept of shallow and deep characters is inspired by Forster's definitions of flat and round characters [70]. 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. The concept of distinguishing between obligatory events that shape the story versus those that are optional for supplementary action is derived from the works of Roland Barthes [17].

3.2.2 Applying Propp's Morphology

We can now begin the application of models to *Portal* and *The Stanley Parable* to better understand the affordances and constraints of these existing models in the context of video game narrative, beginning with Propp's Morphology.

For this application, Propp's Morphology was not used in its original form as described in §2.1, but instead as an amalgam of the original form and two modern game-centric variations. Work by Brusentsev *et al.* [34] investigated the capacity of the original functions and character archetypes to capture specific types of video game narrative and outlined a complimentary framework used when applying Propp's Morphology to games. This framework consists of seven descriptive factors which provide insight into the narrative structure, interactivity, and characters within the game. Bostan *et al.* [29] also modified Propp's Morphology, focusing on the functions and their rules. Some of the existing functions were modified and 15 new functions were added to better support the tropes found within video game narrative. Additionally, the constraints of function ordering were relaxed in order to allow for mapping of more complex structures than those found in the initial folktales that informed the original functions and rules. Combining these three approaches gives us a resulting framework based on the original theory that is better suited for video game analysis. It is mostly a conceptual model, but with the included rule changes, it also has aspects of a data model.

For both games, the application of the framework was done by firstly populating the seven descriptive factors to better understand the structure of each narrative, the interactions the player can make, and the characters involved in the stories. Based on this overview, Propp's character archetypes were assigned to each of the characters where applicable. As concluded by Brusentsev *et al.* [34], the archetypes map well to video game characters. However, in my own application, I found that some were challenging to

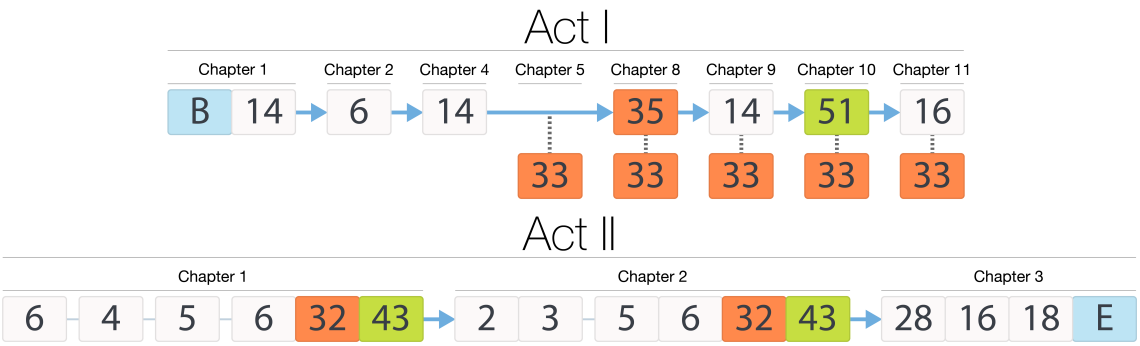
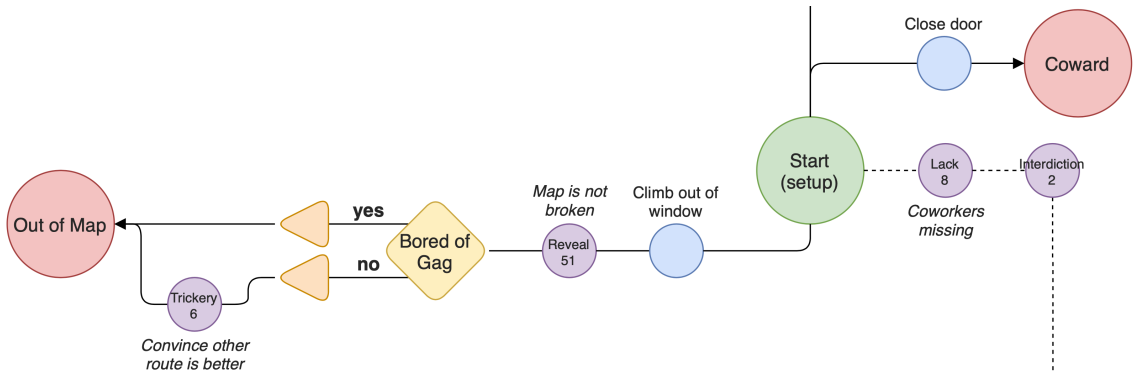


Fig. 3.3 The structure of *Portal* mapped to the modified Proppian functions and rules of Bostan *et al.* [29]. Each number represents a corresponding function, with modified functions highlighted in orange and new functions in green. Optional pathways are connected with dotted lines. *B* and *E* are the beginning and end respectively.

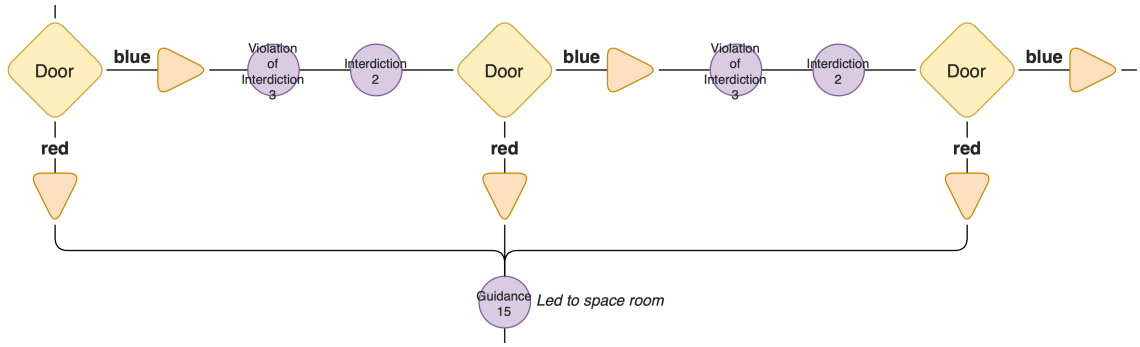
assign when considering transitions between archetypes. In *Portal*, for example, GLaDOS transitions between being a Helper into a Villain, but Propp's character archetypes excel at static personalities, not those that transition. This behavior was also found in a particular branch of *The Stanley Parable* where the main narrator transitions from being a Helper into a Villain. A small number of characters had no suitable mapping at all. The three Personality Cores of GLaDOS in *Portal*, for example, each have a distinct personality and unique traits, but serve as narrative coherence and as a gameplay mechanic; they do not fit well into any of the predefined categories.

Both games' stories were then mapped based on the modified Proppian functions and rules of Bostan *et al.* [29]. Due to the differences in structural complexity, each game was mapped at a different level of granularity. In the case of *Portal*, the game officially lists 11 chapters, but this is misleading as the last chapter contains almost half of the game. Instead, the game was divided into two acts. The first act is composed of 19 'test chambers' spanning the original 11 chapters, and the second act continues after the player's near-death experience following the chambers, which is further subdivided into three chapters — initial escape, villain tracking the player, and final confrontation.

Figure 3.3 shows a broad plot summary of the chapters throughout both acts. This kind of mapping has three key advantages. Firstly, as Propp's functions are designed for, it provides us with a general overview of the complete plot in terms of the kinds of events that transpire. The granularity at which these functions are applied limits the scope of the plot that can be described. As the restrictions upon functions have been relaxed, the model is now capable of identifying optional narrative components. For example, in chapter 5, the player is able to fail a test, at which point GLaDOS must help the player, resulting in an optional form of *Outsider Help*, which is indicated in the diagram by the dotted line. Similarly, in chapters 8–11, discoverable 'Rattmann Dens' serve as both background narrative and also provide *Outsider Help* to the player by warning them of the true evil nature of GLaDOS. However, while optional or discoverable narrative elements can be represented from a functional point of view, this model is unable to provide descriptions of the content of such narratives. Finally, the relaxed constraints of



(a) The *Coward* and *Out of Map* endings, not sharing any common pathways from the start. The former does not show any Proppian functions, whereas the latter shows at least one, conditionally two.



(b) Three sequential choices where choosing *red* will result in the *Zending* outcome, but repeatedly choosing *blue* will result in one of three endings (*Divine Art*, *Games 1*, *Games 2*) based upon later choices. All four endings share an equal plot prior to this diverging sequence of choices.

Fig. 3.4 Excerpts of a complete Proppian mapping in Appendix A.1 of *The Stanley Parable*, based on the modified Proppian functions and rules of Bostan *et al.* [29].

the function set allows us to identify repeated sequences within the narrative structure. For example, both the first and second chapters of the second act end with *Delivery (5)/Trickery (6)/Confrontation (32)/Escape (43)*, which lets us see that while the content in each may be different, the general plot structure is the same.

While *Portal* can be clearly divided into two acts that always occur in the same order, *The Stanley Parable* introduces a more complex structure that makes explicit act or chapter division uncertain. The player always begins at the same starting point, being able to achieve one of several endings each time they play by making explicit choices as they progress through the narrative. A single pathway from the start for a given ending could be defined as the minimum number of explicit choices and actions that are required to achieve that particular ending. However, many of these pathways have significant overlap, and the player is able to frequently jump between the pathways until an eventual point of divergence, or in some cases the player is able to backtrack to change pathway. Therefore, the narrative experience, even for the same ending, is dependent upon the player's individual choices for a given play session, making it challenging to assign acts or chapters. Figure 3.4 shows two excerpts from a complete Proppian mapping, available in Appendix A.1, of this interweaving structure. In this figure, the green labeled circle represents the player's starting point, red labeled circles denote particular endings,

yellow diamonds and orange triangles together represent explicit choices that branch the narrative, purple labeled circles represent Proppian functions that the player experiences as they make their way through the narrative, and blue labeled circles represent actions the player must perform to progress through the narrative but are unable to be mapped to a Proppian function. Using this kind of mapping, we can not only see the Proppian functions that occur, but also the structural overview of choices within the narrative and identify the similarity between multiple possible paths. For instance, the first image shows two endings — *Out of Map* and *Coward* — which both share no plot points beyond the first starting event, meaning that the pathways are wholly distinct and cannot be jumped between. Looking at the *Coward* ending, we can see that the player must perform a single action to achieve it and that no Proppian functions, beyond the start, are encountered. On the other hand, by achieving the *Out of Map* ending, the player is exposed to the *Reveal* function, and conditionally to the *Trickery* function. The second image demonstrates a sequence further on in the game, where the player is given a repeated choice between entering a blue door or a red door. If at any point they choose the red door, the player is guided along a separate path toward the *Zending* outcome. If instead they repeatedly choose the blue door, they are guided along an alternative path resulting in one of three possible endings (*Divine Art*, *Games 1*, *Games 2*) based upon later choices. From the complete diagram in Appendix A.1, we can see that all four of these endings share an equal plot structure prior to the introduction of this diverging sequence. The complete diagram can be followed according to the player's choices in a given playthrough to determine both the particular story arc and Proppian functions experienced. For example, using the complete diagram in Appendix A.1, if we follow the narrator's instructions at every step, we achieve the *Freedom* ending, experiencing the Proppian functions *Setup*, *Lack (8)*, *Interdiction (2)*, *Guidance (15)*, *Interdiction (2)*, *Reveal (51)*, *Interdiction (2)*, *Reveal (51)*, *Interdiction (2)*, *Wedding (31)*.

These two applications of a variant of Propp's Morphology demonstrate the advantages and disadvantages of the model when applied to video game narratives. The mapping of Proppian functions to video game narratives, particularly with an enhanced function and rule set as used above, gives us a broad understanding of what happens throughout a story, retaining a high-level overview, but not the details of each interaction. If mapping a complex interwoven narrative as in *The Stanley Parable*, then we can also identify similarities between pathways in terms of how much Proppian structure they share. However, the nature of Proppian functions limits the granularity at which narratives can be explained, as they pertain to high-level events. If Propp's functions are applied in their original form with their original rules, then they are unable to represent branching narrative due to temporal and logical constraints. This is remedied by either altering the function and rule set to better suit video game narrative as Bostan *et al.* [29] did or by introducing additional functions specifically to handle decisions, as proposed but not explored by Brusentsev *et al.* [34]. Despite these enhancements, Propp's Morphol-

ogy struggled with stochastic variability, such as being unable to randomly select from several dialogue lines which could potentially change the Proppian function mapping. Additionally, while Propp's character archetypes are suitable for the vast majority of characters in storytelling, they do fail to capture characters who transition between archetypes, as described above for both games.

3.2.3 Applying Aarseth's Narrative Theory of Games

Aarseth's conceptual model of narrative theory for games [3] describes a series of dimensions — world (ludicity, structure), objects, agents, and events — that when applied to a game can determine the type of narrative experience the game provides. It is combined with what Aarseth calls a 'variable model' allowing us to plot how the dimensions of different games relate to one another.

Applying the *world ludicity* dimension required analysis of the play space of the games, with both containing a majority of ludic (playable) spaces alongside extra-ludic (not playable) areas that the player is able to see but not reach. Complementing this is the *world structure* dimension, for which *Portal* was simply linear, but *The Stanley Parable*, with its interwoven narrative, is more difficult to assign, being closest to a multicursal structure in Aarseth's terms. Aarseth argues that these two dimensions, together describing the game world, have impact upon the ease by which a particular story can be told. While this is true, the granularity at which they describe the game in a narrative analysis is questionable, which Aarseth himself raises.

Objects within the games were categorized based on the predefined levels of malleability. For the most part, objects fit well into these categories and they describe the interactivity with the player. However, some objects straddle between definitions or do not quite fit, such as the Companion Cube in *Portal*, which by the malleability enumeration would be best suited as a *static usable* object, but in reality the object is dynamic. Similar examples can be found in *The Stanley Parable*, such as the telephone plug encountered in the *Not Stanley* ending, which best fits the *static usable* categorization, but once again is actually dynamic. It is possible that these complications arise due to misnomers; rephrasing and expanding upon the enumeration to include a greater range of objects would likely resolve this.

Characters were similarly defined based on malleability, with some terms borrowed from Forster's own character classification methods [70]. Characters were able to be assigned accurately, with a few exceptions. For example, the Personality Cores from *Portal* each have a personality and unique names, which based on how Aarseth classifies them would make them *deep*. However, considering the definitions inherited from Forster, they would also have to change and develop as the story progresses, which they do not, making them either *shallow* or not able to be assigned accurately.

Events for each game are collectively generalized under one of four enumerations based on the balance of kernel (critical) events and satellite (not critical) events found

within the narrative. Aarseth translates this definition to relate to the level of agency in the game. The coarseness that this approach takes limits its usefulness to describe the overall narrative. However, the labeling of event balance at lower levels of detail, such as per chapter in *Portal*, would be able to provide an overview of how the level of agency changes throughout the narrative as the game progresses.

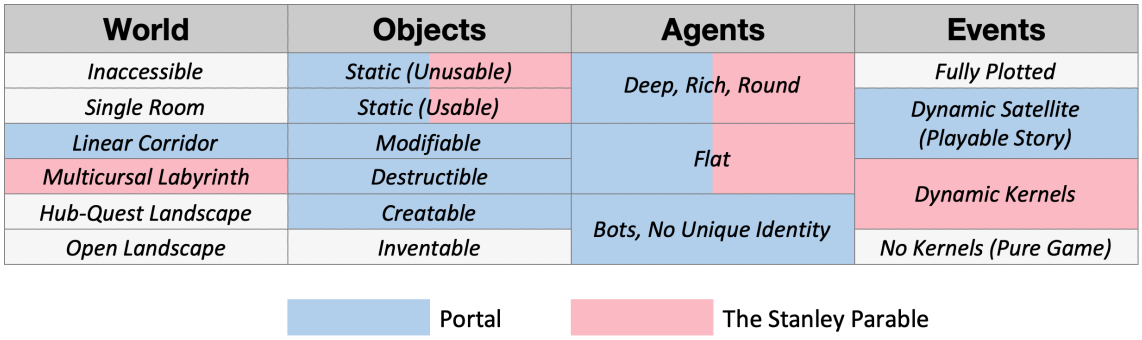


Fig. 3.5 Aarseth's variable model with colored cells for *Portal* and *The Stanley Parable*.

For each game, the variable model was then built by combining the other dimensions together. Entering multiple games on the same graph allows for comparison of the components within each game. Figure 3.5 shows a visually improved configuration of Aarseth's variable model populated for both *Portal* and *The Stanley Parable*. In the original variable model, cells were not shaded but instead had a single horizontal line crossing those which a game makes use of. The practicality of this is questionable as it does not allow for multiple cells to be included per game, especially those that are not neighboring, which is something that Aarseth rightly claims to be commonplace.

3.2.4 Applying CANVAS

CANVAS is a data and runtime framework for the authoring of key plot points with automatic synthesis of narrative between those key points [108]. Its underlying narrative model is built with formal logic and has been implemented into an authoring tool embedded within the *Unity* game engine [171]. Individual entities within the game world are represented as smart objects, each consisting of a state and a set of advertised affordances. A single affordance is an advertised capability offered by a smart object that manipulates the states of the owning smart object and typically other related smart objects. For example, a chair advertising a `Sit` affordance may impact the state of the chair itself and the smart object sitting down. To use a programming analogy, this can be thought of as a member function of the smart object that can modify itself and any input smart object parameters. States are defined as a set of per-object attributes combined with a set of one-way pairwise relationships between smart objects. Given a smart object representing a door, a mutable attribute could be an `IsLocked` value determining whether or not the door is locked. Given a pair of smart objects `A` and `B`, a one-way pairwise relationship could be that `A` has an `IsFriendOf` entry paired with `B`, but the reverse may

Table 3.3 *CANVAS* affordances present in the *Portal* cube-button-door example.

Affordance Name	**State Transformation**
`PickUp`	$\text{carry}(\omega_o, \omega_u)$
`NavigateTo`	$\text{pathTo}(\omega_o, \omega_u)$
`Drop`	$\text{drop}(\omega_o)$
`TogglePressed`	$\text{pressed}(\omega_o) = \neg\text{pressed}(\omega_o)$
`ToggleOpen`	$\text{open}(\omega_o) = \neg\text{open}(\omega_o)$

not necessarily be true. Events are predetermined context-specific interactions between any number of smart objects, where the runtime outcome is dictated by the states of the participating smart objects. Which smart objects are able to participate in a given event is determined by a role filter, which determines whether or not a particular smart object is able to fulfill a desired role in an event. This prevents, for example, door smart objects from being used where a character smart object is expected. The execution of an event is also guarded by a precondition based upon the state of the chosen smart objects which ultimately determines whether or not the event can fire in runtime. If an event does successfully execute, then the actual logical behavior of an event, which is essentially a parameterized behavior tree conditionally invoking affordances, is triggered, which in turn potentially alters the state of all contained smart objects. An individual narrative arc is finally defined as a sequence of authored events that transform the state of the world gradually to the desired goal state at the end of the event sequence.

Due to the detail of this narrative model (individual objects, all affordances with all other objects, world states, individual events, and so on), only early parts of the games were included in the application and the complete narrative was not considered. Both games were broken down into unique roles that objects could take on, all possible states required for interactions with objects, reusable affordances between arbitrary objects, events for interactions (including preconditions and state transformation), the objects themselves, and then broad narrative arcs based on sequences of events.

The narrative model is capable of representing static narratives with great detail about the game world, its states, constituents, and the way in which all relate. However, the model is unable to handle player agency and freedom in that the sequences it describes are predetermined. Additionally, without the use of an authoring tool, the fulfillment of the model's components is a greatly laborious task. To demonstrate this, we can observe an excerpt of the first level of *Portal* where the player must find a cube, pick it up, find a button, and finally drop the cube on the button, which in turn depresses the button and triggers a door to open, allowing them to continue. Let's assume that roles have been defined for the *Player*, *Cube*, *Button*, and *Door*, and that states *Pressed* and *Open* are defined for use by the button and door. Table 3.3 lists the affordances, and as with the authors' implementation into *Unity*, defers the implementation of *State Transformation* to a parameterized behavior tree. This deferral has the benefit of generalizing the model

Table 3.4 *CANVAS* events present in the *Portal* cube-button-door example.

Event Name	Description
MoveToCube(Player p, Cube c)	Player navigates to a given cube.
PickupCube(Player p, Cube c)	Player picks up a given cube.
MoveToButton(Player p, Button b)	Player moved to a given button.
Drop(Player p)	Player drops current item.
PassDoor(Player p, Door d)	Player walks through a given door.

to not rely upon predetermined functionality and allows for arbitrary extension of its capabilities into any runtime platform.

With roles, states, and affordances laid out, smart objects can be defined, which in the case of this example would be a one-to-one mapping with the roles, but typically the roles would be shared and reused as tags. The player object has the affordances `PickUp`, `NavigateTo`, and `Drop`, the cube has no affordances, the button has the affordance `TogglePressed`, and the door has the affordance `ToggleOpen`. In the actual analysis, these listings were much more extensible both in the number of unique objects and their advertised affordances. Table 3.4 demonstrates events whose behavior are again deferred using parameterized behavior trees.

These functions have simple restrictions on their parameters by role (for example, `MoveToButton` can take any object of role *Button* as its second argument rather than a specific one) but usually include more advanced preconditions. With events in place, it's now possible to order them to create a narrative arc, in this case of the player completing a simplified challenge from the start of the game. However, if we read them in the order they occur — `MoveToCube`, `PickupCube`, `MoveToButton`, `Drop`, `PassDoor` — there is one crucial component missing, which is how objects dynamically respond to changes in the game state. Once the player drops the cube on the button, it should toggle its *Pressed* state and in turn the door should update its *Open* state so that the following `PassDoor` event can run. This highlights a limitation of this methodology and the static nature of the model. While this behavior could easily be emulated by deferring everything into a parameterized behavior tree, the model is unable to respond to dynamically changing world states, requiring conditional reactions to be predetermined.

Given the high detail of this model, and its intention for use within an authoring framework of largely static and predefined narratives, it is not surprising that it is unable to support responsive events. However, its role-based assignment of objects and deferral of functionality of affordances and events has great promise for helping to better generalize narrative models due to the agnostic approach taken which means that models are able to rely less on object-specific and genre-specific properties that may limit their scope.

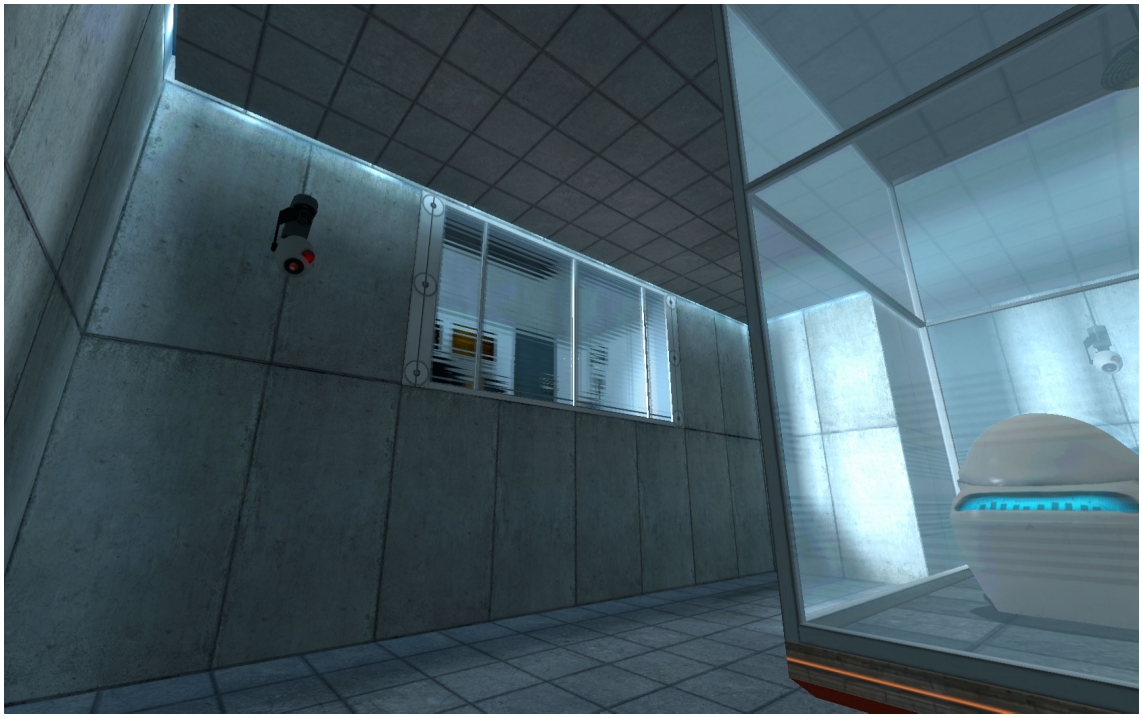
3.2.5 Applying Bernstein's Patterns of Hypertext

Bernstein analyzed a variety of hypertextual media and identified common structural patterns that occur within the texts. These patterns provide a conceptual vocabulary through which we can describe, analyze, and even design complex hypertexts [19]. While not necessarily particular to game narrative analysis, these patterns are hypertextual and as such can be also used to identify occurrences within video game narrative structures, as both are mapping an interactive medium in terms of its content, the relationships between content, and its interactions.

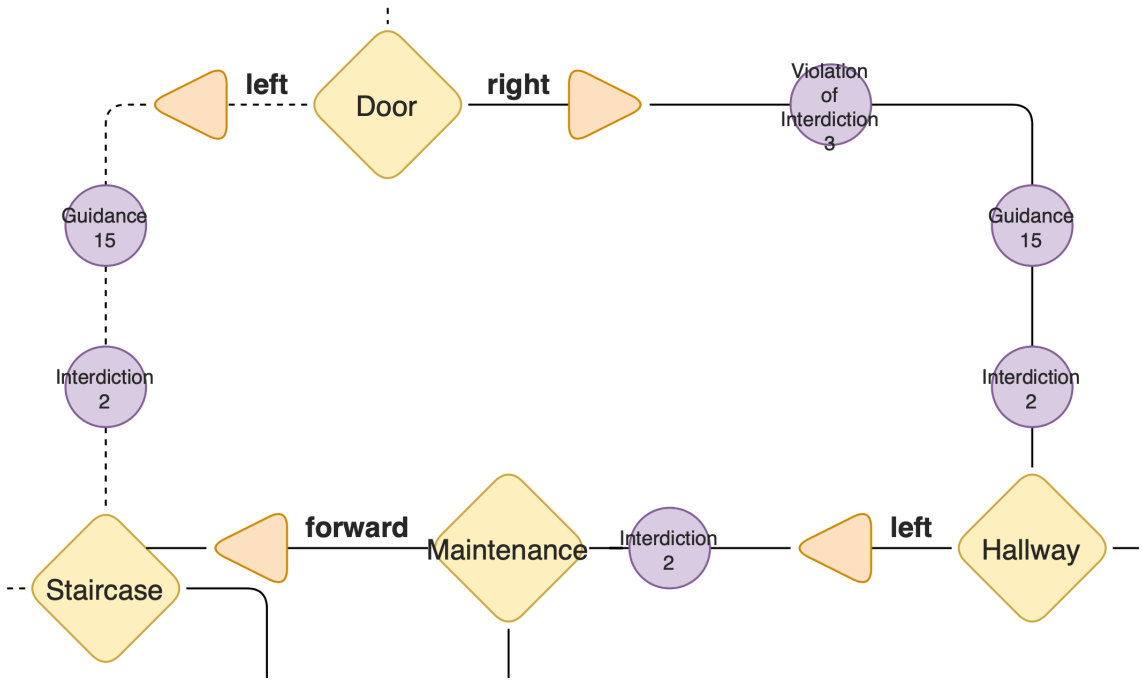
As the games were played through, various structural patterns were observed from the point of view of the player, much alike Bernstein describes readers of hypertext experiencing the patterns as they interact with the texts. As expected based on the linear structure of *Portal*, it had only limited occurrences of structural patterns — Cycles, Joyce's Cycles, Montages, and Missing Links. Conversely, *The Stanley Parable* demonstrated a wide variety of patterns — Cycles, Joyce's Cycles, Contours, Mirror Worlds, Tangles, Sieves, Split/Joins, Rashomon Split/Joins, Overview/Tour Split/Joins, Missing Links, and Navigational Feints — which is not surprising given its structural complexity.

The Missing Link pattern refers to hypertext structures that hint at the presence of a link that does not actually exist for the reader to follow. Bernstein notes that, in text, techniques such as allusion and ellipsis can suggest a Missing Link that cannot be accessed. Moreover, he explains that the introduction of structural irregularity in a context of otherwise regular structure creates an especially powerful Missing Link, representing a place that is referenced but cannot be navigated to, with its inaccessibility making it uniquely attractive. In video game narrative, the Missing Link pattern is commonly used to hide non-ludic spaces yet make them appear as ludic spaces to the player, which gives the illusion that the active play area is greater than its actual scale. In *Portal*, for example, a Missing Link is created with the use of observation offices placed within test chambers that are out of reach of the player, as shown in Figure 3.6a. The presence of these spaces suggests to the player that a world exists beyond the space in which they inhabit, but in reality these spaces are empty. One particularly interesting implementation of this is found in *The Stanley Parable* — the player can enter an elevator that closes, shakes, and plays ambient music, but never reaches its destination. This Missing Link tricks the player into not only thinking that there is accessible space, but that they are actively going toward it, when in reality they are stationary, which is revealed once they choose to exit the elevator and notice that they did not travel anywhere.

Another pattern that is commonly found in choice-based games is Split/Join, where the narrative diverges and then rejoins regardless of the path taken. *Portal*, being linear and without choice, does not have any explicit Split/Join occurrences, at least at the level at which the grand narrative arc is experienced. On the other hand, *The Stanley Parable* thrives on explicit player choice and consequently has a high number of Split/Join occurrences. Figure 3.6b shows an excerpt from the Proppian analysis that demonstrates



(a) An example of Bernstein's Missing Link in *Portal*. The player, in the ludic space, can see non-ludic areas through observations windows which suggests a more expansive world, but the player is unable to gain access to the locations.



(b) An excerpt of *The Stanley Parable* Proppian mapping from Appendix A.1 showing a Split/Join. When choosing *left Door*, the player ends up at the *Staircase* choice, but can also reach the *Staircase* if choosing *right Door*, *left Hallway*, and *forward Maintenance*.

Fig. 3.6 Bernstein's patterns appearing in *Portal* (a) and *The Stanley Parable* (b).

a Split/Join. The player starts with the *Door* choice where they can choose to enter a door on the left or a door on the right. If the player chooses the door on the left, they shortly end up at the *Staircase* choice. However, if the player chooses the door on the right and then makes a subsequent set of particular choices (*left* at the *Hallway* choice, then *forward* at the *Maintenance* choice), they can still end up at the *Staircase* choice. In this way, the player is forced to diverge along one of two paths, but has the possibility to rejoin to a shared choice point.

In hypertext patterns, a *Cycle*, which has several variations, broadly refers to iterative repetition of the same content, such as repeatedly visiting the same node in a hypertext fiction until conditionally exited by relying on iteration count, user interaction, or other properties. We can also see frequent use of this technique in video game narrative. For example, in the first act of *Portal*, the player will encounter *Test Chamber 9* as they are starting out, at which point they only have access to one of the two portal colors and a single box, challenged with solving the puzzle as designed accordingly. However, during the second act when the player is escaping, they encounter this test chamber once more, but this time in a destroyed state and without a box, entering through a tunnel rather than the standard entrance as before, and this time with access to both portal guns, having to solve the puzzle in a different way than before. Moreover, in both versions, the player exits the test chamber differently — in the first instance, they use a standard elevator at the end of the level, whereas in the second, they jump down a hole where the elevator should be. All of these changes between the same test chamber create a cycle, which serves as a narrative device to reinforce the sense of being on the run during the escape sequence. We can also see several cycles being used in *The Stanley Parable*, such as the looping colored door choices as shown earlier in Figure 3.4b. Here, the player is presented with an initial choice of entering either a blue or red door, with the narrator motivating them to enter the red door. Choosing the blue door each time will cause the exact same choice to be repeated, albeit with a slightly modified environment in an attempt to convince the player to choose the red door. If the player repeatedly chooses the blue door, the cycle will end after three choices along a specific pathway toward one of three endings based on later actions. However, if the player at any point chooses the red door, the cycle will exit early along a different pathway toward a specific ending.

The strength of these hypertextual structural patterns lies in the vocabulary that they provide for the analysis of existing works. In the context of video game narrative, they are able to identify structural trends of the narrative from the point of view of the player but do so at a granularity that does not involve the content nor the events that make up the actual structures. The patterns, much alike Propp's functions, can therefore serve as a high-level template for understanding and even designing the structural content of a game's narrative by considering the functionality and purpose of the patterns and the impact they have on the player's experience.

3.3 Discoverable Narrative

In the previous section, we have seen an example of how mapping a game narrative with Proppian functions can identify optional narrative content such as the discoverable 'Rattmann Dens' from *Portal*, but this does so without acknowledging the type of content found within the narrative experience. It is also possible to use the logic of *CANVAS* or Bernstein's patterns (*Split/Join*, *Montage*) to structurally represent such optional content. These models, while able to acknowledge the presence of optional and discoverable content, do not differentiate it from explicit branches (e.g., optionally seeing content would structurally appear equal to a choice). Moreover, they do not describe the narrative delivery mechanism through which the optional content is experienced, such as through in-game graffiti on a wall, metadata in an in-game database, or discoverable items that enhance the game's narrative. Additionally, while the *Indexical Storytelling* model discussed in §2.3 does contain indices that could represent discoverable content using the idea of *interpretation of remains* through authored indices left for the player to find, it also does this without acknowledging the type of content or the methods in which the content is discoverable.

To resolve this, I propose an addition to future game narrative models in the form of *Discoverable Narrative* for representing elements of video game narrative that are discovered, observed, or experienced. These narrative elements are often not mandatory, serving to increase narrative depth. In this context, 'discoverable' means that the narrative element is not explicitly provided to the player and that they must seek the items (either actively or by chance). This often comes in the form of books, letters, and other collectibles that are spread out throughout the game world for the curious player to find. 'Observed' refers to narrative elements that are consumed through the passive act of observation rather than being consciously interacted with. This sometimes appears in the form of environmental storytelling in that certain elements, such as text printed on walls, can tell a story, or at least suggest a story to the player without the need for direct interaction. 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 something (whether in-game or otherwise) can mean that the player experiences narrative inherently by interacting with the game's systems. As a result, this descriptive concept can broadly categorize these types of narrative elements that players encounter.

The representation of such elements is defined as an amalgam of the four dimensions displayed in Table 3.5. 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

Table 3.5 *Discoverable Narrative's* four-dimensional descriptor consisting of *Tangibility*, *Functionality*, *Clarity*, and *Delivery*.

Tangibility	Tangible Attached to an in-game object.	Intangible **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.	Passive Is observed or experienced.

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 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.

Although forms of environmental storytelling can be described through *Discoverable Narrative*, their purposes and functionalities differ. Environmental storytelling is a fuzzy term for a variety of narrative techniques that use the physical space of a video game world to portray narrative experiences as discussed in §2.3. On the other hand, *Discoverable Narrative* provides a vocabulary for describing the metadata of such narrative techniques and through this enables classification and comparison. Moreover, *Discoverable Narrative* is able to capture narrative elements beyond environmental storytelling. In the examples that follow, the first and second entries are forms of environmental storytelling, relating to the use of the environment and objects within it to portray narrative. The third and fourth examples both describe concepts outside of environmental storytelling, the former referring to in-game codex entries for supplementary narrative lore, and the latter using the game's mechanics as a narrative metaphor for the player character's mental state.

3.3.1 Examples

Application of *Discoverable Narrative* to an existing (or planned) narrative experience is done by evaluating the four dimensions and selecting the most appropriate option for each. Two (or more) applications of *Discoverable Narrative* can be directly compared to determine how the narrative experiences differ. This is done by comparing the assigned enumeration value for each of the four dimensions with the resulting differences becoming

apparent as mismatches are encountered. If evaluating a larger number of narrative experiences, then *Discoverable Narrative* can be used as a way of categorizing them based on the selected enumeration values from within the four dimensions. For example, a collection of narrative experiences could be queried, much alike using SQL, to retrieve only those that have *Tangibility* equal to *Tangible*, or even combinations of the enumerations, resulting in a subset of the items that can be further investigated. Following is a set of demonstrations of how *Discoverable Narrative* can be applied to existing narrative experiences found within video games.

Active Tangible Objects

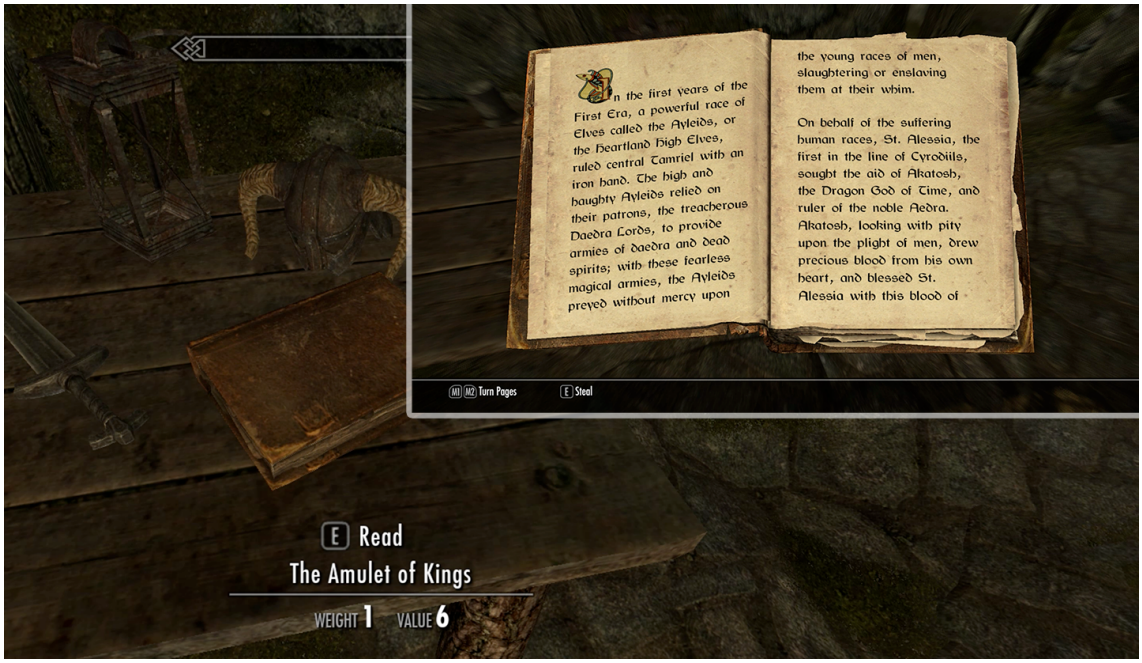


Fig. 3.7 The book titled 'The Amulet of Kings' discovered in a dungeon in *The Elder Scrolls V: Skyrim*, providing additional lore about the game's world and its history.

In *The Elder Scrolls V: Skyrim*, there exists 820 variations on books, all scatted around the vast game open world for players to find, with 307 of these books serving no other purpose than for narrative filler[43]. Within the books is usually a mixture of text and images that the player can read and interact with. An example of such a book is shown in Figure 3.7. As there is a physical in-game object associated with the narrative text, they can all be defined as *Tangible*. Functionally, the 307 books that exist solely for narrative purposes would be considered as *Narrative*, and the remaining books would be declared as *Mechanical*, as their text is secondary to their primary action (such as presenting the player with a quest, giving the player a new spell, acting as a quest item, and so on). The contents of the books regardless of Functionality are *Explicit* as the text and images are well defined. To be consumed, the player must actively discover the books and consciously interact with them in order to experience the narrative texts, therefore can all be considered *Active*. If we were to apply this to each book individually, we would have a

resulting library of books that can be separated based on, in this case, their Functionality, with all other enumeration values being shared. If this were further expanded to other items within the game, they too could be categorized and compared.

Passive Tangible Objects

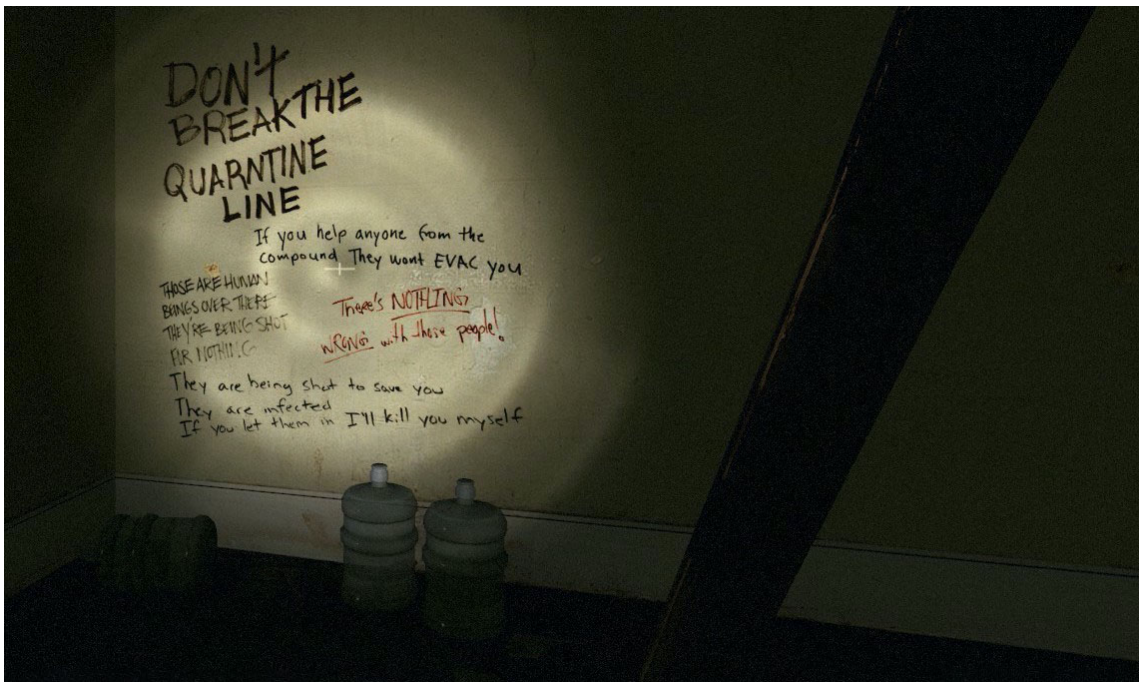


Fig. 3.8 Survivor graffiti found in the 'Quarter' safe room of 'The Parish' campaign in *Left 4 Dead 2* revealing stark warnings of danger to those who discover it.

In *Left 4 Dead 2*, there are a number of safe rooms spread throughout the campaigns that break up the game's pacing and act as no-harm zone checkpoints for players. The environments of these safe rooms are often used to present an ongoing narrative to the player based on the context of the story being told in the particular campaign. One such example of this is leftover graffiti from previous occupants of the safe rooms as shown in Figure 3.8, in this case having left stark warnings to others in regard to quarantine and the impacts of interacting with the infected. The graffiti found in these safe rooms can be considered *Tangible* as there is a physical in-game object associated with the narrative text. Functionally, they serve a *Narrative* purpose, as they play no other role than supplementary lore or narrative hints. The content of this graffiti is *Explicit* as there is a physical in-game representation of them. The narrative text occurs regardless of conscious direct player interaction and is instead consumed through observation as the player discovers the element, so they are *Passive*.

Active Intangible Objects

In *Mass Effect 2*, the vast galaxy that the player inhabits is populated by various planetary systems, with each being the potential home of one of the many in-game character races.

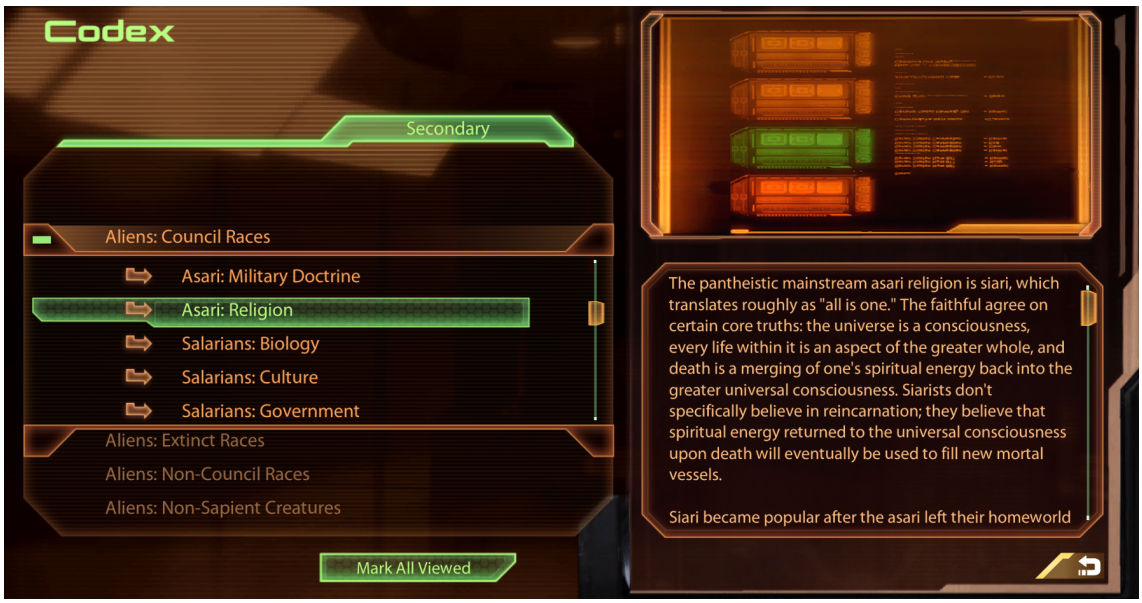


Fig. 3.9 A codex entry from *Mass Effect 2* providing supplementary lore about the religion of the Asari race within the game's universe.

The game features a codex that provides detailed information on the various aspects of the game world, including supplementary lore about each of the character races, contributing to the creation of a more believable and coherent experience. Figure 3.9 shows a sample entry from the codex which outlines the religious beliefs of the Asari race. These kinds of texts are *Intangible* as they are not associated with a particular physical in-game object but instead reside as part of an encyclopedia of sorts. Functionally, they are of a *Narrative* purpose as they serve only to enrich the lore and provide no mechanical benefit to the game. Their content is *Explicit* and clearly defined, as there is readable text associated with each entry. Codex entries are typically gated from appearing in the codex until some in-game condition is met (such as encountering a race triggering the unlock of all codex entries about that race). Once available, codex entries are consumed by the player by consciously finding and selecting the entry, therefore making them *Active*.

Passive Intangible Objects

In *Spec Ops: The Line*, the player's avatar is gradually becoming disillusioned and experiencing an early onset of post-traumatic stress disorder. In one encounter, the player's avatar is engaged in a heated battle with a heavily armored and armed enemy. During this sequence, the screen repeatedly flickers black and attacking the enemy results in him turning into a mannequin for him to shortly appear elsewhere back in human form. This is a demonstration of using the game's mechanics as a metaphor for the gradual degradation of the character's mental state. Although tangible objects make up this experience, there is no particular object that it is related to (i.e., it occurs in the game world and is in part delivered by objects, but it is not inherently tied to an object), therefore it is *Intangible*. The functionality of this kind of narrative element is difficult to assign as it

is often based upon the developer's intentions. It can be considered *Mechanical* due to its nature as an in-game mechanic and use of game mechanics but can also be considered *Narrative* due to its impact on the player and the story. This divide is a potential limitation of the model's descriptors that must be resolved on a case-by-case basis. In definition, it is *Implicit*, as there is no concrete instance of the element's content, such as text. It is also experienced by the player rather consciously chosen to be interacted with, and the resulting effect is interpretative, making it *Passive*.

3.4 Novella Narrative Model

In §3.2, we have explored existing modeling approaches and have subsequently applied a variety of these existing models to two video games of differing narrative complexity to better understand their affordances and constraints in the context of video game narrative. This section presents a new game-centric narrative model that not only builds upon these affordances and constraints, but also considers my concept of *Discoverable Narrative*.

This narrative model provides the underlying structural foundations upon which the rest of the architecture is built, influenced by the findings from the application of existing models to video game narratives. It is designed to be completely agnostic of genre, as to not include type-specific functionality, and has a philosophy of being readily implementable rather than simply representing or describing elements of video game narrative. The creation of this model also directly answers the first research question and aids in answering the third research question. The first research question seeks to not only understand the capabilities of existing models for capturing video game narrative complexities — which was previously explored in the application and discussion earlier in this chapter — but also to use these findings to create a new model that may better support this. By building this narrative model, the latter part of the first research question is directly addressed. Moreover, this model fulfills a major component of the game narrative authorship architecture outlined at the beginning of this chapter, which serves in part as the foundation for the design of a new authoring tool in §5 which in turn is used within a study in §6, targeting the third research question.

The philosophy of this model is to remain practical rather than only descriptive; it is designed with implementation and integration into game engines in mind, such as *Unity*, *Unreal Engine 4*, or other custom runtimes. The core structure of the model has three layers, each serving a slightly different purpose. The first layer consists of *Groups* acting as high-level asynchronous containers for the game's structure. The second layer has *Sequences* that likewise work to segment off individual collections of narrative material but can act both synchronously and asynchronously. The third layer consists of *Events* that represent the individual narrative actions that occur in games. Additionally, within the second and third layers there are *Hubs* and *Returns* which allow for more advanced structural configurations. The layers and functionality, while simple on their own, create

great complexity when combined whilst remaining easy to understand. While the core structure is provided by these layers, the actual in-game functionality of each element is deferred to a particular implementation of the model with the use of its extensible scripting language, *Logic*, which is discussed in more detail below. In this context, an 'implementation' refers to a realization of the framework in a programming language that can be used within a runtime. Figure 3.10 shows a high-level UML diagram of the main components of the model.

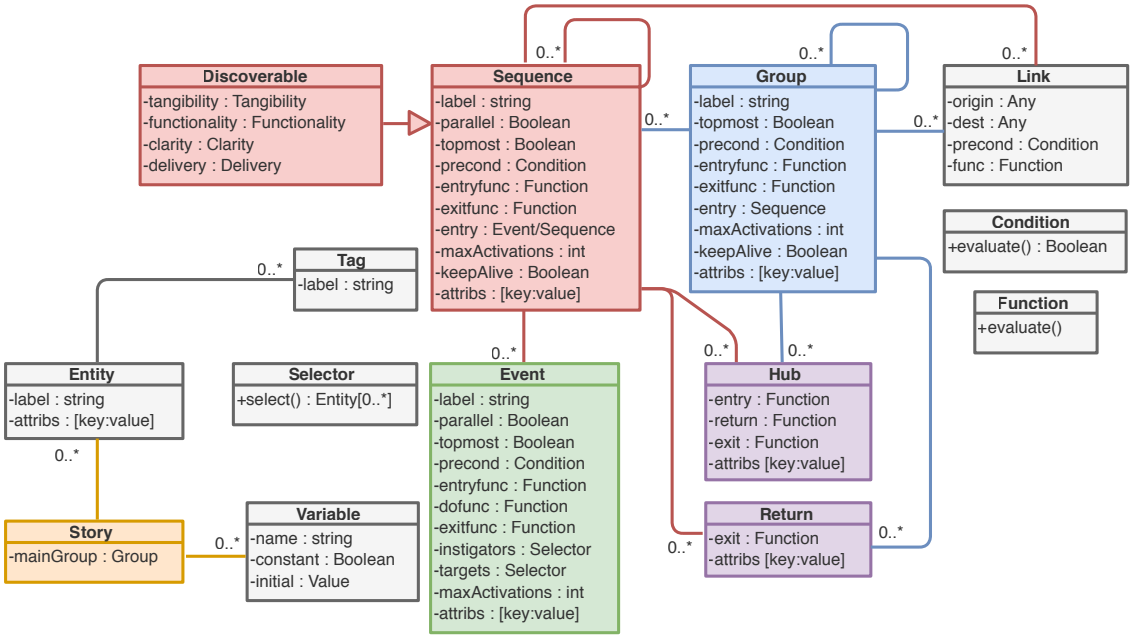


Fig. 3.10 A high-level UML diagram for the *Novella Narrative Model*.

As a high-level example of the model's practical use, let us briefly consider an implementation of the model in context of the architecture described in §3.1 applied to a hypothetical quest-based role-playing game. The *Narrative Model* component of the architecture as shown in Figure 3.1 is fulfilled by an implementation of this model in any suitable programming language, and the *API* component can be similarly implemented by exposing the model to the game engine both in terms of handling the model's data and serving as a bridge for *Logic* to bidirectionally communicate between the model and the game engine. Assuming an appropriate authoring tool based on the model has been created, an author could represent a single questline with a *Group*. Within that *Group*, several *Sequences* could be created, each representing different stages of the quest. Within each *Sequence*, a network of *Events* would ultimately handle things like dialogue, in-game events, cutscenes, or other triggers. To make this kind of structure practical for use, several advanced features of the layers, which are described in detail below, would be utilized including appropriate conditional triggering of each *Sequence* and *Event* with real-time in-game queries, as well as deferral of *Event* functionality through *Logic*. If the quest featured cyclical structures at either the *Sequence* or *Event* level such as those found in dialogue trees, then the *Hubs* and *Returns* could be used accordingly.

The model is chiefly a data model as at its core it defines individual low-level structures for narrative elements and the relationships between them. However, it also strongly integrates runtime features with its use of *Variables* and *Logic* for bidirectional communication with an engine as described below. It also has aspects of a conceptual model with the *Tag* system for categorizing *Entities* sharing a common role as also covered below.

3.4.1 Story

The *Story* is a custodian of sorts, responsible for the creation and management of all narrative elements (that is, the other classes). All *Variables* are globally accessible and are likewise managed by the *Story*. There always exists a single top-level *Group* within which all other narrative elements exist, referred to as the 'main group'. At any point, the *Story* can be used to execute a set of *Logic* code. This is of particular use when runtimes wish to query the state of the *Story*, or even forcefully modify the *Story*, such as when triggering an in-model element from an external source.

3.4.2 Variables

The state of the story from the model's perspective (i.e., what the model knows about the story, internal to the model, separate from the main game code) is controlled by a set of global *Variables*. A *Variable* is a type-restricted piece of data, such as a Boolean, a floating-point number, or an integer, which can store arbitrary information. All *Variables* are mutable by default but can optionally be declared constant. *Variables* have an initial starting value that is restored when *Story* execution begins. Since *Variables* are global, their identifiers must be unique to avoid ambiguity.

3.4.3 Groups

Groups are the highest level container. Within a *Story* there is only one main *Group* and every other *Group* within must be nested. *Groups* can be nested indefinitely for structural or organizational purposes. *Groups* act as a scope for their contents; elements inside are only able to act while their parent *Group* is active. All *Groups* nested 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 siblings would execute in parallel. However, *Groups* do not necessarily execute immediately as they are guarded by a *Condition* that determines when the *Group* will trigger. This means that while all *Groups* are parallel to their siblings, their *Conditions* may result in different execution timing. *Groups* therefore may execute instantly, with delay, or not at all. *Groups* contain a list of *Sequences*, *Hubs*, and *Returns*. The connectedness of these constituents is determined by a list of *Links*. Each *Group* has an entry point that is either empty or one of the contained non-parallel *Sequences*. When entered during simulation, a *Group* will automatically trigger its entry point if set; if the entry point is not set then no action is taken (this could occur if, for instance,

all contained children are parallel). When a *Group* is entered and exited, a *Function* is optionally run that can modify the *Story* state. *Groups* can be marked as topmost which means that when entered, everything in the model pauses execution until the *Group* exits, at which point paused elements automatically resume.

3.4.4 Sequences

Sequences are individual segments of narrative made up of nested *Sequences*, *Events*, *Hubs*, and *Returns*. *Sequences* are sequential as defined by their link structure in their parent but can be optionally marked as parallel. A parallel *Sequence* will attempt to trigger when its parent is active (first at entry and then every tick) with its activation being determined by its *Condition* being met. As with *Groups*, this means that they may instantly execute upon entry, with delay, or not at all. Within a *Sequence* is a list of contained *Sequences*, *Events*, *Hubs*, and *Returns* as well as a list of *Links* determining their connectedness. There is an entry point that is either empty or one of the contained non-parallel *Events* or *Sequences*. As with *Groups*, when a *Sequence* is entered and exited, a *Function* optionally runs. *Sequences* can also be marked as topmost.

3.4.5 Discoverables

Discoverables are a specialization of *Sequence* that partially implements *Discoverable Narrative* by appending its four dimensions as enumerations and is always parallel. Due to the hierarchical scoping of this model, the parent *Group* determines when the *Discoverable* can be found. The added enumerations act as labels and have no direct influence upon their contents. Instead, contained elements can use *Logic* to query the state of these enumerations and respond accordingly. This means that *Discoverables* are essentially a parallel *Sequence* with extra information that is triggered by a given *Condition* within a given scope. Because of this, more abstract kinds of *Discoverable Narrative* such as mechanics as metaphor are not easily represented. This is intentional as to not overcomplicate the model; attempting to wrangle every form of *Discoverable Narrative* into this model would increase its complexity without substantial return.

3.4.6 Events

An *Event* is a representation of a single narrative event. Where *Groups* and *Sequences* are containers, *Events* are leaves of such a tree and cannot be divided further. Sibling connectedness is defined by a set of *Links* in the parent. Similar to previously, Events can also be declared as parallel and topmost. *Functions* also fire on entry and exit as before, but *Events* are unique in that they also have an additional `do` *Function*. This *Function* is responsible for the actual behavior of the *Event* and is therefore delegated to the actual implementation via *Logic*. This is the primary manner in which the model defers *Event* specifics to a runtime. *Events* also contain two *Selectors*. The first determines the set of

Entities that are instigating the *Event*, and the second determines the set of *Entities* that are targets of or are otherwise involved in the *Event*. The results of these *Selectors* can be queried and used by the *Event's Functions*. *Events* do not require the use of *Selectors*, but they instead serve as metadata about the involved *Entities* for use in *Functions* both inside and outside of the *Event*.

3.4.7 Hubs & Returns

Hubs and *Returns* are located within *Groups* (for use with *Sequences*) and within *Sequences* (for use with *Events*). They function as additional structural assistants that provide an easy method of creating cyclical networks and introduce several features that would be tedious to implement, if at all possible, with only *Links* and *Variables*. Both *Hubs* and *Returns*, just like the other components, can be connected to and from with their parent's *Links*. A common use case of this structure in video game narrative is for cyclical dialogue patterns. In these scenarios, several topics are offered to the player with some of those topics eventually returning back to the same offering of original topics and others instead continuing the story and breaking the cycle. It is also possible for these cyclical networks to be either sequential or even embedded within each other. While it is possible to emulate this kind of network without their inclusion, some additional functionality is provided when utilizing them. A *Return* has no outputs but does have an optional property defining the unique label of the *Hub* that it relates to which must occur prior to the *Return*. When a *Return* is encountered, it jumps to its labeled *Hub*, and if one is not provided, it instead automatically jumps to the last encountered *Hub*, in both cases triggering an exit *Function* upon doing so. A *Return* being present before a *Hub* is therefore in violation of this rule, but it is okay to have any number of *Returns* following at least one instance of a *Hub*. *Returns* are also equipped with a *Condition* determining whether or not they can be executed. *Hubs* have the standard entry and exit *Functions* but also have a `return` *Function* that triggers whenever a *Return* node results in the *Hub* being activated again. This differs from the entry *Function* in that it provides an additional level of control for these cyclical structures. Moreover, using the *Logic* `activations` function (see Table 3.6), the number of times a *Hub* or *Return* has been triggered can be queried which can be used to dynamically alter content within the cycle each iteration, for example.

Figure 3.11 shows a simple example of *Hubs* and *Returns* with *Events* being represented as blue nodes. Here, *Hub A* presents the player with *Event* options *A*, *B*, and *C*, where choosing either *A* or *B* will always trigger a return to *Hub A*. Only once the player chooses *Event C* does the narrative continue, in this case to *Hub B*, which in turn presents its own three *Event* options *D*, *E*, and *F*. Choosing *Event D* will return the player to *Hub B*, but choosing *Event E* will return the player all the way back to *Hub A*. Only once the player chooses *Event F* will the cycle of *Hubs* and *Returns* end. While not used in this diagram, the `activations` function could be used to introduce variety as the player

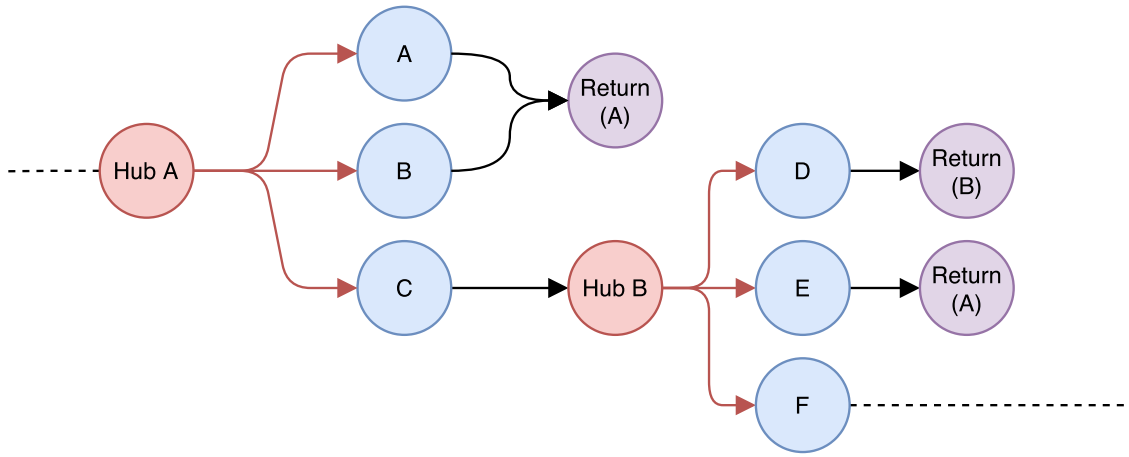


Fig. 3.11 An example of sequential *Hubs* and *Returns* with blue nodes as *Events*. The first *Hub* cycles until *Event* `C` is chosen, at which point a second *Hub* begins, with *Events* `D` and `E` eventually returning to *Hubs* `B` and `A` respectively. Choosing *Event* `F` will exit the cycles. The `activations` function could be used for per-iteration variety in all nodes.

cycles through both *Hubs*. For example, *Event B* could have its *Condition* check whether *Hub A* has activated at least twice and *Hub B* has activated at least once. This way, for *Event B* to become available as a choice, the player must enter *Hub A* and choose *Event A*, return to *Hub A* and choose *Event C*, enter *Hub B* and choose *Event E*, finally returning to *Hub A* at which point *Event B* becomes available as its *Condition* is satisfied.

3.4.8 Attributes

Groups, *Sequences*, *Events*, *Hubs*, and *Returns* all have an associated dictionary that maps a string-based key to a value of any type. This system, referred to as *Attributes*, serves no structural purpose within the model, but rather provides metadata that can be passed from the model to the runtime through the API. Using this *Attribute* system, it is possible to represent features such as the location within which an element 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 subsections as *Sequences*. Locations can then be represented by an *Attribute* with the key `location` and value of the location name. The *Logic* or runtime can then query this *Attribute* and respond accordingly. Earlier it was acknowledged that some forms of *Discoverable Narrative* were not plausible in this model. However, using *Attributes*, we can emulate simplified forms of environmental storytelling. An *Attribute* could be created, such as `environment`, that stores a textual string written by designers instructing the runtime on how elements should be laid out. An interchange specification could be used for writers to adhere to (or a tool can provide an interface for) which could then be parsed by a runtime. A similar approach was explored by Mobramaein *et al.* [154] using a natural language interface for designers to inform and describe a procedural content generation system within a game engine. This is beyond the scope of this work but is important to consider.

3.4.9 Simulation Data

Groups, *Sequences*, and *Events* each have an integer determining the number of times that they are allowed to activate during simulation which is incremented when they are successfully executed. This integer can either provide a positive upper bound to limit the executions or be set to zero meaning there is no limit. If an activation request would put an element past its threshold, then the request should be declined irrespective of whether its *Condition* passes. As these objects within the model can be arbitrarily triggered through the API using the `trigger` *Logic* function, it is important to track the activation count. Moreover, any *Logic*, both internal to the model and even from the runtime using the API, can query the number of activations an object has had with the `activations` function. This could be used, for example, to make an object's *Condition* dependent upon one or more other objects' activation counts, or to provide alternative content within an *Event* based on activation counts, particularly within *Hub* and *Return* structures as discussed above. Additionally, a single element can only run one instance at a time; an already active element cannot be activated again until it terminates. By default, all container-like elements only remain active while their contents are currently active or have the potential to activate. Once all contents that could possibly activate have done so, the parent element will automatically terminate. If this behavior is not desirable, the elements contain a 'keep alive' Boolean that when true will cause the element to maintain its active state even when its contents have expired. This could be useful, for example, to force an exit *Function* to not trigger unless explicitly desired. This does not apply to *Events* as they have no contents, their lifetime instead being determined by the duration of the in-engine action that takes place in their `do` *Function* and are therefore one-shot by nature (i.e., activate, `do`, deactivate).

3.4.10 Links

Links tie together two sibling elements. *Links* cannot, however, be connected to a parallel element but can originate from them. Each *Link* has a static origin, a mutable destination, a *Condition* determining if the *Link* can be traversed, and a *Function* that is fired upon a successful traversal. During execution, if a set of *Links* are siblings (i.e., have the same origin) then their *Conditions* are evaluated to determine availability. If more than one *Link* remains valid, then the implementation is required to resolve the stalemate to proceed, such as by presenting a choice in the runtime to the player. The concept of choice here is deferred to the implementation and does not necessarily result in explicit questioning à la a dialogue wheel; it simply means that a particular narrative chain cannot progress until one option is chosen. If an evaluation of sibling *Links* results in none being available (or where a singular *Link* is used, it not being available), then the simulation would encounter deadlock. This is an inherent limitation of dynamic conditional link systems and is something to be resolved at a higher level than this model's structural descriptors.

From an authoring perspective, *Links* would be considered calligraphic [23], in that by default nothing is linked and authors create the *Links* manually.

3.4.11 Functions & Conditions

A *Function* is a set of logical statements that are able to, at a minimum, read and modify the *Story* state. *Functions* are used throughout the model at important structural points, mostly commonly entries and exits. *Conditions* are a specialized *Function* in that they must result in a Boolean and are not able to mutate the *Story* state. They are used as guards in the model to prevent or allow something.

3.4.12 Entities & Tags

Characters and other narrative objects are enumerated as *Entities*. An individual *Entity* is a unique instance of an in-engine object. *Entities*, like other objects, have an associated dictionary of *Attributes*. *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 fulfill that role. *Tags* can be used as wildcards to enforce restrictions on *Entity* participation in *Selectors*, as described below. For example, we may request all *Entities* of *Tag* `Enemy` and `Weaponized`, which would select all *Entities* that have these two *Tags* (enemies alone wouldn't suffice as they must also be weaponized to be included within the *Selector* result).

3.4.13 Selectors

A *Selector* is a special logical function written in *Logic* that returns an array of *Entities*. *Selectors* query only and are not mutating objects, meaning that they can only request and read the *Story* state within the model rather than write to it. As *Selectors* use *Logic*, they are dynamic by nature and can resolve in runtime. That is, by means of extended *Logic* function calls through the API, *Selectors* are able to query the active state of the runtime, such as getting all *Entities* nearby the player. *Selectors* are not restricted to *Events*, but most commonly occur as input to them, used to determine which *Entities* are involved in runtime. Consider the following example that demonstrates the usefulness of *Selectors* when partnered with *Events* and *Logic*. Let's assume a *Selector* has been created with *Logic* that requests the seven closest *Entities* to the player with the *Tags* `Enemy` and `Shielded`. As this is not a natively supported *Logic* function, the API would implement it and expose it to *Logic*, making it available for use here. When evaluated, this *Selector* would return up to seven *Entities* nearby the player that match the specified *Tags*. Within an *Event's* `do` function, for example, *Logic* is then able to optionally query a *Selector's* captured *Entities* with the `entities(selector: Selector)` function. Indeed, anywhere *Logic* can be used, *Selectors* can be queried and reused, not only in *Events*. This includes *Selectors* combining the outputs of several other *Selectors*, if desired. In this case, though,

Table 3.6 Mandatory *Logic* declarations required for core functionality.

Variables

`setValue(var: Variable, value: Type)>Void`
Sets a Variable value.

`getValue(var: Variable)>Type`
Gets a Variable value.

Entities

`entity(named: String)>Entity`
Gets an Entity by name.

`entities(withTags: [Tag])>[Entity]`
Gets all Entities with the given Tags.

`entities(selector: Selector)>[Entity]`
Evaluates and gets all Entities from a Selector.

`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(element: Type, delay: Int)>Void`
Triggers a given Group/Sequence/Event with a delay in ms.

`stop(element: Type, delay: Int)>Void`
Stops a given Group/Sequence/Event with a delay in ms.

`pause(element: Type)>Void`
Pauses a given Group/Sequence/Event.

`resume(element: Type)>Void`
Resumes a given paused Group/Sequence/Event.

`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.

`activations(element: Type)>Int`
Gets number of times the Group/Sequence/Event/Hub/Return has activated.

the *Event* would be able to utilize the dynamically retrieved listing of nearby *Entities* that match a given role rather than referring to exact *Entity* instances and the configured *Selector* could be shared and reused throughout the model anywhere *Logic* is present.

3.4.14 Logic Scripting

The *Logic* of this model is a set of functions for interacting with the *Story* state both for querying and for runtime manipulation. This collection is designed to provide minimal mandatory functionality for an implementation to fulfill as listed in Table 3.6. *Logic* does not detail itself with a particular language; it is equally viable to use LUA, JavaScript, Python, or anything else, as long as the mandatory functionality is supported. *Logic* is also chiefly how a runtime and the model communicate, happening in two different ways. Firstly, when considering the model's place within the greater architecture, the API

(which implements the model and exposes it to a runtime) is able to arbitrarily run Logic on the internal model, including returning of values, meaning that a runtime, through the API, is able to both query the model and explicitly direct it. This is particularly useful in situations where certain game states or triggers are handled within the runtime yet still wish to trigger aspects of the model. Secondly, *Logic* is intentionally designed to be expanded through a runtime. That is, using the API, a runtime can implement any functionality it wishes, both querying and mutable, *that will execute in the runtime*, and then create and expose a new *Logic* function for the authors to use. If we hypothetically implement the API (and therefore model) in C# and use LUA for *Logic*, then this can easily be achieved by writing custom functions in C# to be executed in the runtime, and then use LUA hooks to execute the C# code when calling a LUA function. This approach allows for *Logic* to remain neutral yet allow for arbitrary extension with code both querying and mutable running on a runtime being exposed to those using *Logic*. As a practical example, let's assume a runtime has defined a function that checks whether the player is near a defined *Entity*. This could then be exposed to *Logic* as, for instance, `isPlayerNearEntity(withTag: Tag, radius: Double)>Boolean`, which when evaluated by the API would call and return from the runtime function. This newly added *Logic* function can then be used in a *Condition* internal to the model to trigger an element dynamically. A similar approach could be taken to query and modify the actual game's state. *Logic* is therefore a bidirectional communication layer between the model and a runtime.

3.5 Worked Examples

In order to demonstrate the capability and flexibility of this model, we will now, using the model, step through worked examples from contemporary narrative-driven video games — *Life is Strange* and *Return of the Obra Dinn*.

3.5.1 Life is Strange

The chosen sequence from *Life is Strange* chiefly demonstrates the modeling of player choice, gating of progress, looping, and parallel narrative strands. A simplification of the sequence is displayed in Figure 3.12. Following the opening nightmare sequence, Max jolts awake during a photography lesson (1). She then gathers her bearings while a lecture takes place (2). During this, Stella drops her pen (3), Taylor bullies Kate (4), and Victoria's phone rings (5). These three events happen in parallel to the lecture and Max's pondering. At this stage, Max can interact with a photo on her desk, and after doing so, four more items become available (6). Interacting with all objects except the camera will not halt the lecture but have Max narrate them in parallel. If the camera is not selected in due time, the lecture begins to loop a series of questions to the class. When Max interacts with her camera (7), which disturbs the lesson, the narrative continues. Max is then punished for her intervention with a question (8), after which the bell rings to end class.

Fig. 3.12 A high-level overview of the *Life is Strange* sequence. 1. Max wakes up in class. 2. A lecture is taking place. 3. Stella drops her pen. 4. Taylor bullies Kate. 5. Victoria's phone vibrates. 6. Max's desk interactions. 7. Max's camera usage prompt. 8. Max's questioning for disturbing class.

Entities

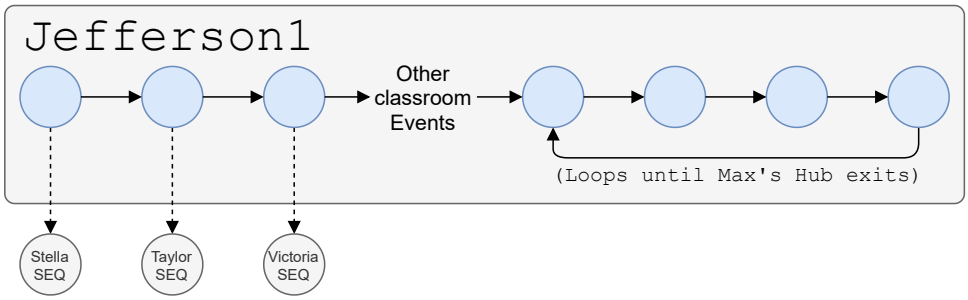
We must first decide and enumerate the involved Entities. Depending on the granularity desired, different objects may be included. For instance, fine-grained approaches may consider the paper ball Taylor throws at Kate as an Entity and model the individual Event for throwing it, whereas a coarser approach may represent the short bullying scenario as a single Event including two Entities. This is usually influenced by the particular use case. This concept of granularity applies to all of the model. For simplicity, we will take a coarse approach. As such, we can enumerate our characters as Entities — `Max`, `Jefferson`, `Stella`, `Kate`, and `Victoria`. Tags are not necessary for these Entities as we can rely on direct referencing.

Groups & Sequences

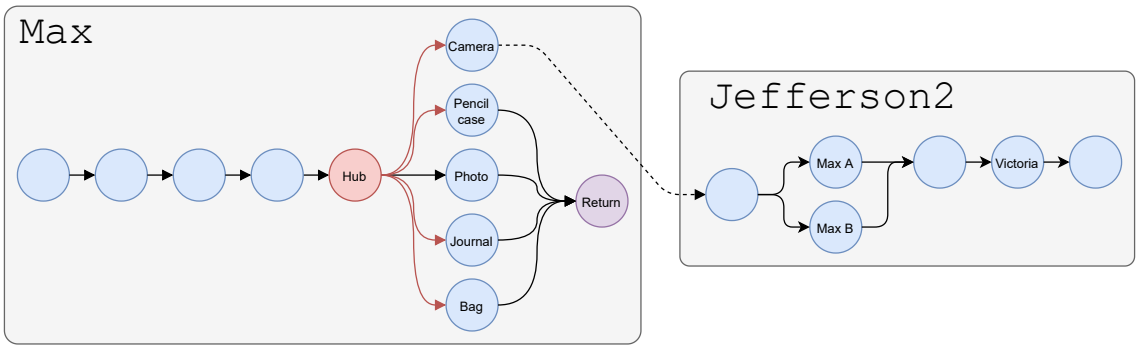
We can assume that the classroom scene is part of a single Group. Since there are several individual happenings that run in parallel and some that halt until given conditions are met, we need to define Sequences to represent this structure. Max talks to herself four times before being able to interact with all five items on and around her table. This happens while Jefferson's lecture is ongoing and therefore we can put it in a Sequence named `Max`. Jefferson's lecture can be broken up into two Sequences, `Jefferson1` and `Jefferson2`. The former runs until Max interacts with her camera, looping otherwise. The latter involves querying Max and the class ending which is instigated by Max interacting with her camera. We can also enumerate the Sequences `Stella`, `Taylor`, and `Victoria` for Stella dropping her pen, Taylor bullying Kate, and Victoria's phone vibrating. These short Sequences happen in parallel to `Jefferson1` and `Max`. An abstracted overview of these Sequences and their contents are displayed in Figure 3.13.

Max's Inner Thoughts

During the `Max` Sequence, Max narrates to herself four times. This can be thought of as dialogue with oneself. However, since the speech is inner (i.e., thinking), it may be



(a) The `Jefferson1` Sequence showing three parallel Sequences for `Stella`, `Taylor`, and `Victoria` that occur while other Events are transpiring. The Sequence ends with a loop of four Events until the `Max` Sequence (below) exits.



(b) The `Max` Sequence which gates narrative progress until the `Camera` Event is triggered, after which the `Jefferson2` Sequence activates. A choice is presented to the player with Events `Max A` and `Max B`, but they are immediately resolved with a split/join pattern.

Fig. 3.13 A high-level overview of the `Jefferson1`, `Max`, and `Jefferson2` Sequences, which are all part of the same parent Group. The `Stella`, `Taylor`, and `Victoria` Sequences contain corresponding Event chains.

advantageous to differentiate it from regular dialogue. Let's assume our implementation has defined a `thinking` function that requires an Entity and textual string, which now becomes exposed to Logic and can be used within the model. This can then be used within an Event's `do` Function, reading the instigating Entity and providing the spoken string. For Max's thoughts, these Events would simply have Max as the instigator and call the implementation's `thinking` function from Logic code, passing in the instigator and a text string for whatever Max is thinking for the particular Event.

General Dialogue

More general dialogue could be handled with a defined `dialogue` function, again provided by the implementation. This would largely mimic the `thinking` function but differ in that it takes two sets of Entities read from the instigators and targets. For example, in the `Jefferson1` Sequence, Victoria and Jefferson engage in conversation. These individual speech acts could be represented by a sequential set of connected Events that use the `dialogue` function with appropriate Entities selected for involvement. In the case where Jefferson addresses the whole class, if we wanted to include all students as target Entities, then we could use a Selector to directly reference them all, or if a `Student` Tag had been assigned, select all Entities with that Tag.

Parallel Sequences

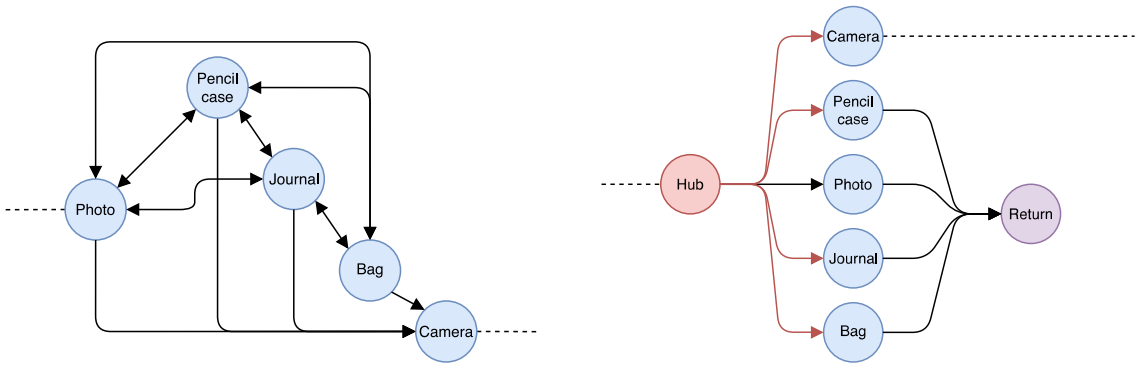
The `Stella`, `Taylor`, and `Victoria` Sequences all execute during `Jefferson1` at fixed times. We could handle this in a variety of ways. If specific timing is known, all three Sequences could be started upon entry of `Jefferson1` with a delay using the `trigger (seq: Sequence, delay: Int)` Logic function. They could also be triggered after a particular Event for more fine-grained control with or without delay. Alternatively, the Sequences could have a starting Condition that checks a truth value of a Boolean Variable defined within the model which could be set to true to trigger the Sequences. If considering a pure in-model solution, this would be set in Logic (e.g., in an exit Function of an Event or Sequence). As mentioned earlier, Max's four thoughts are parallel to Jefferson's lecture. This can be implemented by having the `Max` Sequence as a sibling of `Jefferson1` 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 remain inactive until the photograph is first picked up. This could likewise be implemented in multiple ways; we will look at two approaches.

The first approach, shown in Figure 3.14a, uses complex interlinking between non-parallel Events within the `Max` Sequence. The `Photo` Event, following Max's pondering Event chain, acts as a gate for the other Events as it must be encountered first. To represent the cyclical nature of this setup, all Events (except the `Camera`) are connected to themselves and are interconnected to all other Events (represented in the diagram as double-headed arrows for simplicity but recall that bidirectional links are not a feature of the model), which allows the player to freely choose between all Events repeatedly until the `Camera` Event is chosen which breaks the loop. This approach has the advantage of not using variables to track state and emulates the gating and looping mechanic through structure alone. However, the linkage is fairly complex making it difficult to both create and modify. Moreover, as this setup can only use `entry`, `do`, and `exit` functions, it is unable to respond, without significant work, to the looping mechanic.

The second approach, shown in Figure 3.14b, uses a Hub and Return setup to simulate the same gated and cyclical network. All five Events are connected to the Hub, with all but the `Camera` Event being connected to a shared Return, which means that the player can indefinitely cycle between all of the available Events until breaking the chain. To gate the player behind the `Photo` Event, a Boolean Variable is created and used as the returned truth value of a Condition attached to each of the other Events (shown in the diagram as red arrows). This means that when the player initially encounters the Hub, only the `Photo` Event is present, which when chosen will set the Boolean to true, and therefore upon returning to the Hub, all other Events become available. Although this setup requires the use of a Variable and Condition, it is easier to configure as the



(a) Use of complex interlinking (simplified as double-headed arrows) between nodes except with `Camera`. `Photo` is first, creating a gating effect.

(b) Use of a *Hub* and a shared *Return* using conditioned links (indicated in red) on all but `Photo` to create a gating effect.

Fig. 3.14 Two approaches for handling Max's interactivity gating in *Life is Strange*.

complexity of the linkage is significantly reduced. It is also more extensible as adding a new Event into the cycle (even gated) simply requires an Event to be connected to the Hub and Return only and having its Condition assigned. On the other hand, doing the same in the previously discussed approach would require significant interlinking to each existing Event, which increases in difficulty as the number of Events increases. This approach also benefits from the specialized Return and Hub functions, which while not used in this example, enable it to respond to the looping mechanic.

If Max does not interact with her camera in time, `Jefferson1` begins to loop a set of four dialogues from Jefferson to the class, which can be achieved by simply linking the final Event to the first of the four dialogues. Once Max interacts with her camera at *any* point that it is available during `Jefferson1`, the Sequence terminates and `Jefferson2` initiates. This can be handled by setting the Function of the Event representing the camera interaction to use the `stop(Jefferson1,0)` and `trigger(Jefferson2,0)` Logic functions in order. This would firstly terminate `Jefferson1` regardless of its progress, and start `Jefferson2` immediately thereafter, which is how the game works.

Player Choice

`Jefferson2` starts after Max interacts with her camera. In this Sequence, Max is made to answer a question as punishment for disturbing class. This presents two options to the player, both of which reconcile to the same Event to give a perception of agency over the narrative. We simply need to link the preceding Event for dialogue to both potential answers, as multiple outputs must be resolved before continuing. The way in which the choice is presented is left up to the implementation. In the case of this example, a dialogue wheel is presented to make a binary choice which would in turn trigger the model to follow one pathway. Similar cases later in the game are more complex as they pause all other elements during execution. This can be achieved by manually using the `pause` and `resume` functions, or by marking the appropriate element as topmost.

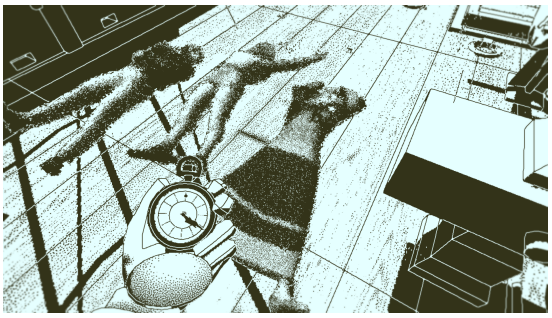
3.5.2 Return of the Obra Dinn

We will now look at how we can use the model's structural hierarchy to allow for some forms of *Discoverable Narrative* by analyzing a segment of *Return of the Obra Dinn*, a contemporary murder mystery game that makes extensive use of this concept. The *Obra Dinn* is a ship insured by the East India Company that went missing in 1803 and has since washed up with all sixty passengers dead or missing. The player's task as a Chief Inspector is to determine the fate of all aboard including their names, where and how they met their fate, who or what their killer was, and if applicable, their location, should they have survived. Upon boarding the ship, the player discovers several decomposed bodies and is given the freedom to roam most of the ship, although several regions are initially locked and become available as the narrative progresses. The player is also given a special pocket watch which when activated near a corpse will trigger several seconds of audio prior to the person's death, followed by a limited freeze-frame of the exact moment of death that the player can navigate around freely. Within this narrative vignette, the player can make note of the scene's details to aid their main task and find other corpses to further traverse back in time through the events prior to their current scene. The end goal of the player is to gradually piece together limited, vague, and unordered fragments of information through these vignettes to understand the exact sequence of events that led to the crew's fate.

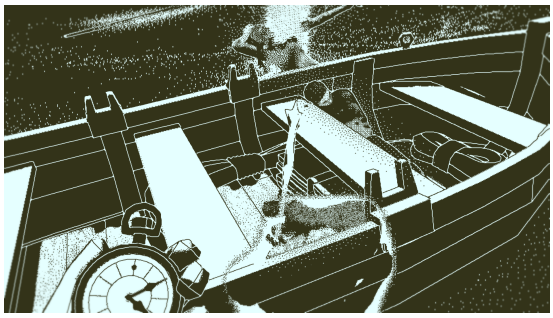
The game is presented in chapters that are made up of a varying number of parts that form a sequential narrative. The player does not necessarily access them chronologically, often beginning in the final part of a chapter and then gradually traversing backwards one part at a time by finding and interacting with the deceased or other related items. While access to some parts is always gated by others, once discovered the player can typically reenter a chapter at any point they choose, only being able to traverse backwards from that entry point. For example, if a chapter has six parts, the player may begin by entering part six and making their way backwards to part one, then having the freedom to enter some (if not all) of the parts in an arbitrary order.

Figure 3.15a shows an example of the player viewing Bun-Lan Lin's body on the main deck which when interacted with using the pocket watch will transport the player to Chapter IV, Part 4. For this chapter, there is no way to access previous parts without firstly entering through Bun-Lan Lin's body. Once the player enters a given scene, they are able to locate and interact with previously deceased characters which act as triggers to backtrack through the rest of the chapter. Within this particular scene, the player can interact with three of the deceased, equating to the preceding three parts of the same chapter. Figure 3.15b shows the player discovering and highlighting the deceased Patrick O'Hagan which when interacted with will project the player to Part 2 of the chapter.

We can model this structurally using Groups, Sequences, and Logic. We can first declare a Group for the overall chapter, `ChapterIV`, and then create and nest a Sequence within that Group for each of the parts. These nested Sequences must be parallel and have



(a) Entering Part 4 of Chapter IV 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'Hagan's body.

Fig. 3.15 A player in *Return of the Obra Dinn* entering into Part 4 of Chapter IV by interacting with Bun-Lan Lin's body on the main deck. From Part 4, they discover Patrick O'Hagan's body, which can take them to Part 2 of the same Chapter.

Conditions set to never activate, as we will be doing so manually with Logic. Each child Sequence would contain a number of parallel Events, one for each of the parts prior to it. An in-model solution would see these Events be guarded against immediate execution with a Condition that uses a custom function implemented in the runtime and exposed to Logic through the API for checking interaction with a given Entity with the pocket watch (in this case, deceased characters). Alternatively, the Conditions could simply disallow the Events from firing at all and the runtime could manually trigger the Events through the API when the game internally detects interaction with the appropriate objects.

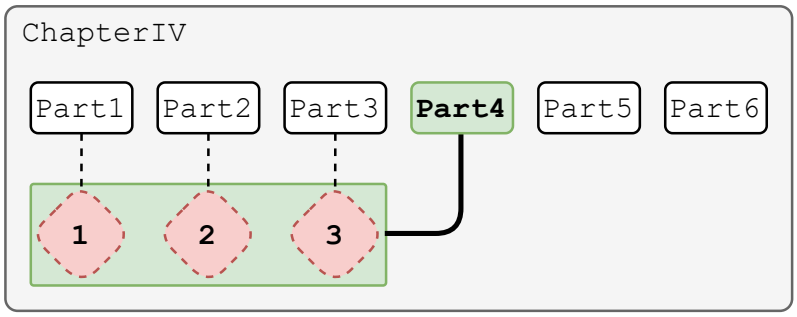


Fig. 3.16 An abstraction of `ChapterIV` from *Return of the Obra Dinn* showing parallel Sequences for each Part. `Part4` is expanded, showing three parallel Events that trigger the three preceding Parts when interacting with the deceased within the scene.

This structure is visualized in Figure 3.16 with the `Part4` Sequence expanded, showing its three internal Events that link to `Part1`, `Part2`, and `Part3`. We can then use the exit Function of the Events to handle the consequences of interaction. In this case, we could terminate the current parent Sequence by using the `stop(sequence: Sequence, delay: Int)` function, and then start the new Sequence by using the `trigger(sequence: Sequence, delay: Int)` function. While not necessary in this case, we could string an additional set of Events to those already discussed to have a set of actions occur after interaction but before switching Sequence. Alternative to this structural setup, we could have used Discoverable elements instead of Sequences, which would provide us with

a queryable set of *Discoverable Narrative* enumerations that could further enrich the possibilities from within and external to the Discoverable. Another advantage of using Discoverables instead of regular Sequences is that a runtime, querying the model, may wish to differentiate between regular Sequences and Discoverable Sequences regardless of the additional functionality provided.

3.6 Conclusion

This chapter set out to apply existing approaches to narrative modeling onto video game narratives to discover the advantages and disadvantages of each approach, and in response, design a new model that was able to better capture aspects of video game narrative, contextualized by being seated within a complete authoring pipeline.

A complete architecture of video game narrative authoring, consisting of several components including a narrative model, was first proposed. This architecture, while imposing limited constraints (as all architectures must), remained neutral to how the resulting narrative was realized and was not tied to a specific game engine or genre, ensuring that it was usable in a wide variety of contexts. The main purpose of this architecture was to provide a framework within which to design a new narrative model, meaning that the model's design can be informed by its neighboring components to better situate itself in the greater process, as opposed to being developed in isolation which risks a disconnect between the model and its neighboring components when deployed.

Following was a practical application of four narrative models that varied in technique to two contemporary video games of differing narrative complexity. Each modeling approach was evaluated with its application being described, during which the advantages and disadvantages of the models were discussed in situ.

From this application, it was found that aspects of video game narrative that are discoverable, observable, or experienced were not well represented. This led to the creation of my concept of *Discoverable Narrative* that provided a vocabulary for the analysis and categorization of such narrative aspects. This was accompanied by a set of worked examples from contemporary video games to demonstrate how it functions.

Finally, one of my major contributions, the *Novella Narrative Model*, was presented along with two worked examples to demonstrate its functionality. This model was developed with the architecture in mind meaning that it was contextualized with respect to its neighboring components within the authoring, integration, and realization pipeline.

The discussion and application of existing models to video game narratives, creation of *Discoverable Narrative*, and the *Novella Narrative Model* each directly answer the first research question. This research question sought to not only understand the capabilities of existing models for capturing video game narrative complexities, but to also use these findings to create a new narrative model that may better support these intricacies. Moreover, the *Novella Narrative Model* fulfills a major component of the proposed game

narrative authorship architecture which serves in part as the foundation for the design of a new authoring tool in §5 which in turn is used within a study in §6, targeting the third research question.

The unique approach taken by the *Novella Narrative Model* for the representation of video game narrative contributes several key innovations in its methods.

Game Content Support. This model was built with game content in mind. Through the use of *Logic*, *Attributes*, *Tags*, and *Selectors*, the narrative model is able to build up an interchange for the content of *Groups*, *Sequences*, *Events*, *Hubs*, and *Returns*. This moved beyond what has been provided by some existing models of interactive narrative which supply rules and descriptions of structure but not the content itself. The additional content model can be used to indicate and communicate the effect of the story on characters or the game environment, changes in rules and systems, or to communicate how something is displayed or presented within the user interface. For example, making a specific choice might put a particular item into the character's inventory (use of two *Entities*), a scene may change dynamically if there are hostile enemies nearby to the player (use of *Selectors*, *Tags*, and *Logic*), or a scene might occur that is not directly delivered to the player in the foreground but appears in the background as a piece of environmental storytelling (use of *Attributes* along with *Discoverable Narrative*).

Extensibility and Deference. An advantage of the model being built within an architecture is that its design can consider its neighboring components. This enabled the *Logic* grammar of the model to be extensible as through the use of an API (as part of the architecture) the mandatory function list can be added to on a per-game basis (see §3.4.14 for details). The way in which *Logic* works also takes advantage of the greater architecture by deferring the actual execution of in-model *Logic* to the API implementation, and allowing the extensions made to *Logic* to be executed within the runtime environment. Additionally, elements within the model have dictionaries of arbitrary data attached to them, meaning that a runtime can require any information that it desires as the data is not predetermined. Deference to runtimes can also be found throughout the various uses of *Logic* within the model. For example, by using extended *Logic* functions within a *Condition*, the model is able to dynamically activate elements based on conditional code executed within the runtime itself. Similarly, the actual functionality of *Events*, handled in their `do` *Function*, is deferred in the same way, with the actual runtime effects being executed, through *Logic* extension, in the runtime. This approach is flexible in contrast to predetermined elements with a model where conveniences are made at the expense of flexibility. On the other hand, this does increase the complexity of authoring certain features but does so to avoid introducing platform or genre-specific enforcements.

Bidirectional Communication. Another advantage of being situated within an architecture is the consideration of interactions between the other components with the model. This allowed for planned bidirectional communication between the model and the runtime through *Logic*. The first is through extension, as described above, in that

Logic can be written in-model but executed in the runtime. The second is execution of arbitrary *Logic* code in-model by the API, which means that a runtime, using the API, can at any point query or manipulate the model. Both are discussed in more detail in §3.4.14.

Discoverable Narrative. A limited set of *Discoverable Narrative* elements are supported by extending a standard *Sequence* to expose the respective enumerations, allowing for *Logic* queries so that the same element, other elements, or even the runtime, are able to react accordingly to the values. The second worked example demonstrated how this is also partially supported through the model's structure. As discussed in §3.4.8, it is also possible to emulate environmental storytelling as part of the *Attribute* system. This is an area that could be improved to better support *Discoverable Narrative*, although the tradeoff for supporting each combination comes with its own cost of an increase in complexity to the model and potentially its structure.

Neutrality. This model has been designed to be free from any specific runtime or genre-specific tropes, aiming to provide a vocabulary that is able to describe as wide of a variety of video game narrative as possible. One of the main ways this is achieved is through the deference found in *Logic* and *Attributes*, as by making the realization of actual runtime behavior external, the model is freed from the specifics of what needs to happen. However, this comes at the cost of increased complexity to achieve things that are native to other models. Take, for example, a model for an adventure game, which may impose a room-based structure with interactable contained objects. This is possible in my model through structural or parallel connections, *Entities*, and *Logic*, but the task of doing so requires significantly more authorship and complexity than using functionality that's already native to a model. However, by introducing these kinds of abstractions and conveniences, models restrict themselves and impart genre-specific limitations.

CHAPTER 4

Interface Paradigms & Workflow

In §3, we established a complete architecture for video game narrative modeling, consisting of a demonstrated narrative model, an interchange format, and an API, which work together with an external engine and authoring tool. We will now turn our attention to the authoring tool component, which permits authors to create stories in terms of the underlying model, and begin to investigate existing variations in designs and their impact upon authoring workflow. In this context, 'authoring workflow' refers to the methods by which authors write in terms of frequency of actions taken, focus on particular actions, and the ordered flow of the actions. We can achieve this by analyzing the current state of the art in authoring tool design through a user experience study that examines these factors, resulting in an understanding of common design variations and their potential impact upon authors.

Despite the rich history and popularity of interactive narrative, authorship remains a notable challenge. The technical skills required to develop these stories can act as a gatekeeper preventing creatives from using the medium [149, 201]. The research community's collective solution to accessibility for creatives in this space has been authoring tools — software aiming to enable creatives to design and build interactive narratives — many of which were examined in §2.4.2. While some of the research in this space inherits design principles from programming environments, interactive narrative authoring tools often aspire to the added challenge of providing access to less technical or at least non-programmer users — authors who might use such a tool to create interactive stories with a creative or scholarly, but not necessarily technical background [150]. This demands careful consideration of the interface design and UX of these authoring tools. While there are established UX principles/heuristics [56, 57, 114] and evaluations [8, 175, 176] for the games or interactive experiences themselves, comparatively little of this work focuses on authoring tools, the emphasis being firmly on the experiences created. This has led to a UX focus in this space on the audience rather than the authors. The few works reviewing the state of the art in IDN authoring technology that do exist are concerned more with classification and identification than critical evaluation [189]. While these classifications are valuable, they lack the careful exploration of a UX evaluation, telling us more about what exists rather than what is needed in an effective authoring

tool. It is possible that limited evaluation work on authoring tools is due to the fact that evaluating interactive narrative tools presents unique challenges for UX research, in a field where established methodologies often employ 30–60 minute user study tasks [81], and where writing a meaningful story takes significantly longer. This can make the use of the tool challenging to effectively evaluate with existing techniques and demands a new approach to UX evaluation in this area.

This chapter sets out an evaluation of the state of the art in this critical area of authoring for interactive narrative, presenting a novel methodology for evaluation of authoring tools, and describing a key set of identified principles that may be derived for future authoring tool design and evaluation. This work seeks to establish these key principles to answer the question of *what the impact is on authoring workflow for different user interface paradigms within interactive narrative authoring tools*, which directly targets the second research question of this thesis. Through these principles, it aims to build a better understanding of what creative impact a given authoring tool has with respect to its exercised interface paradigms upon the authoring experience for creation of interactive narratives, which lays the foundations for answering the third research question.

The entire contents of this chapter have been present in several publications. The initial authoring tool selection, including identification of user interface paradigms, was published at *ICIDS 2018* in a paper nominated for *Best Paper* [85]. The statistical clustering of authoring tools was originally published in the *Authoring for Interactive Storytelling Workshop* alongside the *ICIDS 2018* [86]. The chapter in its entirety was also published in the prestigious ACM *Journal on Computing and Cultural Heritage* (JOCCH) [88].

4.1 Authoring Tools & Study Preparation

When designing the user interface of any system, whether it be an application, a website, or another interface, it is important to identify and understand the way in which design decisions will impact a user's experience with the system. If interacting with a system that is unclear or difficult, cognitive friction [45] will rise, which can bring about frustration and result in a negative user experience. This is especially important for authoring systems where we want to avoid distracting users from the creative process of authoring an interactive narrative. General design heuristics for the design of and interaction with interfaces have existed for quite some time [163]. These principles are ideal for high-level design but do not aid us in understanding the impact of these design decisions in domain-specific processes, such as the creative workflow of authoring for an interactive story. Therefore, we must go beyond these best practices to identify the impact of various design decisions within the context of authoring interactive narratives. By determining the impact upon creative workflow that different design decisions have, we can better inform designs of new authoring systems.

4.1.1 Representative Authoring Tools

As mentioned above, evaluating the UX of authoring tools presents unique challenges in that the tasks typically undertaken by users are as long and varied as any creative work. Furthermore, there are a variety of technologies available in this space and to begin to study the state of the art in this area demands we identify a representative selection of the existing approaches. The authoring tools considered in this study are a subset of those discussed and categorized in detail in §2.4.2, as that listing built upon the initial collection used for this study at a later date. In total, 29 individual authoring tools were included in the survey. 14 of the tools 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 or otherwise free projects. 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 these tools both in terms of features and user experience.

Availability

When evaluating the included authoring tools, it is important to determine their availability, in this context referring not only to the accessibility of the end product (e.g., executable binary, web service, etc.) but also their online presence and whether their source code is accessible in the case of products that are licensed accordingly. We must also account for systems that used to be available but have since succumbed to fates such as the presence of dead links, terminated websites, and software entropy[44].

The availability of a tool directly impacts 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 be devalued relative to their initial contribution, as they are unlikely to be adopted in the long term by authoring communities unless kept at least functional. *Inform*, for example, has been developed and used since 1993, but due to its strong online presence and dedicated community-driven development, it still remains a strong contender among the interactive fiction authoring community even today.

Online presence is an area that academic authoring tools struggle to maintain. Of the 14 selected tools from an academic background, only six ever had a dedicated website, of which only four still exist today and are seldom, if ever updated. All other tools (except *HyperCard*, which was discontinued in 1998) have an active online presence either through dedicated pages or a repository in the case of open-source projects. It is not uncommon to encounter research projects that are hosted on a specific academic's personal page. These websites are susceptible to becoming invalid, such as when the researcher leaves the institute and the personal pages are terminated. This, and other temporary websites often result in unreachable links and are not a suitable substitute for a long-term dedicated project page where tools can be publicized. To illustrate this,

an article of the academic authoring tool *EmoEmma*[45] [40] provides links to academic papers and binaries, but all of the links reside on a university staff page that no longer exists, making it difficult, if not impossible to source the original contributions.

Another area that the selected academic tools struggle with is the distribution of binaries (or other runtime) and source code, where applicable. Some projects are understandably protected intellectually, but those that are not should attempt to share their work to better maximize their chance of adoption by authoring communities and to provide opportunity for further research to be conducted. Of the 14 academic tools, only five ever offered binaries at some point, with only one remaining publicly available today (*StoryPlaces* [98], which at the time of writing offers both a web service and source code). Sometimes software is made available upon request. *StoryTec* [79], for instance, used to provide a software request form online, but has since removed it in 2015[46]. The *ASAPS* [110] software has a request procedure that I had followed in mid-2018, but I was unsuccessful in obtaining the software as no response was provided with adequate time given. This highlights the need for care to be taken to ensure that if software is intended for 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 and authoring research communities, as it prevents the reproduction of any experimental results and hinders their study so that we might incrementally improve and iterate upon existing work or form a greater understanding of existing works. While some conclusions can be drawn from documenting articles or publications, this is not a suitable replacement for interacting with the software itself where a more direct evaluation can take place.

For the authoring tools that were included in this study, 15 out of 29 were available for public use. Both the tools sold as commercial products and those from other non-academic sources were wholly available, contributing 14 of the available tools, with only one of the academic tools being available for use. This further demonstrates the trouble of availability within the academic space.

Clustering of Tools

In order to determine a representative selection of the broad authoring tools available for study, we can identify clusters of tools based on their features and then pick representatives from these clusters. As the focus of this investigation is the UX of these tools, the features come from both UX and systematic affordances of the applications rather than only the capability of their underlying narrative models. This includes, for instance, the presence and capabilities of scripting, how content is displayed, how editing is chiefly done, the presence and capabilities of testing, the delivery methods of the software, and the exporting functionality, among others. A complete listing of the features used can be found in Appendix B.1.

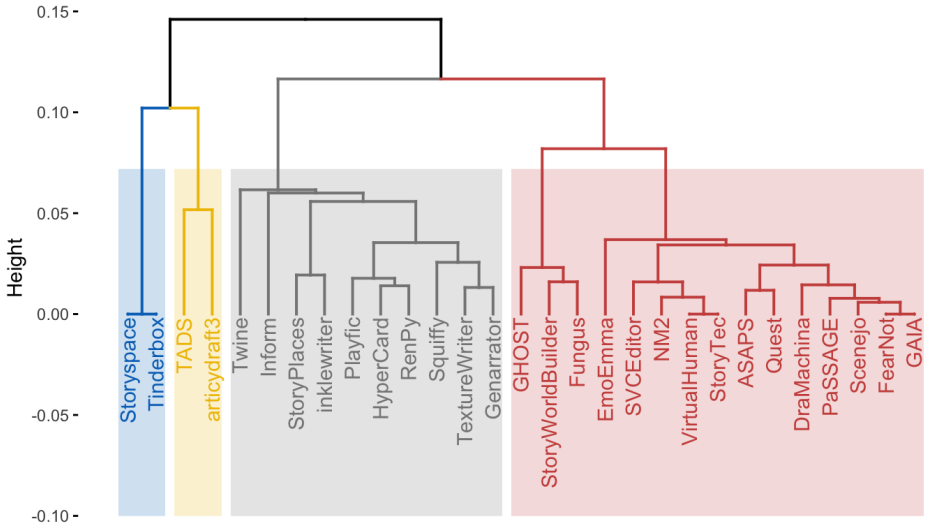
Each tool was systematically reviewed through a combination of actual usage and a review of relevant documentation and research papers to identify which features it had. The list of features was then gradually built up as they were observed in each tool, and when a new feature was identified, all previous tools were revisited and evaluated appropriately. It should be noted that while every effort was made to retrieve and use the tools, in some cases, as discussed above, they were not available. In the case of a tool not being available to use, features were inferred from research papers and other sources like project pages or presentations. If a given feature for a tool could not be ascertained, the feature was assumed not to be supported instead of imputing to avoid the declaration of non-existent features.

Clustering was done using the `R` language and related third-party packages, particularly `FactoMineR` [126] for clustering. As the data has individuals described by purely categorical descriptors (i.e., only enumeration and binary columns), Multiple Correspondence Analysis (MCA) was used as a preprocessing stage, followed by Hierarchical Clustering on Principal Components (HCPC) to determine clustering of the tools based on the features, as described in the `FactoMineR` documentation. 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[47] as the number of clusters increases. In this case, I used the number of clusters that were suggested by the algorithm, which was four.

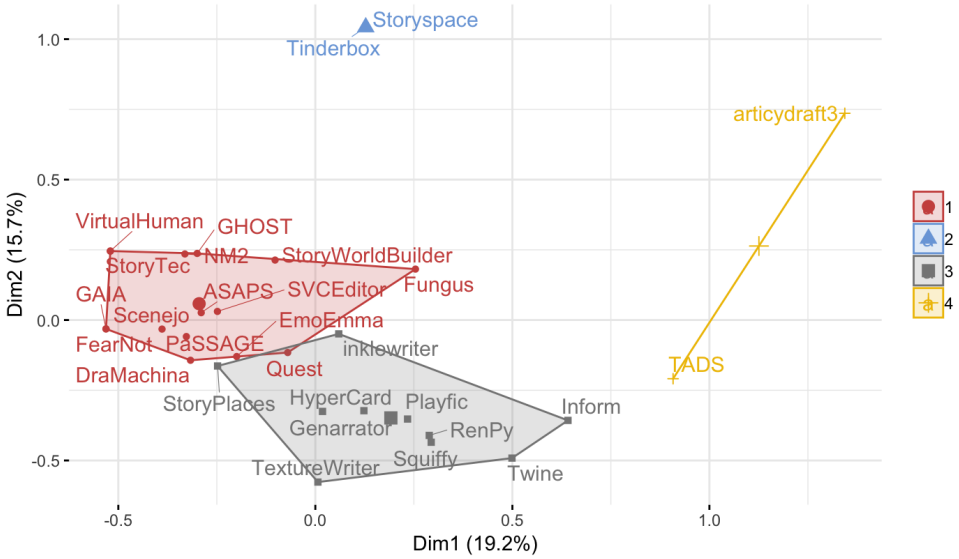
A planar projection of the endpoints of the 3D tree generated by the HCPC algorithm can be seen in Figure 4.1b. This figure visualizes the clusters by color. Note that distance in two dimensions does not necessarily correlate to similarity as the points are the projected result of a branching hierarchical tree in three dimensions. However, the dendrogram depicted in Figure 4.1a 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.

Selected Representative Tools

We must consider the side effects of small datasets when interpreting these results. Clustering reliability would increase as the total number of tools and features increased. The number of tools cannot be controlled as there is a limited count of specialized public interactive narrative authoring tools. Features can always be improved, but with increasing granularity comes an increase in resources and a higher chance of clusters not forming due to features being too specific to the individual tool that they originate



(a) Dendrogram of authoring tool HCPC result clusters. Groups are indicated by colors.



(b) Planar projection of the 3D HCPC clustering algorithm. Groups are indicated by colors.

Fig. 4.1 A dendrogram and planar projection of the HCPC clustering algorithm output.

in. For the purposes of selecting a subset of representative authoring tools, the number of tools and features are acceptable.

For this study, *Quest*, *Inform 7* (*Inform*), and *articy:draft 3* (*Articy*) were chosen. The first two of these tools come from the largest clusters in the analysis, and the third from one of the smaller clusters. Each of these programs presents variety in their UX and exercised UI paradigms. Figure 4.2 shows the three authoring tools in context. *Quest* makes heavy use of multifaceted system controls such as text boxes and combo boxes, *Inform* hosts a natural language scripting system within an augmented text editor, and *Articy* primarily uses a dynamic node graph. This variation ensures that as many of the primary UI paradigms can be tested, rather than choosing, for instance, two systems with similar interaction paradigms from different clusters. From their individual clusters, *Articy* was chosen due to its popularity within creative industries[48] and *Inform* was chosen due to its popularity with interactive fiction communities[49]. *Quest* was chosen as within its cluster only itself and *Fungus* were available for use, and *Quest* presents a much more multifaceted-centric experience, which was the only paradigm not yet covered. *Fungus*, on the other hand, does have multifaceted controls, but also requires a graph and scripting, both of which were already covered by the other selected tools.

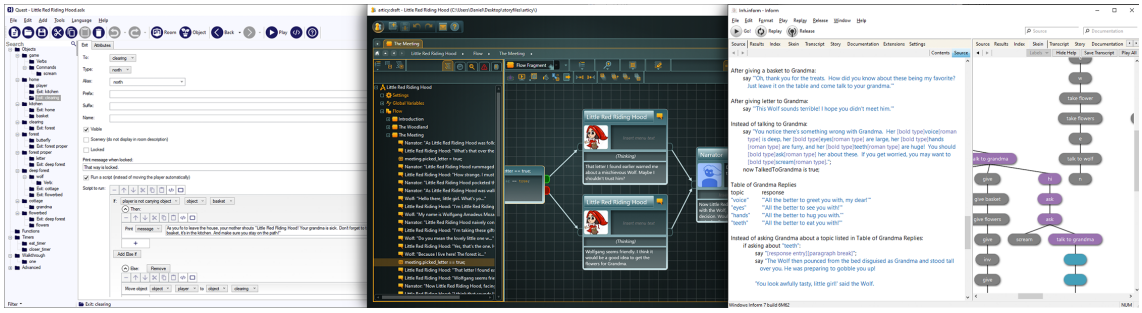


Fig. 4.2 The three authoring tools included in the study. *Quest* (left) primarily uses a multifaceted control-based interface, *Articy* (center) primarily uses a dynamic node graph, and *Inform* (right) primarily uses an augmented text editor.

Representation of the Selection

The scale of the greater population of authoring tools demands that in studying them we must take a representative sample. The process described above demonstrates the means of taking a representative selection of authoring tools from the greater population, which in turn were analyzed to identify three representative tools for this study.

Inevitably, the resulting selection does not include all tools. Despite this, those present in the larger review in §2.4.2 that are not present in this study are still broadly represented within this selection. For example, the *Arcweave* authoring tool chiefly revolves around a node graph with rich media connected by conditioned lines, *Celtx Game* similarly uses a node graph for editing, although without rich media, and *HypeDyn* presents a traditional spatial hypertext canvas for the freeform organization of text-based nodes, which can also be found as the primary editing method within *Twine*, *Storyspace*, and *Articy* in

this study. *Inky*, *Chronicler*, and *Villanelle*, on the other hand, are authoring tools that wrap a domain-specific language, presenting an augmented text editor for authoring of the narrative, along with select features beyond the text editor such as in-tool testing, which can also be found in *Inform*, *Squiffy*, and *Playfic* within this study. The authoring tool selection also covers a wide range of tool contexts from games authoring (*Articy*) to traditional hypertext narrative (*Storyspace* [20, 21]) to locative systems frequently used to deliver narratives for cultural heritage (*StoryPlaces* [98]). It also covers a variety of sources ranging from paid proprietary tools like *Articy* to free community tools like *Inform* to academic research projects such as *EmoEmma* [40].

Consequently, the representative selection of authoring tools, and those ultimately chosen from the clusters for use within the study, are representative of the wider field of related authoring tools, including those found within varying disciplines, even though it was not possible to individually analyze each individual tool.

4.1.2 Representative Story Features

As this study focuses on interactive and game narrative authoring tools, any UX experiment must make use of a task for participants that mimics contemporary game writing. Since there are countless variations within the vast corpus of video games writing, we likewise need to use a representative selection of writing challenges and features for this task that mimics the genuine act of creating interactive narrative. Towards this, feature analysis of a set of story-focused games was done to determine a listing of commonly employed storytelling features. These features were derived from common game design theory literature such as Mason's model of choice [141], Booth's narrator classification [28], Jenkins' refinement of environmental storytelling [103], the concepts of *Discoverable Narrative* as described in §3.3, and other tropes found in the medium as described in other literature [6, 186]. The frequency of occurrence of these features across the selected games was then used to influence task design in the story templates, as discussed below in more detail.

In total, 17 video games were included in the analysis from a broad range of genres, typically being selected based on their prominence as a narrative experience. Each game was played through and analyzed to determine high-level narrative features employed by the authors. `R` was then used to conduct a frequency analysis of the individual features over all games. Results can be seen in Figure 4.3. It is clear from this graph that some features, such as perpetual endings and interactive cutscenes, are insignificant within the corpus analyzed, whereas others, such as standard cutscenes and environmental storytelling, are prominent. A complete listing of the features, their detailed descriptions, and the included games can be found in Appendix B.2.

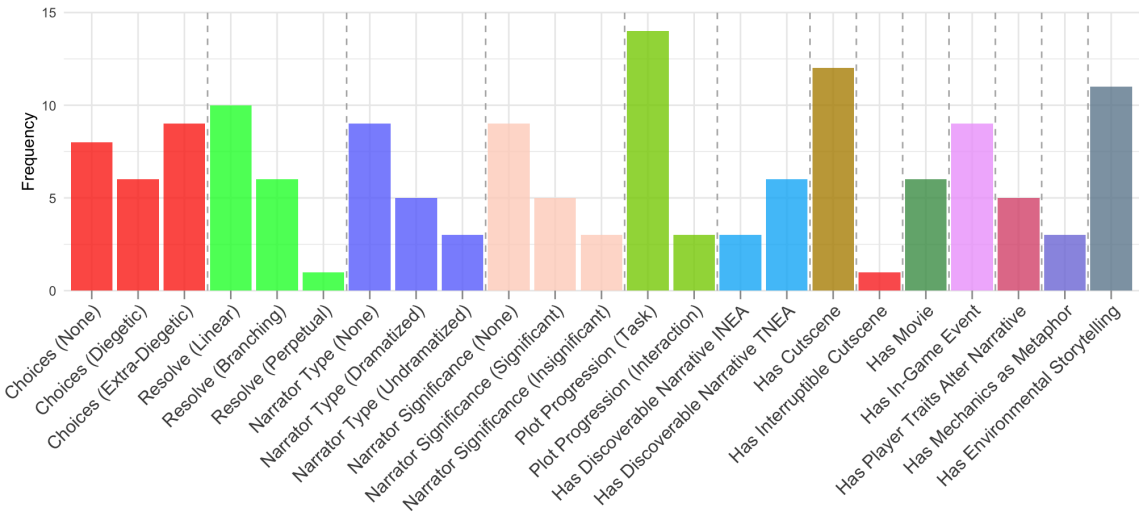


Fig. 4.3 Game story features frequency analysis. See Appendix B.2 for more details.

4.1.3 Evaluation Story Template

When designing the methodology, it was initially considered to allow participants to write a story of their choosing as to not limit creativity. However, as previously discussed, this is challenging to study within this domain given the time it takes to write a full interactive narrative, and the potential fluctuations in participant creative inspiration. Taking into account the demographic, it was elected for the methodology to instead have participants finish a partially completed story. Participants would be presented with a partly written story and asked to complete it and its missing features while also giving them some limited creative freedom in how they did this. This way, the context under which participants were acting was relatively consistent, there was sufficient time to pragmatically evaluate the UX for each participant, they were undertaking to write for a story representative of interactive narrative, and the chance of writer's block coming from a blank page (which might waste study time) was reduced.

The common game storytelling features identified earlier were formulated into a planned interactive version of Little Red Riding Hood (LRRH). This particular fairytale was chosen due to its ubiquity and international presence to reduce the chance of a participant being unfamiliar with the story's arc. It is also a commonly used story within the IDN research community[50]. The inclusion of a partially completed LRRH adaptation serves as a means for participants to interact with the authoring tools in a context that is relevant and populated with recognizable content. The story was broken up into an introduction and three distinct chapters. 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. The complete LRRH script, written in an interactive screenplay format, is available in Appendix B.7.

A story template implementing the LRRH script was then created for each of the chosen authoring programs. This template amounted to an adaptation of the LRRH story detailed above in each authoring tool in the medium appropriate to that tool. Certain liberties were taken with the story as demanded by the medium — for example, adapting the story for the adventure game, room-based style of *Quest* and *Inform* resulted in a quite a different interactive version of LRRH to that of the dialogue trees favored by the narrative model in *Articy*. However, each template still followed the same overall chapter structure and still made use of the representative narrative features described above. This template would later be edited to remove a section which participants would be asked to complete as the basis for the study.

4.1.4 Training Videos and Documentation

Short annotated training videos were prepared for participants to watch prior to their use of the tools to ensure a baseline level of knowledge for all participants. The three training videos introduced the interfaces and their essential functionality by building a fictitious story that exhibited many of the features of the LRRH story that they would be using in practice. Care was taken to ensure that the tutorial videos differed from and did not include aspects of the LRRH story, but instead provided the participants with an understanding of the tools required to complete the study. A documentation webpage was also created for participants that was available to them to read prior to and throughout the study. The page contained tutorial text explaining the procedure of the study as well as instructions on what they are expected to do. As each chapter in the LRRH story was based upon a set of typical authoring tasks in games writing derived from the story features analysis, an overview of the LRRH story chapters broken down by the respective tasks that they are made up of was presented in this document also. This consisted, for each task, of the name, a general description of the task, and a grounded example of how it was implemented into the story template. The remainder of the page contained the LRRH script segmented by chapter and color-coded by character. Participants were informed that they could use this document to copy from in the event that they want to follow the given script.

4.2 Methodology

The methodology for this study was approved by Bournemouth University's Ethics department with the approval number 26170.

To answer the proposed research questions for this study, participants must use interactive narrative authoring tools to engage in a typical authoring scenario associated with games writing. The tools included in the study — *Quest*, *Inform*, and *Articy* — were chosen from the clusters discussed earlier as representatives of the broad features and interface paradigms found in the complete selection of considered authoring tools.

The target demographic for this study was students and professionals of digital narrative who have an interest in creative writing, particularly for the application of video games writing. Participants were not expected to be experts in writing, but it was expected that they are competent users of computers. Both student and academic participants were sourced from related courses and disciplines from Bournemouth University, Southampton University, and one from Arts University Bournemouth. A total of 21 participants were involved in the study, consisting of 14 students and seven academics. An incentive was offered to participants to encourage participation and reward their contribution. Any participants that completed the study were optionally entered into a prize draw with a chance to win one of three £30 Amazon vouchers. The winners were randomly selected after all data was processed and participant performance was not taken into account.

As part of the preparation, each participant was assigned one of the three tools and was shown the corresponding tutorial video for their assigned authoring tool. Participants were not timed when watching the video so that they could pause and rewind accordingly. This ensures that all participants had at least a base level of knowledge required to complete the study. Accompanying project files from the training videos were optionally given for participants to explore before the study proper, but none wished to do so. The training video was also allowed to be referenced again during the main part of the study and counted as accessing a source of documentation.

The starting template that participants received implemented the script as earlier discussed. However, the second chapter, where LRRH meets the wolf, had been removed and replaced with empty placeholder content. Instruction was given in the edited templates for what (but not how) the original unedited script had done to fulfill this now missing section to serve as guidance. The challenge for participants then became filling in the missing content between the first and third chapters. The second chapter was chosen as it includes nonlinear pathways, dialogue, and quests, which provides enough variation for participants to be creative. The first chapter is comparatively linear, which may have produced less interesting results, and the third chapter involves complexities that may have been difficult for participants (especially those new to the software) to implement given time limitations.

4.2.1 Participant Protocol

A single session for one participant took around one hour in total to complete. Each participant was randomly assigned one of the three tools. Due to uncertainty with potential participant counts, it was decided that simply cycling through the tools will provide the most even distribution, as providing the participant count was a multiple of three, all tools would be evenly chosen. This is in comparison to a technique such as block randomization, which for a larger sample size would have been the preferred option. Before the study began, participants were given an information sheet that describes the purpose of the study, the expectations, incentives, and so on. The participant could

ask any questions about the procedure here. To begin, the participant was given the documentation page described as described earlier. They were given as much time as desired to read the information about the study, familiarize themselves with the listing of tasks, and to ask any questions prior to their authoring session starting. The participant then watched, at their leisure, their corresponding training video and was offered the opportunity to explore the respective sample project built in the training video. Following the training, the participant was then given a hard limit of 30 minutes to complete the missing section of the template story in their assigned authoring tool. Participants were allowed to finish early if they consider their work complete. Participants were also allowed to ask questions during their authoring session, which were noted for analysis. While participants were encouraged to link the first and third chapters together following the chapter's tasks, they were informed that these tasks are guidelines and that they have complete creative freedom as long as it remains within context of the story. This decision was made rather than having a strict set of tasks that must be followed to the letter to encourage a more natural authoring environment. It also provides structure for those that may struggle to come up with ideas under test conditions. Finally, the participant was interviewed after they have completed their authoring session.

4.2.2 Data Gathering

Data was gathered from multiple sources to maximize the output of the study. Each participant was assigned an incremental unique identifier upon beginning the study which was used to collate all data gathered about them during the study and served as a way of retaining the grouping once anonymization had taken place.

While participants were implementing their story, observational notes were taken, focusing primarily on building a profile of the participant to aid in the post-study interviews. This included highlighting moments of obvious struggle and frustration, as well as periods of adept use of the authoring tools.

The screen of participants was also recorded along with audio capture. This data was gathered to allow for a post-study analysis of the actions and usage of the authoring tools. Participants were also encouraged to speak aloud during their authoring process as to give more insight into their decisions. Unfortunately, few participants did this as they felt uncomfortable doing so.

Following the study, the participant filled in a short demographic survey which served as a way of grouping participants in the analysis stage. A complete listing of the questions can be found in Appendix B.3. After the demographics questionnaire, a semi-structured interview was conducted with each participant to determine their experience and opinions with the authoring tool, identifying features that they believe hindered or aided their authoring experience, and any modifications to existing features or addition of new features that would have made their authoring experience easier. A complete listing of the questions can be found in Appendix B.4. The questions were firstly

asked in order, and then any significant points from the observational notes were raised with the participant, resulting in richer information for each question where points were available. The audio for this interview was recorded for later transcription.

Information collected about participants was kept in regulation with Bournemouth University's Data Protection Legislation. Data were anonymized and transcripts were written as soon as all participants had been through the study process. Participants who entered the incentive prize draw had their emails stored privately and were removed once the draw had taken place and the winners had been contacted.

4.2.3 Limitations

While efforts were taken to ensure that this study was fair, it is still important to make note of the limitations and potential biases of the methodology.

Authoring tools for the study were chosen as representatives of the wider selection, best maximizing the difference in user interface paradigms. All of the tools represent key different approaches to enabling the authorship of interactive narrative but also share some aspects in common. For example, *Quest* and *Inform*, while from an authoring perspective provide vastly differing experiences, both use a room-centric 'adventure game' style model of narrative, whereas *Articy* does not. This means that the tropes of adventure games, such as the use of 'rooms' and command-based interactions, will be more prominent in two of the three software packages. It is therefore expected that while *Quest* and *Inform* have different authoring experiences, they are possibly more similar in overall writing style to each other than they are to *Articy*.

The original eighth participant was removed from the data and replaced with another using the same tool. With the original participant, there were misunderstandings due to a substantial language barrier (the participant was not a native English speaker) which also prevented accurate gathering of verbal data.

4.3 Analysis

Of the 21 participants, all but three participants (one student, two academics) had no previous knowledge of their assigned tool, with one academic reporting moderate knowledge and the other two reporting little knowledge. Most participants had little to no knowledge of similar tools, with three reporting moderate knowledge (one student, two academics) and two reporting plenty of knowledge (one student, one academic). Five academics reported moderate knowledge of interactive digital narrative, with the remaining two declaring plenty of knowledge and little knowledge respectively. Students were more varied with five reporting little knowledge, four reporting none, three reporting moderate, and two reporting plenty. All but three participants thought that the training video was satisfactory, and those that didn't expressed that they wanted a longer, more detailed training session. However, the video presented to them was condensed and limited in the

interest of participant time and instead served as an introduction rather than a course. Only five people considered their own story complete within the time allowed. There is little variation of this between tools, other than *Inform* having one extra incomplete story versus the other two which each had five. Twelve of the participants had a functioning story where 'functioning' refers to being able to play through the story from start to end without error, as intended by the author. *Quest* has one of seven that didn't work due to incorrectly configured objects, which suggests that it is relatively easy to make a functioning story in this tool. *Inform* had a mixed rate of success with two compiling properly, three compiling but having syntactical errors resulting in incorrect behavior, and two that didn't compile at all due to errors. *Articy* had three functional and four nonfunctional stories, with all four having some form of missing connections. Of those four, two had missing connections at the beginning and end of the story, and the other two are generally unfinished along with syntactical errors.

Workflow

Each of the screen recordings was manually annotated with timestamps and indicators of a given action that a participant did, along with a description of what took place. The actions themselves were enumerated based on the observed behavior of participants rather than being defined beforehand. A single action represents a vignette of the larger authoring experience. It was decided to only include the occurrence of a given action rather than to record its duration due to variability and noise when doing so. For instance, a participant stopping to think about what to write next shouldn't extend the time of a previously unfinished action, and this cannot be consistently determined with any reasonable degree of accuracy. Each action has an assigned letter, a categorizing description, and where applicable a program-specific instruction that encapsulated what activity should and should not be considered as a particular action taking place. For example, the World Building action is described as "Present text to the player that describes and conveys the state of the world", and if a participant had performed something that fell within this description, then a corresponding timestamped note would be made on their screen recording. All of the actions and their descriptions can be found in Appendix B.5.

Figure 4.4 contains a stacked bar chart of the individual action frequencies for all participants colored by authoring tool, showing the total occurrence of actions across the entire study as well as the distribution of actions between the tools. The total number of tasks across all relevant participants for *Articy* was 204, for *Inform* was 191, and for *Quest* was 192. While detailed investigation beyond the scope of this study would be required to determine the relationship between different interface paradigms and the pace or frequency of actions, the closeness of these totals tentatively suggests that authors worked at a similar pace in all three authoring environments.

Additionally, the trends of the average rate of actions taken per minute in each authoring tool are also close, further showing that one of the selected authoring tools was

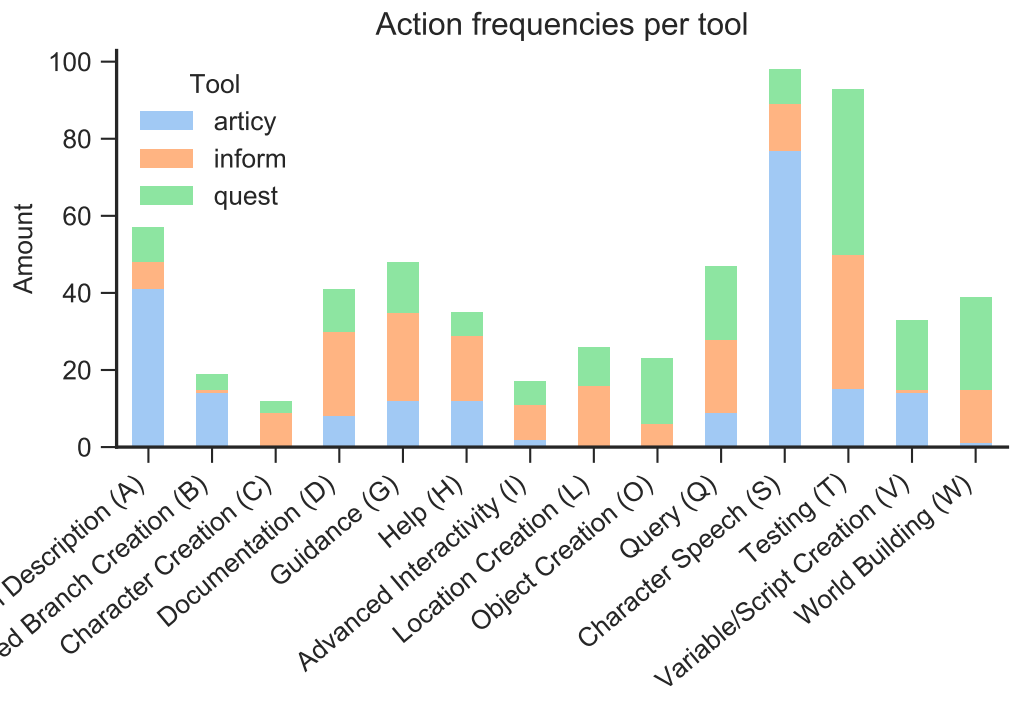
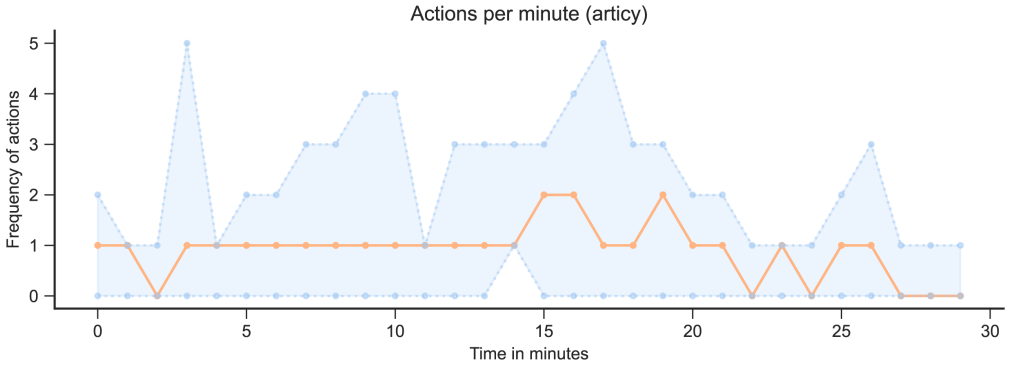


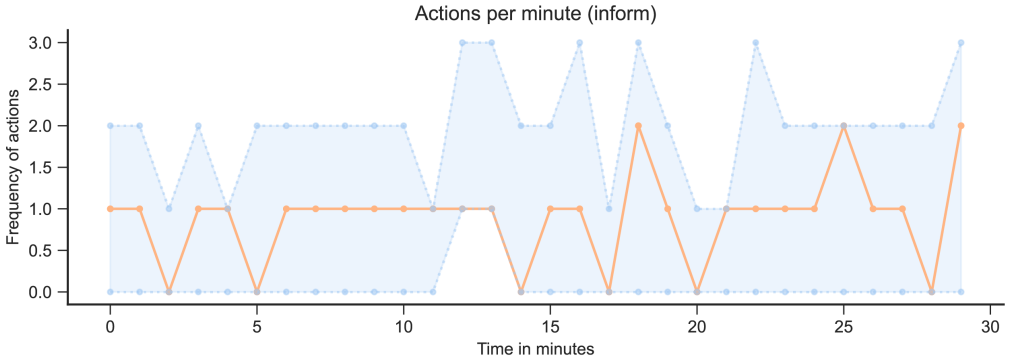
Fig. 4.4 Action task frequencies for all participants stacked by authoring tool.

not unbalanced to consistently generate more or fewer data points than the others. This is demonstrated in Figure 4.5 with *Articy* being displayed in 4.5a, *Inform* in 4.5b, *Quest* in 4.5c, and all tools combined in 4.5d. In these diagrams, the horizontal axis represents the time elapsed in minutes and the vertical axis represents the frequencies of actions within those minutes. For example, data points at minute 5 represent the frequency of actions taken with timestamps taking part in the fifth minute of the participant's authoring session. The blue shaded area of the graph represents the minimum and maximum number of actions taken within the corresponding minute by any participant using the given authoring tool. The orange center line on the graph represents the median number of actions taken within the corresponding minute by all participants using the given authoring tool. Of the three graphs representing individual authoring tools, we can see that the median typically centers around a value of one with short deviations only by a value of one, which is confirmed in the joint graph where the median is almost consistently a value of one. This means that between all authoring tools, the median number of tasks was typically equal, with the exception of occasional deviations. Both *Articy* and *Inform* have an action frequency equal to zero a total of six times each with *Quest* having five. Although the positions at which they occur differ between tools, this does suggest that on average none of the tools had any excessive periods of zero-sum actions where the participants were unable to progress, and that those occurring were relatively equal between tools.

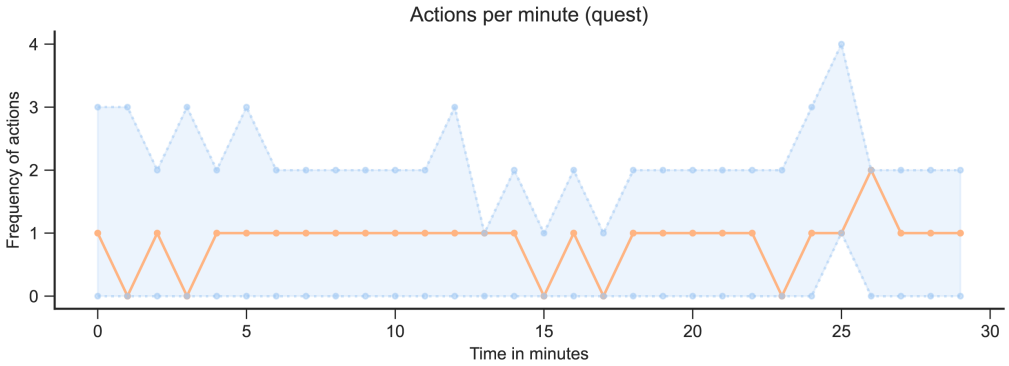
It is possible to determine patterns in the authoring workflow by combining a single participant's annotated action timeline into a single consecutive string and using a sliding



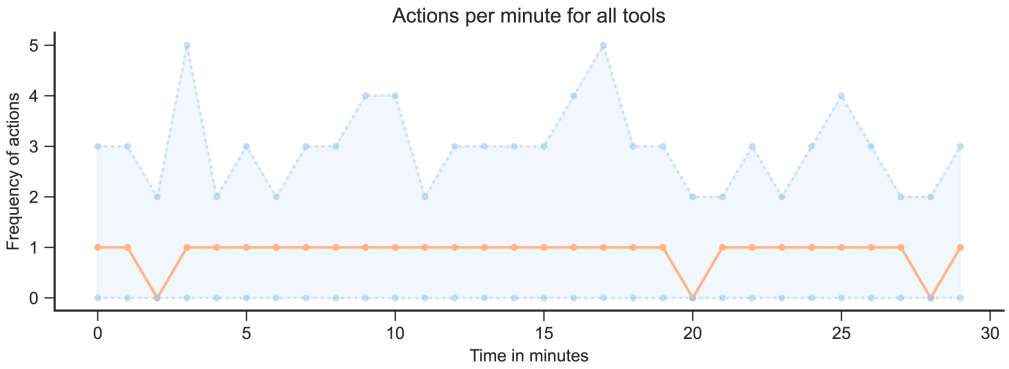
(a) Minimum, median, and maximum actions performed per minute for *Articy*.



(b) Minimum, median, and maximum actions performed per minute for *Inform*.



(c) Minimum, median, and maximum actions performed per minute for *Quest*.



(d) Minimum, median, and maximum actions performed per minute for all tools.

Fig. 4.5 Frequency of actions taken per minute. The shaded area represents the minimum and maximum tasks performed within that minute, and the middle line is the median.

window to generate ngrams. For example, if a participant created a variable (V), authored a branch using the variable (B), and then created two instances of character speech (SS), then the combined string becomes VBSS. These combined strings are referred to as flows. When all participants of a given tool have their flow ngrams combined, it is then possible to determine any repetitive patterns, their frequencies, and how many participants contributed to them. This means that for any given ngram of size n, we can determine the patterns that exist per tool, and we can also cross-reference these patterns to further determine what common patterns exist between the tools. An important point to consider with ngrams is their use of a sliding window which makes them sensitive to long sets of repeating characters. For example, if we have the string 'tttttt', an n of two will result in five sets of 'tt', when actually the full string is a truer representation of the pattern. This can artificially inflate the reported number of occurrences. However, a high number still signifies that this feature is important, just perhaps not as important as it initially appears. A solution to this is to increase n and see if similar patterns continue to emerge. Taking a sliding window and reporting all occurrences does not account for situations where a given pattern with a high count is attributed chiefly to a low number of participants, which can falsely indicate a wider spread of the pattern. For example, if 'tt' is reported 15 times in a given authoring tool, it may appear that repeated testing is frequent throughout the tool, but without knowing how many participants contributed to this count, it is possible that one author contributed to the majority of cases, meaning that the pattern is not as widespread as indicated. This can be mitigated by ensuring that a minimum threshold of unique participants performed a given pattern. For this study, ngrams were run ranging from $n = 2$ to $n = 9$, with a minimum threshold of three participants for an ngram result to be considered. The threshold was set to three as it is the closest floored integer below half of the participant count per tool (i.e., $\lfloor (7/2) \rfloor = 3$).

Bigrams

In *Articy*, there are ten unique bigrams that contain either Action Description (A) or Character Speech (S). Alone, these bigrams are not of significance, but the prominence of these two particular actions does suggest that speech and narration are popular in this particular tool. Of interest are repeated Character Speech (SS) and sequential Action Description followed by Character Speech (AS) and vice versa (SA), which together make up the three most common terms in both frequency and number of contributing participants. These further demonstrate the prominence of Action Description and Character Speech in the authoring tool, and given that explicit dialogue nodes are a feature of the tool, this is expected.

In *Inform*, Documentation (D) is involved in seven different combinations, which shows the commonality of help usage. Given that *Inform* required the most Guidance (G), Help (H), and Query (Q) of all three authoring tools, particularly due to being a domain-specific language rather than a graphical interface, this is expected. Additionally, Testing

(T) is involved in eight different combinations, each with a relatively high number of occurrences, and two repeats of Testing (TT) occurred eight times between four participants, which both suggest that authors test very frequently when using *Inform*. Moreover, sequences of Character Speech followed by Testing (ST) and World Building followed by Testing (WT) with three and four occurrences respectively both split between three participants suggests that testing commonly takes place immediately after introducing a new textual element of visual feedback to the player.

In *Quest*, there are significantly fewer bigrams than in the other two programs, although the reason for this is unclear. Of interest are sequences of Object Creation followed by World Building (OW) and vice versa (WO) which suggests that authors create objects and add descriptions to objects in succession, which was also observed in the study. Testing (T) is present in two combinations and as a repeated pair (TT) occurring 18 times between four participants, which demonstrates that authors frequently test within this authoring tool at a level similar to *Inform*.

The only bigram that is shared between all three programs is repeated Testing (TT). *Articy* had the lowest number of repeated Testing actions totaling four occurrences shared between four participants. All participants found it easy to begin testing but struggled to exit the testing mode, with four requiring explicit help to do so. This difficulty could deter users from launching the built-in testing harness and could partially explain the lower frequency of repeated (and general) Testing. *Inform* has higher cases of repeated testing with eight occurrences shared between four participants. In *Inform*, code is typed in, and the only way to validate whether something functions as expected is to test the story, which explains why this authoring tool sees such high amounts of testing. While five participants did complement the ability to fast track to a specific point of the story when testing, the testing harness does not allow for rewinding of choices made, and therefore if an author wishes to try multiple pathways, they must engage in separate testing sessions, which can explain the increase in repeated testing in this tool. *Quest* also has prominent repeated testing with 18 occurrences shared between four participants, although this is inflated chiefly by one participant who tested more often than others. *Quest* suffers from the same problem as *Inform* in that the only way to verify functionality is to test, and that choices made when testing cannot be undone to test multiple pathways. When adjusting for the inflated rate of repeated testing, *Quest* is comparable to *Inform*, and given that they both create adventure games and have a similar testing setup, this is expected.

Trigrams

In *Articy*, all trigrams contain some permutation of Action Description (A) or Character Speech (S) either together or with other actions. Most appear to be noise with low frequencies shared between the minimum number of participants. However, of interest are Action Description followed by repeated Character Speech (ASS), Action Description surrounded by Character Speech (SAS), repeated Character Speech followed by Action

Description (SSA), and repeated Character Speech (SSS), with the latter occurring 21 times between six participants. This further highlights the prominence and focus of dialogue within this authoring tool.

In *Inform*, only two trigrams are present. The first, Character Speech surrounded by Documentation (DSD), is likely by chance, as there are only three occurrences spread between three participants. The second, Documentation followed by repeated Testing (DTT), is of equal value and is not prominent enough to base claims upon.

Quest also only has two trigrams, although they are more insightful. The first, repeated Testing (TTT), occurs 11 times between three participants, which further demonstrates the prominence of testing in this tool but also highlights the artificial inflation. The second, Variable/Script Creation followed by repeated Testing (VTT), while only occurring five times between three participants, does tentatively suggest that participants tested after creating scripts more than once, perhaps to test the impact of the script when conditional statements were present, for example.

There are no shared trigrams between all three authoring tools.

Quadgrams and Beyond

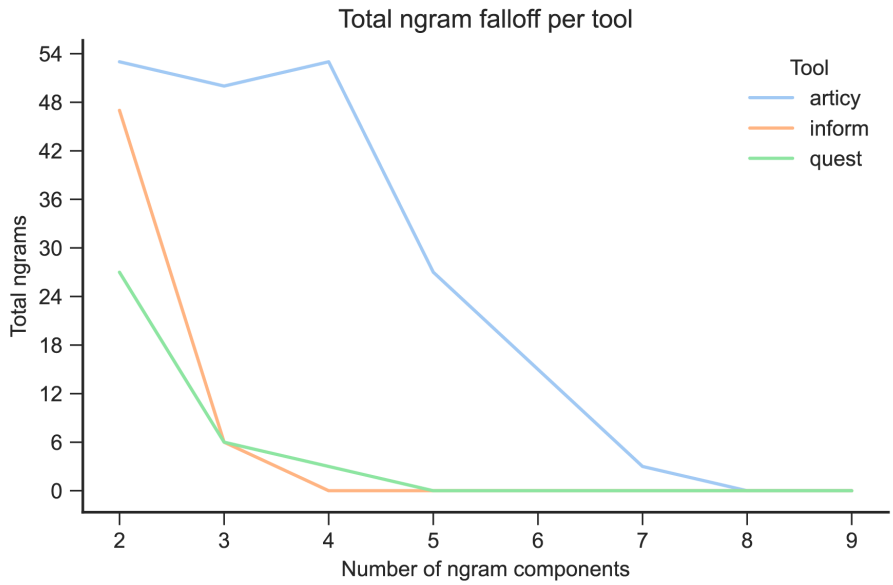


Fig. 4.6 Falloff of returned ngrams as n increases for all authoring tools.

When $n = 4$, *Inform* has no ngrams occur. *Quest*, on the other hand, has repeated Testing (TTTT) occurring six times between three participants, which further demonstrates the prominence of Testing within this authoring tool. With *Articy*, the number of ngrams increases, but are mostly permutations of Action Description (A) and Character Speech (S), each with relatively low frequencies. Permutations of interest are ASSS and SSSS, which occur eight and nine times respectively, both shared between six participants each, which further highlight the prominence of dialogue within the tool. At this point, no ngrams are shared between authoring tools. Moving through $n = 5$ to $n = 7$, only

Articy returns ngrams, all involving permutations of Action Description (A) and Character Speech (S), but with low frequencies and appear to be noise. At $n = 8$ and $n = 9$, no tools return any ngrams. Figure 4.6 shows the falloff of ngrams per tool as n increases.

Qualitative Coding

Participant interviews were firstly transcribed and anonymized in preparation for qualitative coding. The transcripts were then coded using Exploratory Methods (open-ended investigation of data before a more refined coding pass) in an Initial Coding pass by means of Descriptive Coding (assigning topics to passages of text), as described by Saldaña [185]. If a block of text demonstrated a previously unencountered concept, then a new code was created to encapsulate that point. If instead the block fell under an existing code criterion, then it was coded under the existing code as to tally various commonalities between participants. These blocks were coded using Simultaneous Coding in that the same block, or overlaps between blocks, can be present in multiple codes if necessary.

The coding information was stored in a spreadsheet, with each code being given a unique identifier, a summarizing title, and a list of criteria to satisfy membership. Alongside each code was also the total number of occurrences throughout all transcripts, of which a single participant could contribute to multiple times by matching a code's criteria more than once, listed for each authoring tool individually as well as a combined total. Additionally, the total number of unique participants involved in each code was also stored, again per authoring tool as well as combined. Within participant transcripts, manual annotations were added to declare blocks of text a part of a code by inserting the unique identifier of the matching code. Table 4.1 shows a summary of the code titles alongside the occurrence and participant figures for each, sorted in descending order by the total number of occurrences of each code. For more detail of the codes including their criterion, please refer to Appendix B.6.

4.4 Principles

After detailed analysis of the data gathered from the study as discussed above, a set of guiding principles for consideration during the design and development of authoring tools supporting interactive narrative were derived. This involved taking the coded qualitative data from the interviews and cross-referencing it with both the observational data taken within participant sessions and the overall workflow analysis as outlined above to identify trends between the participants based on which authoring tool was being used. The principles were then formed by joining together related trends. These principles serve as indicators for the potential impact of a given design decision on the way in which authors work. Each participant was assigned a unique ID starting at 1. Where participants are mentioned below, they are referred to as P followed by their ID, such as P4 for the participant with an ID of 4.

Table 4.1 Code titles alongside the number of occurrences and number of unique participants per tool (*A=Articy*, *I=Inform*, *Q=Quest*). See Appendix B.6 for more details.

	Occurrences			Participants		
Summarizing Title	A	I	Q	A	I	Q
Unclear UI elements cause confusion	10	0	12	6	0	7
Inclusion of an intelligent assistant when writing code	0	8	0	0	5	0
The graph system is good/helpful	7	2	0	7	2	0
Elements of the interface are too small	1	1	5	1	1	5
Assistance to return to a specific state during testing	0	5	2	0	5	2
Using the tool is easy having gotten used to it	2	3	1	2	3	1
Unclear terminology caused confusion	3	2	1	3	2	1
Error messages are difficult to understand	0	5	1	0	5	1
Desire of a graph system when it wasn't present	0	3	3	0	3	3
Variables are useful	2	0	3	2	0	3
Support for more advanced dialogue functionality than built-in	1	3	1	1	3	1
Difficulty in use of built-in syntax	0	5	0	0	5	0
Populating the world with objects is easy	0	2	2	0	2	2
Creation of the world structure is simple	0	2	1	0	2	1
Hierarchies are useful	1	0	2	1	0	2
Variables should be easier to handle	2	0	1	2	0	1
Creation of branches is easy	2	0	0	2	0	0
Intuitive syntax	0	2	0	0	2	0
The overall interface is overwhelming	0	0	2	0	0	2
Editing content is easy	1	1	0	1	1	0
Shortcuts were useful to create layouts	2	0	0	2	0	0
Support for more advanced game-centric features than built-in	0	1	0	0	1	0
Good use of context menus	1	0	0	1	0	0
Intelligent syntax assistance is good	1	0	0	1	0	0

4.4.1 Metaphor Testing

Interfaces that use a visual metaphor to represent story structure and connectedness will result in less testing of non-complex stories.

All seven participants using *Articy* used the built-in testing harness at least one time with the median being two. P4 tested five times, but this was an exception to the norm, as they ran an initial test before writing and experimented with a more complex structure involving 'Jump' nodes. *Articy's* main interface chiefly relies upon a spatial hypertext graph, which is already understood to visually provide an overview of underlying structure and connectedness [18, 190], and is a commonly encountered paradigm within the authoring tool space. In this context, the spatial hypertext graph visually represents the underlying structure and connectedness of the story content, and when authors are given freedom to arrange nodes how they wish, they often do so in a manner that visually represents the underlying structure of the story [111], such as in an arborescent form. The inherent structural visualization properties of spatial hypertext graphs and their tendency to be used as a method for representing and understanding structure can in part explain the low frequency of testing observed in *Articy*. The nodes within *Articy's* graph display attributes such as character profiles and spoken text strings, and have clear connectivity between them, meaning that they can serve as a lightweight replacement for simple testing, as authors can see what would occur in the story without explicitly using the

built-in testing harness by visually following the chain of nodes. Given this situation, authors would not require use of the built-in testing harness to validate structural behavior unless such behavior could not be inferred from connectivity and exposed attributes alone. In the study, all but one of the participants using *Articy* were observed periodically scrolling back through their story graph to remind themselves of the proceeding content and structure before continuing to add more nodes without explicit use of the built-in testing harness, with a median of two occurrences per participant and a maximum of four. Moreover, as shown by the *Desire of a graph system when it wasn't present* code, six participants split evenly between *Inform* and *Quest* expressed the desire for an interactive graph to be present for the creation and visualization of connections. For instance, P9 describes wanting "an action map, some form of flow chart, which makes it a lot easier to know exactly where [the connections] are going". P21 issues a similar statement, stating that they would like their story "represented in a sort of mind map" so that they could "see different branches going away [from other content]".

The way in which authors primarily write with *Inform* is through its augmented text editor. Consequently, *Inform* offers little in the way of visualizing the story structure beyond the inclusion of a static map that updates only upon successful compilation, which four of the seven participants made use of to understand the actual generated structure from their instructive text. P11 specifically described the map feature as "very useful to know where everything is", particularly given the textual nature of the authoring process in this tool. P14 offered a similar sentiment, expressing that "the map helped a lot, it helped me to see how things weren't connected when they should have been and helped me to solve problems". Beyond this, there is no other way to check if something will function as expected without using the built-in testing harness, as code is compiled and does not execute sequentially in the order in which it is typed, and therefore cannot be read line-by-line to consistently predict what may happen. This reliance upon the built-in testing harness due to a lack of sufficient visualization can in part explain the higher total testing count found in *Inform*. All participants using the tool used the built-in testing harness at least one time with the median being five. P14 tested 12 times, but this was an exception as not only did they frequently test more than other participants, but also repeatedly tested when encountering problems with their story.

Quest, relying upon multifaceted user controls rather than a graph or an augmented text editor, lies visually in between the other two authoring tools. Although *Quest* does not have a static graph to visualize the complete room structure as seen in *Inform*, it does allow the author to link individual rooms to one another with buttons arranged to mimic the directionality of the connections (e.g., the buttons that represent the cardinal and intercardinal directions are laid out like a compass), which provides minimal and isolated visual feedback to authors in the context of the room currently being edited. Moreover, *Quest* also features an abstracted scripting system that can be read sequentially to understand the dynamic behavior of given actions within the story, although this is in

isolation to the individual script being edited. Despite these visual features that could be used in a limited way to understand the story without using the built-in testing harness, and that two participants didn't test their story with no reason given, the accounts of testing in *Quest* remain high on average with a median of five occurrences and a maximum of 17, although this is an isolated case. If we account for the outliers in both *Quest* and *Inform* by removing them from the sum, recalculating the median, and then adding them back as the new median, *Quest* is still higher with a total of 31 instances of testing versus *Inform's* count of 27. This further suggests that the visual aids for understanding structure reduce the occurrences of testing, and that visualization in isolation is not sufficient and instead requires contextualizing within a greater scope.

4.4.2 Fast Track Testing

Letting users jump to any state of the story enables more rapid and focused testing sessions.

In *Articy*, starting the built-in testing harness can be achieved by either firstly selecting a node and using a corresponding toolbar button, or by using the context menu of a node and selecting the appropriate menu item, either of which begins the testing session at the node in question rather than from the beginning each time. All participants used this feature to test from arbitrary locations within their story every time, typically starting at a location nearby the content or interaction that they wished to preview. *Articy* records and stores the history of choices that an author makes in a single testing session as a 'Journey', which can be later loaded to similarly fast forward the author to a specific part of the story. However, no participants found it necessary to load a Journey, perhaps in part due to the small scope of the stories. Alternatively, the author can use a list view of the history of their choices while testing to revert to any previous state within that session to test alternative choices, although this was only explored by P13.

When launching the built-in testing harness in *Inform*, authors are always placed at the beginning of the story, which can become problematic with repeated testing, particularly as stories grow in size. To alleviate this, *Inform* includes a 'Skein' feature which builds a hierarchical tree of every action that an author performs across all testing sessions and displays it visually within the interface. The nodes of the tree can be interacted with to fast forward the author to a specific part of the story by automatically repeating all actions until that node, much alike the Journey feature found in *Articy*. Six of the seven participants using *Inform* almost exclusively used Skein to test from arbitrary locations in the story, typically starting at a location nearby the content or interaction that they wished to preview, the exception being when playing through the whole story from the start to understand the content. Five of the seven participants explicitly complimented the Skein feature in their interviews. P5, for instance, described it as "useful" and explained that it "made it a lot quicker . . . to get where I'd [previously] gotten to and then work out where I was going from there". P11 similarly explained how it's useful to "get to the point

that you need to test before losing your mind repeating the same things over and over again". P20 described the tool as a "really nice testing tool" and that it helped them with "replaying up to a certain point", concluding by highlighting the importance of being able to return to a previous state. *Inform* also supports an advanced syntactical `TEST` command which enables testing of specific sequences at arbitrary points of the story, although no participants explored this.

Quest features a 'Walkthrough' mode which is analogous to *Inform's* Skein, although it does not provide an interactive tree of options and must be manually created, and is instead instigated with a toolbar option. In this study, a single Walkthrough was provided in the starting project that if activated would forward the participant to the removed chapter that they would be editing. No participants attempted to create their own Workflows. Of the five participants that made use of the built-in testing harness, four used the provided Walkthrough, with three of them beginning their testing sessions with the Walkthrough on all but one of their unique testing sessions. The other participant explained in the interview that while they tried the Walkthrough mode, they were uncertain of its function and avoided it in subsequent testing sessions. P21, on the other hand, described Walkthroughs as "very easy", explaining that the feature saved them from repeatedly traversing to the same state each time they wanted to test something, and further praising the feature for showing the steps that the Walkthrough takes within the interface for clarity. As with *Inform's* Skein, this provides further evidence of the desire to skip unnecessary content when testing specific parts of a story.

An additional feature unique to *Quest* is clickable links on keywords while interacting with the built-in testing harness, which helps authors traverse to a given state quicker than typing and also helps them to explore commands that may be otherwise unknown. All five of the participants that used the built-in testing harness made substantial use of this feature by incorporating it into almost every unique testing session that they ran, which further highlights the desire of authors to achieve a desired state of a story quickly.

These points across all three authoring tools highlight the importance of allowing authors to return to a given state of the story with efficiency to avoid repetitive traversal to achieve a desired state, which over time can become frustrating. Moreover, having to traverse from the beginning of a story risks the author losing focus of the intent of the testing session, having spent more time than necessary to achieve the state desired for the testing session proper to begin.

4.4.3 Structure

Interfaces that use a visual metaphor to represent story structure and connectedness are intuitive for management of an author's own story structure.

The user interface of *Articy* centers around a spatial hypertext graph, which of the three authoring tools makes the best effort to visually mimic the story structure in a way

that is already understood to provide a visual overview of underlying structure and connectedness [18, 190]. As part of the *The graph system is good/helpful* code, all seven participants spoke positively about their experience with the graph system. For example, P4 describes the graph as "really nicely designed" specifically due to the interactions and visual feedback in terms of connectedness of the story content. P10 described the graph as "intuitive" and highlighted that the visual feedback of the node attributes and the simple methods of visually connecting content together were key to this interpretation. The spatial hypertext graph in *Articy* gives authors the freedom to arrange their nodal content however they wish. Prior studies have shown that with this kind of setup, authors will typically organize their nodal content in such a way that it visually represents the underlying structure of the story [111], such as in an arborescent form. In the case of this study, this remains true, as all seven participants laid out their stories from left to right in an arborescent form, with the horizontal axis representing the progression of time and the vertical axis representing variation in the story such as with multiple branching pathways. This behavior observed in this study as well as existing evidence from other studies strongly suggests that spatial hypertext graphs, especially in the context of interactive storytelling, support intuitive organization and management of one's own story structure.

As *Inform's* primary method of editing content is through an augmented text editor, there is little in the way of physical content organization other than segmenting text by headings or by otherwise neighboring text based on some relationship such as in-game location or involvement in a mechanic. However, during the study, no participants demonstrated thought toward organization of textual content. In *Quest*, content organization can be achieved by appropriately nesting objects within one another, essentially replicating a hierarchical folder structure. All seven participants using *Quest* were observed nesting interactive objects (items within the game world) within room objects (locations), although this is the default encouraged behavior of *Quest* when creating new interactive objects by automatically suggesting a parent object. No participants were observed organizing their content beyond this. To this regard, P15 stressed that this method of organization, where everything is a form of object, "put me off at the beginning . . . [which is] why I was confused", adding that the hierarchical overview of the nested relationships of objects was difficult to read, particularly for level of indentation, which is how object nesting is represented. P21 echoed a similar statement, expressing that the authoring tool "felt like data entry" and that they would have to "go back and forth [between objects] to figure out how different things connect", concluding that they would prefer something visual so that they could "see [my] different objects more clearly". In both *Inform* and *Quest*, organization of story content appears to be of more difficulty than in *Articy*, and the observations and interviews reflect this as described above.

4.4.4 Experimentation

Interfaces that use a visual metaphor to represent story structure and connectedness enable easier experimentation of structure and connectedness.

As *Articy* chiefly relies upon a spatial hypertext graph, it represents the story content as nodes and the connections between them as joining lines, which is already understood to provide a visual overview of the underlying structure and connectedness [18, 190]. It is possible that the clarity brought by a spatial hypertext graph in a storytelling context provides an understandable framework within which authors can easily experiment with structural configurations, as objects and their connections can be easily created, manipulated, or destroyed without hindrance. In the study, five of the seven participants using *Articy* were observed reconfiguring an already completed structure at least once with one participant altering their completed structures three times. On the other hand, only two participants using *Inform* and none using *Quest* were observed altering completed structures (in this case, the linkage of locations or logic of movement between locations). *Quest* allows the author to manage linkage between individual locations in an interface with buttons arranged to mimic the directionality of the connections (e.g., the buttons that represent the cardinal and intercardinal directions are laid out like a compass), which provides minimal and isolated visual feedback to authors in the context of the room currently being edited. *Inform*, on the other hand, requires the author to create locations and link them together through declarative statements in the story's source text, providing a static map that updates upon a successful compilation that can be used to validate the connectedness. This visual disconnect can be problematic and result in authors losing track of connectedness, which inherently impacts the ability to experiment with structure. For example, two participants using *Inform* had renamed the provided location *MyRoom* in favor of their own name but did not successfully rename all instances in the text, therefore *MyRoom* was still created and their renamed room was unknowingly disconnected from the rest of the story. The fewer instances of structural experimenting in these two authoring tools could be attributed in part to the lack of visual coherence of the story structure and connectedness between locations, making it challenging to effectively identify the overall relationship between locations within the story. Indeed, some participants as part of the *Desire of a graph system when it wasn't present* code split evenly between both authoring tools expressed desire for an interactive graph for the creation and visualization of connections. For instance, P9 describes wanting "an action map, some form of flow chart, which makes it a lot easier to know exactly where [the connections] are going". P21 similarly describes that they would like their story "represented in a sort of mind map" so that they could "see different branches going away [from other content]".

4.4.5 Branching

Interfaces that use a visual metaphor to represent story structure and connectedness enable easier creation and management of branches.

In *Articy*, all participants created at least one branch with the median being two and the maximum being three for two participants. *Articy* centers its user interactions for story creation chiefly around a spatial hypertext graph, which between the three authoring tools best visually mimics the story structure in a way that is already understood to provide clarity to underlying structure and connectedness [18, 190]. It is possible that the interaction and visual affordances that such graphs provide in the context of storytelling aid in the ease of creating branching structures. P1, who created three branches in their story, explained how they found "creating branching dialogues with choices quite easy" using the graph system. Although other participants did not explicitly discuss the influence of the graph system upon branch creation, we can infer that given their successful creation of branches without the need for help or guidance, that it at least adequately supports the task.

In contrast, four participants using *Quest* created a single branch each and only one participant created a single branch in *Inform*. We must consider the possibility that the difference in frequency of branching comes from the types of stories that authors were creating despite being based upon the same script. Given that both *Inform* and *Quest* implement an adventure game adaptation of the script yet both differ greatly in total branch count, this is unlikely to be a significant influence. In *Inform*, branches are created with declarative statements in the story's source text, requiring, at a minimum, several location declarations and several further declarations linking them together. Although *Inform* provides a static map to preview linkage between locations, which four of the seven participants made use of to understand connectivity, this provides only limited insight into branches and does not show dynamic branches such as those that block traversal based on state checks. *Articy*, on the other hand, visualizes dynamic branches by distinguishing the connection from others with a change in color. Due to the potentially complex nature of location connections and dynamic blocking of movement, the only reliable method of verifying branches in *Inform* is to explicitly test them. This lack of visual representation and comparative complexity of creating such structures could explain why *Inform* sees fewer branches than the other authoring tools. *Quest's* creation and display of branches are less visual than *Articy's* graph but less abstract than declarative statements found in *Inform*. Branches that stem from room connectivity are created in an interface that arranges buttons to mimic the directionality of connections, although this is in isolation of the source room in question and lacks context. Conditional branches are instead created using an abstracted scripting system, although this is again edited in isolation of the object hosting the script in question. These methods of creating branches are more visual than those found in *Inform*, providing a similar level of visualization. It is

instead possible that the enhanced tangibility over *Inform* increased the ease of creating and managing branches, but that the less visual and interactive nature of the interface in comparison to *Articy* limited it from becoming as frequent.

The evidence discussed above suggests that the reduced visualizations and increase in difficulty in creating branches as found in *Inform* and *Quest* contributed to the reduced frequency of branches with those systems. Moreover, as shown by the *Desire of a graph system when it wasn't present* code, six participants split evenly between *Inform* and *Quest* expressed the desire for an interactive graph to be present for the creation and visualization of connections. Although only P21 explicitly mentions the motivation being in part to visualize the created branches, we can infer that the addition of a visual map would aid authors in at least understanding their branches, particularly as others such as P9 had requested the graph to "[make] it a lot easier to know exactly where [the connections] are going". Note that this principle differs from the *Principle of Structure* outlined above in that describes explicit divergent structures and the authoring process around them as opposed to the latter which concerns itself with representing structure convenient for the author in a way that doesn't impact the actual narrative flow.

4.4.6 Contented Authoring

Users that become content with a simple method of authoring are unlikely to explore more advanced features, even if it would enhance their work.

All participants using *Articy* chiefly used Dialogue nodes with simple connectivity, typically creating long strings of sequential dialogues emulating conversation. While *Articy* does excel at discourse, it is not enforced and features much greater functionality just as readily available in the interface. While six participants experimented with including variables within their story to varying degrees of success, only a single participant explored more advanced configurations involving Jump and Hub nodes. It is possible that the simple method of authoring by connecting sequential dialogues satisfied the creative needs of a majority of authors in this study and that consequently they did not feel the desire to explore nor query about the more advanced features within the software. To some extent, the user interface of *Articy* suggests a default method of authoring which could influence this behavior, as nodes must be connected left to right in order, and when nested the entry and exit points are likewise positioned on the left and right extremes, which could suggest to the author that sequential patterns are expected.

In *Quest*, the interface is less suggestive of a 'default' method of authoring in terms of connectivity and structure other than the involvement of rooms and objects, as constrained by the genre of story that the tool creates. *Inform*, revolving around an augmented text editor, provides an even lesser influence, given that participants are only exposed to semi-structured text without any interface elements suggesting a format. Likely due to a lack of a clear suggestive authoring method, participants were observed to explore and

experiment from the start of their session, and as a result, experimented with advanced features more often. In *Articy*, only one participant explored advanced features, but in *Quest*, three participants shared six instances of advanced authoring, and in *Inform*, four participants shared nine instances. Moreover, participants using both *Quest* and *Inform* typically required more Guidance, Help, Queried more frequently, and accessed the provided Documentation more often (see Figure 4.4), which further suggests that authors had to explore more complex authoring patterns in these environments.

Providing a simple and understandable authoring interface aids users as it reduces complexity and makes the program more accessible. However, the above discussion shows the possibility that simplified methods of authoring could result in authors becoming content with a limited form of authoring and therefore be less likely to explore more advanced features available within the program. Moreover, it also implies that having a suggestive 'default' method of authoring, even if not enforced, could inadvertently limit creativity in that authors become unknowingly enclosed within this format.

4.5 Discussion & Comparison to Similar Works

The above principles are derived from empirical evidence. They are intended to serve as guidance for the development and design of future authoring tools so that we can make more informed decisions about our interfaces and UX choices. They also serve as an initial vocabulary to aid further discussion within this space of the UX issues facing authoring tool design. This listing is referred to as 'principles' rather than heuristics as they represent observations of user behavior in response to UX design choices rather than a specific heuristic-based best practices listing or evaluation framework.

In comparison to established usability and interaction design heuristics such as the works of Nielsen [163], these principles differ in that they aid us in understanding the impact of design choices on domain-specific processes rather than providing general rules of thumb applicable to a broad range of interaction design scenarios. Nielsen's heuristics, although derived from computing and widely applied to graphical user interfaces, are in fact broad rules of thumb for what makes interaction design in general, regardless of the medium, easy or difficult to use. Due to the nonspecific descriptions of these heuristics, the substance of the definitions must be interpreted for each specific application. For example, if we consider Nielsen's *Visibility of System Status* heuristic, which encourages designs to inform users of system status through appropriate feedback, the concept of visibility must be interpreted per application. Visibility in mobile software could refer to ensuring the user knows that content is being loaded after a button was tapped, whereas for a voice assistant, visibility may refer to informing the user of state changes through auditory feedback such as sound effects or spoken words. Nielsen also provides an example of this heuristic applied to physical maps with the inclusion of a "You Are Here" icon, where visibility of status refers to the reader's location within the physical environment

represented on the map. Consequently, applying the heuristics in any given scenario produces benefits at the lowest common denominator across all interaction designs, which is the lowering of cognitive friction and the increase in usability. However, following these rules of thumb do not then describe the intricacies of a domain-specific process and the response that such a process may generate, as that is not, by definition, shared between all forms of interaction design. To further illustrate the differences between the principles identified in this study and established heuristics, let us consider a hypothetical design for a new authoring tool. If, when planning the design, developers consider the principles identified in this study, then they will gain insight into the potential response of authors given a certain kind of feature. For example, by considering the *Principle of Metaphor Testing*, developers can work to visually mimic story structure in their design, with the impact being a reduction of reliance upon a dedicated testing harness due to the enhanced visual feedback as described earlier. This takes UX guidance beyond the general principles presented elsewhere towards domain-specific guidance for authoring tools backed by evidence. Existing heuristics for best practices should continue to be used in conjunction with the principles, in that the individual designs that developers come up with remain candidates for refinement of usability. On the other hand, if developers follow established heuristics in isolation, the designs that they come up with may follow best practices, but developers will remain without information as to how authors may respond to such design choices, which is what these principles aim to provide. Further demonstrations for the need of domain-specific UX principles and heuristics can be found elsewhere in the creative and cultural fields connected with technology in Desurvire's work on HEP [53], GAP [57], PLAY [56], and VR PLAY [54], all of which expanded upon general usability heuristics (including those by Nielsen) and other established guidelines to create heuristics lists specific to game playability (HEP), game approachability (GAP), game usability (PLAY), and best practices for virtual reality experiences (VR PLAY). Similarly, Korhonen developed his own heuristics and principles for effective UX in interactive mobile entertainment [114], in which he both subsumed and expanded upon the more generic principles of Nielsen. In both cases, we can see that general UX principles such as Nielsen's *Visibility of System Status* are refined with domain-specific principles such as "Indicators Are Visible" (Korhonen) or "Status Score Indicators Are Seamless, Obvious, Available and Do Not Interfere With Game Play" (Desurvire). The principles identified in this study continue in this tradition providing principles for authoring tools. They do not merely encourage understandable feedback as per Nielsen's *Visibility of System Status*, but specify the importance of visual story metaphors and narrative structure, amongst others. The existence of such works, including these contributed principles, do not replace or otherwise diminish the value of general usability heuristics, but do show the need for more granularity when dealing with particular domains, as general guidelines are inherently limited in scope due to their abstract nature. In developing domain-specific methods and principles, this is following Greenberg's [90] call to not

dogmatically follow established general UX understanding, but to instead formulate methods and understanding specific to the applications in question.

The principles should also be considered in the context of using an adaptation of LRRH as a story template. The critical element in this adaptation was the presence of tropes that are typically found in existing works within the same domain rather than the semantics of the content itself. As mentioned in §4.1.3, this process ensured that participants would interact with the authoring tools in a representative environment for authoring of interactive narrative, that a reasonable effort could be made to complete the task within time while still allowing for substantial interactions with the tools, and to mitigate potential time sinks that may occur by asking participants to rapidly create story material without reference. It is possible that the particular structure that emerged from the LRRH adaptation had an impact upon the way that participants interacted with the authoring tools. The story was there mostly to provide a basis for interacting with the tools, and there is no direct evidence that the story style or premise directly impacted the principles we include, but naturally it remains possible that it influenced outcomes as a methodological context, as with any UX study. However, exploring the specific impact of differing story structures upon authoring tool usage is beyond the scope of this study which instead seeks to establish initial principles for this domain. Similarly, it is possible that removing a different section of the story for participants to complete could have influenced their interactions with the authoring tools. This is why, as described in §4.2, the complexity of each section was considered, the sufficiency of interactions with the authoring tool that this could result in, and the time it would take to reasonably attempt to complete the section given the time restrictions.

4.6 Conclusion

This study set out to identify the impact of different user interface paradigms on the UX of IDN authoring tools. Evaluating authoring tool UX presents significant challenges, as unlike with many short task-focused web applications (which much of contemporary UX practice focuses upon), typical usage is not measured in short interactions that can be completed in minutes but instead in substantial creative endeavors that take months or even years. Consequently, the challenges in evaluating these systems have led to an absence of research exploring the impact of the UX of authoring tool technology despite its importance. This gap has served as the motivation for this study, which is of direct relation to the second research question asked by this thesis.

This study's research question was approached by conducting a user study using a novel bespoke evaluation methodology for IDN authoring tools which addresses the difficulties facing authoring tool evaluation. While this method addresses these challenges, it does introduce some compromises and assumptions, and it is important to recognize the conclusions in context of the constraints of this method. This method involves

authoring tasks that are substantially shorter than full authorship and often did not permit participants to complete fully functional stories and as such may be considered a somewhat stressed UX. It is also possible that the genre and style of the template story may have influenced the UX. The methodology does take steps to mitigate these issues, however, by using informed and representative story features, and presenting the participants with a template to work from such that within the bounds of the study they interact with a range of authoring tool features in a reasonable writing context. Consequently, while it is important that the conclusions are considered within the constraints of the methodological design, it remains suitable, and valuable conclusions can be produced from studies of this kind. In forming bespoke methodologies for this challenging domain such as in this study, we follow Greenberg's call [90] for evaluation methods to be designed for the constraints and context of the applications being studied rather than just applying previous methods established in a different context.

The study was oriented around interviewing authors and observing their behaviors within interactive narrative authoring tools that exhibit varying interface paradigms in order to identify any trends that demonstrate the impact that these differences make upon authoring workflow as defined at the beginning of this chapter. Included was the approach to choosing a subset of representative authoring tools from the wider selection which used hierarchical clustering of various identified features in order to determine groupings, and then individuals from within those groups were chosen. This was done to ensure that the selected subset of authoring tools was a better representation of the global feature space than choosing at random. A similar approach was taken for the selection of tasks, basing them on a frequency analysis of existing storytelling features found within a wide variety of video games. These tasks were the foundation for the structure of the LRRH script, as well as for the tasks given to participants as guidance. Following was an in-depth description of the employed methodology, the procedures for gathering data from the study, and discussion of the techniques behind the data analysis.

The six resulting principles (*Metaphor Testing*, *Fast Track Testing*, *Structure*, *Experimentation*, *Branching*, *Contented Authoring*) highlight relationships between the design of user interfaces in IDN authoring tools and their potential impact upon the workflow of authors. These principles are intended to serve as guidance for future authoring tools and as a vocabulary for this field to further discuss the UX issues facing authorship technology. This enables us to identify how different design decisions for authoring tools hinder or aid the authors, and provides us with an understanding of how the decisions may impact the authors. This can help us as developers to make more intelligent decisions about our interfaces and UX choices, which ultimately serves to create better authoring tools that can reach and better support a wider audience. These six principles also provide the foundation for answering the third research question, which queries the user experience principles for informed design of video game narrative authoring tools.

CHAPTER 5

A Pipeline for Authoring Tool Design

As raised in §4, the user experience of authoring tools is paramount, and if neglected it can act as a gatekeeper for the accessibility of the tool, act as a source of frustration, and impact the creative process of authoring interactive narratives. To increase the usability of our authoring tools, we must not only design them around the underlying narrative data models, but also consider the authoring experiences at all stages by involving established design and development processes. Moreover, authoring tool designs should follow best user experience and user interface design practices to aid in bettering usability and to provide a genuine source upon which we can run user experience studies. Existing publications in this space detail features of the tools and sometimes an overriding philosophy behind the design of the tools, such as support for specific patterns [149] or language-based principles [139]. However, detailed design principles for user experience, such as those found in traditional software or games [144, 145], are rarer. This is despite discussions in this space identifying the key user experience concerns with these authoring tools for some time [201].

This chapter presents an examination of UX-focused design pipelines, from which trends are identified and used to establish a custom design pipeline formulated for the creation of interactive narrative authoring tools. This pipeline is then applied in the context of designing a prototype authoring tool that implements the design principles proposed in §4, using the architecture as discussed in §3.1. The purpose of this prototype authoring tool design is to be used as a device for validating and refining the principles in the study that follows in §6. The creation and use of this professional UX-focused design pipeline ensure that, when used in the study, the design is fair and of high quality, having followed established best practices when being created.

The creation and execution of this pipeline was published at *ICIDS 2020* [84].

5.1 User Experience Design Pipeline

The Nielsen Norman Group (NNGroup) evangelize the inclusion of user research (UR) and user experience research (UXR) at all stages of the product design cycle. They present a generalized set of phases[51] to serve as broad guidelines for the inclusion of UR and UXR

into product design cycles. The model is broken up into four phases — *Discover*, *Explore*, *Test*, and *Listen* — each referring to a different stage of the design process. It is expressed that the phases are not necessarily sequential and are expected to somewhat overlap, and that teams should appropriately choose methods based on their time constraints, system maturity, type of product, and local contextual concerns, rather than rigidly follow their listing of available methods during each phase. The *Discover* phase is where you gain an insight into user needs and begin building listings of requirements and constraints. After this stage, you should understand the concept of what is being built and what users' needs are. This stage is essential, as it ensures that what follows is informed. The *Explore* phase is the densest, where a variety of exploration methods are employed to begin investigating design possibilities to suit the user and system requirements. This includes techniques such as creating personas, creating paper prototypes and low-fidelity wireframes, and using various mapping methods (empathy mapping, journey mapping, etc.). The purpose here is to iterate upon and explore designs that support what your users need and what your system requires. The *Test* phase uses testing and validation methods to check that designs work well for the target audience. This involves various kinds of qualitative usability testing and task-based studies, often using think-aloud to better understand user processes. The *Listen* phase is for continuous refinement further in the product design cycle, helping to understand existing problems and looking for new ones. This usually comes in the form of surveys or other analytic and metric monitoring.

In his book *Wireframing Essentials* [96], Hamm proposes a slightly different approach for the involvement of UR and UXR in the design process. Figure 5.1 shows an overview of the pipeline's four subprocesses. It is again explained that there is no single process to follow, but instead that common patterns occur.

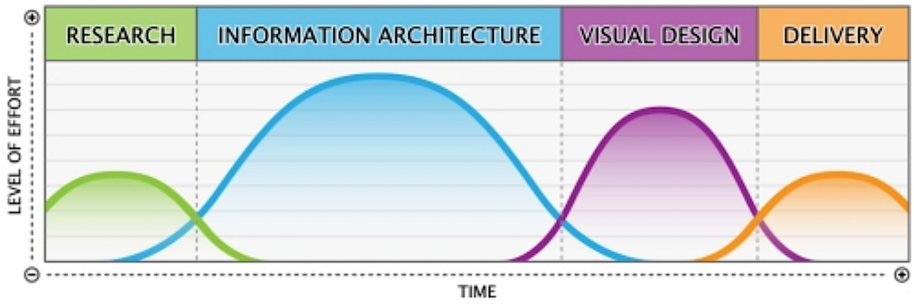


Fig. 5.1 Hamm's four-stage design process as outlined in *Wireframing Essentials* [96].

The *Research* stage is identifying who will be using the software, what their needs are, what technology will be used, and the requirements of the software, as well as any other information that should be known prior to actual design beginning. It is vital to complete this research phase prior to prototyping to ensure that design considerations are informed. This is analogous to NNGroup's *Discover* phase. The *Information Architecture* stage is responsible for exploring the potential approaches and flows of the design. This involves determining the content required to support tasks that users will be doing, creation of wireframes to explore potential designs, and refinement of these designs with

users. This is similar to the *Explore* phase in the NNGroup approach. The *Visual Design* stage is where a given design or set of designs are solidified and gradually transitioned into high-fidelity wireframes, pixel-perfect mockups, or similar accurate representations of the potential final product. Throughout this stage, refinement is continued and further analysis methods are used to determine and fix usability problems. This stage has no equivalent in the NNGroup model but straddles the line between their *Explore* and *Test* phases. Finally, the *Delivery* stage is where the final mockups are used along with other supporting evidence to create documents to be passed to the development team as a basis for implementing the product. This stage does not have an equivalent in the NNGroup model but is closest to their *Listen* phase.

Using these two approaches and their descriptions, we can infer that while there is no fixed pipeline for the involvement of proper user experience processes in the design cycle, general trends appear to surface. We can also say that UR and UXR methodologies used should be chosen based on the local contextual requirements and needs of each project individually to form an amalgam of methodologies rather than following a rigid predetermined list. Broadly speaking, both methods begin with pre-development research, gathering user insights and design requirements. They then draft up potential designs informed by the research, involving users from the start. Candidate designs are then developed further and refined using various user testing and evaluation methods. The end result differs based on individual projects, but generally involves passing solidified designs and requirements to further departments and executing additional UR and UXR activities. Similar sentiments can be found in the book *Designing the User Experience of Game Development Tools* [132] in which it is recommended to begin by understanding users and their needs with methods such as personas, storyboards, and task flows, and also advises on task-based user testing for validation and refinement of materialized designs. Similarities between this high-level approach can be seen within related fields of the game design and development process beginning with concepts, graduating through essential design, and then validating and refining the design. This is done all while considering users [55] and employing a variety of research methods such as PLAY heuristics [56], RITE studies [144, 145], and think-aloud [127] playtesting.

5.1.1 Employed Pipeline

The pipeline that I developed and followed was inspired by a combination of the two above approaches. An overview of the pipeline can be seen in Figure 5.2. There are three phases, similar to those of the NNGroup model: *Research*, *Discover*, and *Refine*. Unlike the other models, a fourth phase is not included as it typically concerns delivery or post-launch analysis of the product which are both beyond the scope of this framework. The *Research* phase includes creation of a persona to represent a typical user of the target audience, analysis of where in the game development pipeline the software would best fit, and a minimum viable product (MVP) listing of all required features per user requirements

Fig. 5.2 The three phases and subtasks of the developed pipeline. *Research* finds out who you are making a product for and what it should do, *Discover* explores initial designs to satisfy those criteria, and *Refine* iterates to produce a tested design.

and expected system functionality. Kano Analysis [107] could also be included here to help determine potential users and the value of requirements but at the expense of resources and time. This ensures that the target audience is known, the software is positioned for practical use, and early requirements are identified. The *Discover* phase involves creation of candidate designs in the form of static, low-fidelity wireframes, and involvement of potential users in an exploratory participatory design process, both based on the previous phase's research. After this phase, there should be a final candidate design ready to progress. The *Refine* phase takes the final candidate design and from it creates a high-fidelity, partially interactive wireframe. This mockup is then refined further using the RITE usability method, which will be discussed in detail in §5.4. This ensures that the final mockup has been tested with actual users to identify and fix any potential usability issues. These three phases are informed by professional industry-standard techniques, and after completion, will result in a refined mockup prototype that is suitable for use as a base in the upcoming study described in §6.

5.2 Research

This section will cover the *Research* phase of my pipeline including persona development, pipeline analysis, and the creation of the minimum viable product requirements.

5.2.1 Persona Development

As explained by NNGroup[52], a persona is a fictional yet realistic depiction of a typical or target user. They work as archetypes but should be treated as if they are an actual person with factors such as age and background considered. Personas directly complement the philosophy of user-centered design by being a focal point during decision making, ensuring that the choices made remain influenced by users.

Personas generally evolve from observations of and insights into actual target users, gradually oscillating toward one or more as roles and trends begin to emerge. Because access to relevant working professionals is limited, an alternative approach was taken to gathering information for the development of the persona. As the typical use of the prototype tool would involve working with narrative structures, it was determined that the vision of the tool fits well with the role of Narrative Designer. 14 related job postings from a wide range of leading contemporary video game studios were analyzed and used

Table 5.1 Tabulated skills and requirements from aggregated job listings.

Skills	**Times Mentioned**
Worked in AAA games development	14
Writing skills	10
Genuine interest in video games	9
Good at working in a team	8
Understanding of interactive narrative	8
Knowledge of scripting or programming	7
Clear communication skills	6
Graduate degree	4
Knowledge of scripting tools like Twine	4
Good knowledge of game production pipelines	3
Knowledge of Unity or Unreal	3
Good knowledge of game design	2
Mastery of the English language	2
Familiarity with task and scheduling tools	2
Requirements	**Times Mentioned**
Create narrative structures	12
Work with writers	12
Document the narrative structure	12
Assist implementing narrative into the game	11
Additional writing	8
Be a point of contact for film and voiceover talent	7

as a representative sample of expected skills and requirements in place of traditional methods of gaining insights into the target audience. The job advertisements were filtered to avoid entries too specific to the company such that only general skills and requirements were left. Tabulated values of both skills and requirements are listed in Table 5.1.

These tabulated skills and requirements were combined with supplementary information on the role of a Narrative Designer present in *The Game Narrative Toolbox* [100]. In particular, that there is a clear overlap between Writers and Narrative Designers, but that the latter are often more technical and have additional game design skills. It is also described that Narrative Designers are familiar with but not experts in contemporary game engines and the basics of scripting.

Together, these factors informed the development of a complete persona for Catalina Drake, a Narrative Designer at Libertalia Games, including her personal profile, behavioral considerations, frustrations, and typical work tasks. For example, Catalina's profile states that she has studied creative writing in college and game design in university, and that she enjoys working in-engine to implement aspects of the story rather than focusing exclusively on writing. Moreover, she typically works with writers to create informed narrative structures, assisting with basic scripted events and sequences in-engine, as well as being a point of contact for dialogue and voiceover recording and implementation. The properties of this persona should be considered at all times when creating an authoring tool design to ensure that it stays aligned with the target audience's skills and typical

tasks. For instance, knowing that the target audience is comfortable with writing basic scripts permits lower levels of abstraction when building conditional statements within the authoring tool's user interface, whereas somebody who is not comfortable with scripting would require a solution at a higher level of abstraction. The complete persona is available in Appendix C.1.

5.2.2 Pipeline Analysis

An important part of early research for authoring tools development is to not only know how the tool will be used but *when* it will be used. In the case of video game narrative authoring tools, this is equivalent to knowing where the tool fits within the game development pipeline. No two pipelines in the video games industry are the same. The point at which the narrative team is involved and their level of engagement with development differs not only between studios but even between games of the same studio.

In games with little to no authored narrative, such as the likes of *Tetris*, it is unnecessary to have a team of dedicated narrative developers, but with a game that focuses on its narrative experience, the narrative team will be larger and have a more significant impact in the development process. This is demonstrated in a GDC Panel[53] during which the panel explains that their studios (which develop games with a strong narrative) have dedicated Technical Designers and Narrative Designers that liaise with the non-technical Writers to determine capabilities and limitations, as well as assisting the core development team with the implementation of narrative into the game. This approach is further discussed in a talk[54] by the Lead Writer and Mission Design Director on *Assassin's Creed III*. It is explained that Mission Scripters work closely with traditional Writers from the beginning to implement prototypes of narrative behavior into the game early on. These are then either refined or replaced with Writer support as the game matures and more technical implementation begins.

When dealing with positioning something in a pipeline, it is also important to consider what other steps come before and after it. *The Game Narrative Toolbox* [100] refers to prior processes as *upstream* and to those after as *downstream*. That is, you depend on upstream to do your job, and downstream processes require you to finish your job before they can complete theirs. Narrative Designers are under constraints based on this upstream and downstream structure, which like pipelines, differs per studio. These constraints can impact both the writing teams and the teams responsible for implementation.

Using the above information and keeping in mind the persona and underlying functionality of the theoretical Novella model that backs the authoring tool, we can now formulate a typical use case of the system within an example games development pipeline to help solidify its place. The model and authoring system are targeting Narrative Designers (and similar) that work with other teams, chiefly Writers, to implement the narrative vision of the game by, for instance, planning the narrative structure and hooking it up into the actual game engine using the tool. Evidence has shown that this often requires

liaising with a writing team to plan and implement simple prototypes of narrative features and interactions. Therefore, we can say that regardless of pipeline configuration, it is likely that Writers (and similar) can be considered an upstream constraint for Narrative Designers using this authoring system. The downstream teams could differ greatly based on the structure of the team, but it is conceivable for this system to work in parallel to other processes such as general design and gameplay implementations.

Let's now look at a hypothetical authoring scenario for complete use of the Novella authoring platform, including use of the API, content creation, and the authoring tool. For this example, let's assume that a game studio is working on a project that involves a lot of player interactions and has dynamic narration based on their actions. Firstly, various teams work together to identify what features and direction the game likely has, followed by an implementation by technical and design teams that proves a development playground of essential gameplay and early level prototypes. At this point, teams can work together to decide what to expose and implement in Novella's Logic, which essentially defines the level of interaction and control that the Narrative Designers would have when writing Logic code within the authoring tool. With this information, programmers can then use the Novella API to hook up Logic into actual game code such that Narrative Designers can sufficiently interact with and manipulate the game to satisfy the vision of the design and writing teams. Finally, Narrative Designers can work with Writers to implement their vision into the game using the Novella authoring tool to create structural layouts and connect them to the game engine using the built-in and exposed Logic functions added earlier. At this stage, the content from the Writers, such as dialogue for response to player interactions, can be added in the authoring tool using Logic. This process can be done at varying levels of detail and can be repeated and refined throughout development rather than being a one-time process.

5.2.3 Minimum Viable Product

The Minimum Viable Product (MVP) is the simplest set of core features and actions for a product that allows it to be initially deployed as a working product. The mindset behind this approach is to rapidly deliver a product that implements core functionality and nothing more, with further refinements and additions being made after the product is launched based on actual user feedback. From a business perspective, assuming the correct research has been done, this means that an initial product can be released and tested quickly, helping resolve problems and gain feedback for future iterations. The MVP can be created at varying levels of granularity, with finer details being suitable for a well-defined product, and coarser details being more suitable for exploratory works.

In the case of this prototype, the MVP is building and understanding a core list of constraints and requirements that consider the persona, pipeline position, satisfy the underlying Novella model functionality, and cater to the expected functionality of the

tool, all while keeping in mind the principles identified in the previous experiment, as they are the motivating purpose of why this prototype is being developed.

For example, if we consider the above persona, we know that the target audience expects to implement narrative structures into the game which requires interaction between the model and the game and that they are comfortable doing so with basic scripting. Moreover, we know that managing structure of their content, as well as the ease of editing and iterating upon it, is important to them.

Considering the *Novella Narrative Model* described in §3.4, we can also infer that the authoring tool design must support hierarchical and nested structures due to the multi-layered nature of the model, support appropriate connectivity between sibling elements, have a way for editing properties of all elements including links, allow for creation and editing of variables and scripts to handle internal Logic and external communication with a runtime, provide a way to edit Entities and Tags, and distinctly separate each type of element within the user interface.

When looking at the principles in the context of the MVP, *Metaphor Testing*, *Structure*, *Experimentation*, and *Branching* each benefit from the use of spatial hypertext graphs with nodes representing content connected visually by lines between the nodes, which suggests that a node-based design would be best suited to satisfy the principles. Moreover, *Fast Track Testing* highlights how the proposed design should ideally support a built-in testing harness that allows testing to begin at any location or with any state for improved testing experience and quality.

5.3 Discover

The *Discover* phase is about creating and iterating upon candidate designs based on the output of the *Research* phase, preparing them to be further developed in the *Refine* phase that follows. The designs at this stage are preliminary ideations rather than being detailed and thought out, as their primary function is to provide a base upon which a mature design can emerge. In order to achieve this, a methodology was developed with a participatory design process as its foundation where interactive narrative authors would help to shape the initial designs. Participatory design does not refer to a single process but is an umbrella term for a vast number of related yet differing methodologies [156]. Regardless of the details, all variations include involving people from distinct backgrounds and mindsets working together in the design process. In the context of the authoring tool prototype, this process was chosen to not only build and refine candidate designs with actual potential users, but also to gauge how they interpret and envision such a tool without having seen a design in advance. This approach also helps to uncover faults or desires early on in the design process by involving users in ideation. The NNGroup pipeline also recommends involving stakeholders in the exploration of early designs.

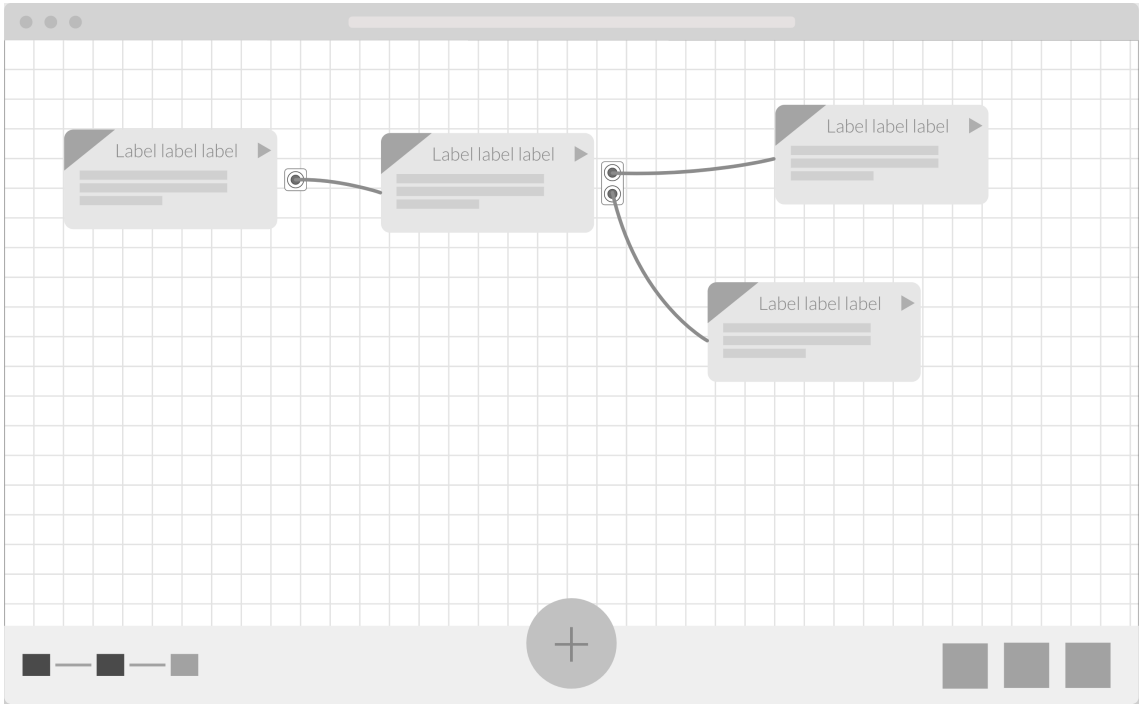
5.3.1 Preparation

Taking into account all information gathered in the *Research* phase, several initial candidate designs were sketched out, with two designs being implemented as single screen static wireframes using *Adobe XD*. Each of these designs is visible in Figure 5.3. Considering both of these designs from the perspective of the MVP and preliminary principles from §4, several traits are present despite being basic exploratory wireframes.

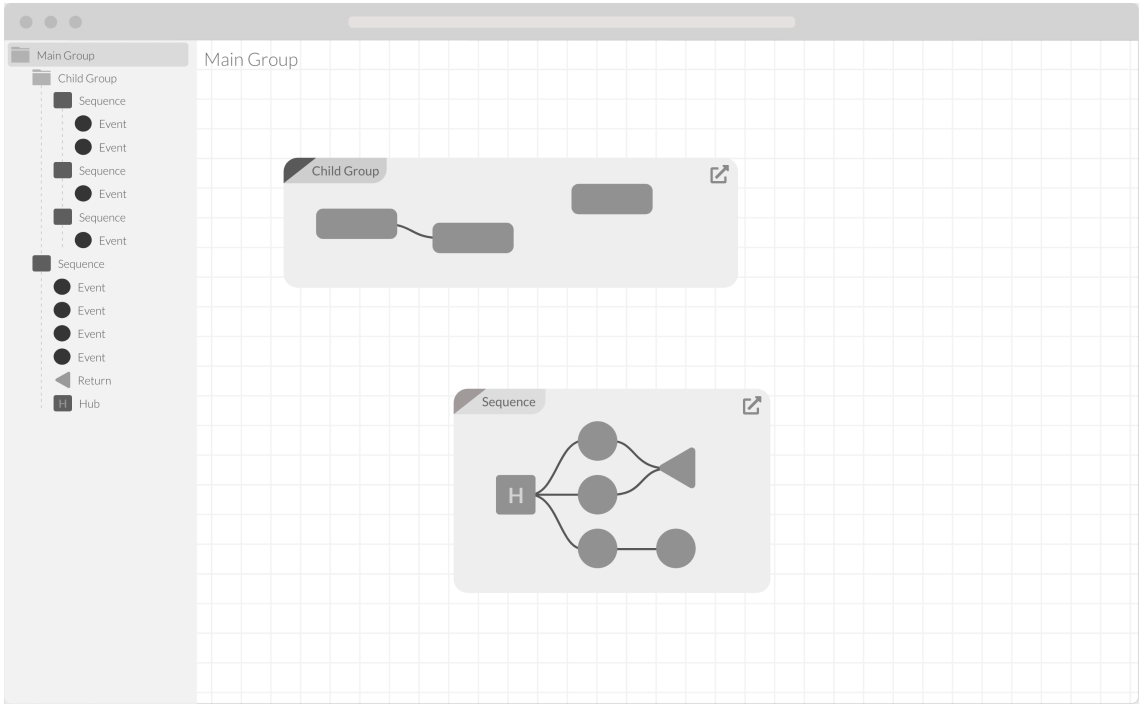
For instance, both designs support hierarchical structures which are required by the underlying model; the first uses a breadcrumbs system to represent depth visible in the bottom-left of Figure 5.3a, and the second uses an outliner visible on the left of Figure 5.3b as well as showing the contents within each node in the workspace on the right. Similarly, both designs feature clear connectivity between the nodes which aids several principles, although the former provides a detached visualization of the links whereas the latter simplifies connections to solid lines. The nodes within the second design differentiate the element types with varying shapes (e.g., *Hubs* are square and *Returns* are triangles) to improve readability, whereas the first design instead relies upon a colored flag in the top-left of each node to determine what kind of element it is (e.g., *Hubs* could be orange and *Returns* could be purple).

Both designs make use of spatial hypertext graphs to display the model elements as nodes connected together with lines, providing a well-established metaphor for the underlying structure of a story as also shown by the selection of graph-based authoring tools as discussed in §2.4. The use of this graph-based visual metaphor of story structure enables, in part, the preliminary principles of *Metaphor Testing*, *Structure*, *Experimentation*, and *Branching*. However, both designs consider the principles in slightly differing ways. For example, the principle of *Metaphor Testing* in the first design is partially fulfilled by having distinct nodes with clear connectivity and content from within the nodes exposed on the interface, making it easy to not only understand structure at a glance but also the content within that structure. Despite the rich information presented in this design, it is limited to sibling nodes as only one level of the hierarchy is visible at one time. On the other hand, the second design presents less information upfront on the nodes due to their compact size but does allow for insight into the internal structures within certain nodes. While content cannot be evaluated as efficiently in this design, it does have the benefit of being able to identify, at a glance, not only the structures at a sibling level, but also their internal structures too. These differences in design also impact the implementations of the other aforementioned principles, too, such as how the principle of *Structure* may be altered due to the larger node sizes in the second design, or how *Branching* may be altered as more structural information is available in the second design.

Moreover, both of the designs follow established design trends, particularly the *Gestalt Principles* and several others as discussed in §2.5. For example, the concept of a node within both designs is clearly separate from both other nodes and the canvas upon which they exist because of the *proximity principle* (nearby objects are perceived as related),



(a) The first candidate design. A single node type exposes rich information from the node's content, with model element types being differentiated with a colored flag. Hierarchical navigation is indicated with breadcrumbs in the bottom left. Only one layer of the hierarchy is visible at once.



(b) The second candidate design. Each model element has a distinct node type but does not expose rich information as in the first design. Hierarchical navigation is indicated by a detailed outliner on the left side. The active workspace can show a single layer of the hierarchy, but certain node types allow for viewing the internals of the node as well.

Fig. 5.3 Two candidate design wireframes for use in the *Discover* phase of the authoring tool design pipeline.

the *common fate principle* (objects that move together are perceived as related), and the *common region principle* (objects that share a common border are perceived as related). *Jakob's Law* is also followed in that several of the user interface components in both designs are established and ubiquitous, such as how an outliner is used to show hierarchy in the second design, or how breadcrumbs are used to represent the depth of a hierarchy in the first design. Both designs follow several other axioms of user experience design to ensure that they provide a grounded user interface to build upon, should they be chosen for use in the following phase.

An introduction document that would be presented to participants during their session was also prepared, consisting of several sections starting with an opening paragraph about the goals and the process of participation. This was followed by the persona from the *Research* phase, then a high-level description of the vision of the tool was laid out to give an understanding of the target audience and scope. Also included was a detailed description of the underlying Novella model that the prototype design would implement, and a set of required functionality derived from the MVP. It is critical to the methodology that this section includes no leading or suggestive phrasing that could alter the initial vision of an interface that participants have, and as such, careful consideration was taken with the wording to account for this, avoiding phrases suggestive of any visual representations. Finally, the participant was reminded of their task and reminded that they can review the document before continuing. The complete introduction document is available in Appendix C.2.

5.3.2 Methodology

This methodology was approved by Bournemouth University's Ethics department with the approval number 28234.

The participant demographic for this process was students of digital narrative who have any level of experience with existing authoring tools for interactive narrative. Participants were sourced from related courses at Bournemouth University, although this was not a restriction. Invitations were distributed to potential participants by email.

Participant Protocol

Each participant had their own dedicated session. Although a small focus group could have stimulated discussion, it was avoided as it involves the risk of participants influencing each other's natural impressions of how the interface may look and function. For each participant, they were first taken through the introduction document and given time to ask any clarifying questions, particularly due to the complexity of the underlying model. Following this, they were given a piece of paper that was filled with an empty application window (the same window used in the *Adobe XD* sketches but with all content removed) and were asked to use think-aloud protocol while sketching and annotating how they intuitively envisioned the interface for such a tool based on the descriptions that they

had just read. In other words, given a high-level description and constraints of a potential application, what is the natural design that first comes to mind?

The approaches sketched out by participants were then explored in an unstructured interview focused around their design choices and why they made certain decisions, with questions working to further explore and better understand their creative vision. A topic-based unstructured interview was preferred over a structured or semi-structured format as the goal with this questioning was to identify the reasoning behind their individual designs and choices that they had made, which inherently differ per person. Notes were taken on aspects of their design and the motivations explained behind them. The aspects of the participant's design, in part or in whole, were then considered for use in the prototype, as long as they satisfied the constraints.

Participants were then shown the two existing candidate design variations and engaged in discussion of their interpretation of the designs, as well as the advantages and disadvantages of each approach. Notes were also taken throughout this discussion.

Finally, participants were asked two demographic questions (identifying which interactive narrative authoring tools they have used and what their estimated knowledge of interactive narrative is). These questions were asked at the end to avoid influencing their designs by having them list previously used software before sketching their ideas.

On average, each session took around one hour with slight variance due to the unrestricted time to allow participants to sketch their ideas and the length of interviews being different based on their response detail.

Data Gathering

For each participant, data gathered in their sessions included notes taken by hand on their design, notes taken by hand on both candidate designs shown to them, demographic responses, and the original sketches that they had produced. All of this data is anonymous.

5.3.3 Results & Discussion

The participatory design process was run with three participants. All had previous training in interactive narrative, had experience in the same authoring tools (*Twine* and *Inform*), and reported either moderate or plenty of knowledge of interactive narrative.

All participants intuitively defaulted to graph-based systems with nodes representing objects and lines being used for connections. The visual and organizational approaches differed for each participant, but their concept of nodes and lines remained the underlying technique. Hierarchical relationships were specified by all participants using some form of outliner panel, always positioned on the left of the interface. Within the graph, the first participant relied on nesting within the elements but did not design how to delve inside them to view deeper layers. The second participant used lines to represent both connections and hierarchical structure but later stated that the decision isn't suitable due to the lack of clarity between the two and a lack of support for structure. The third

participant displayed the children of a single element and allowed for the top-level element to be altered, thus changing what is displayed, i.e., traversing through the hierarchy. Property editing had a variety of approaches. The first participant had a popup window appear after double-clicking on any element in the interface. This window was temporary and would disappear after clicking off. The reason for this was to have the graph maintain its larger space due to its importance and still allow non-intrusive editing of properties that did not permanently take up screen real estate. The second participant included a sliding property panel on the right which activated when an item was double-clicked. This panel was fixed but could be hidden, again to save visual space for the graph. The third participant had a hybrid of the other two approaches with a floating panel appearing upon double-clicking an element that could be closed after editing, again with the reason being to save space for the graph. All participants highlighted the importance in their approaches of using distinct shapes and colors to differentiate between elements of the model, with the intention being to aid quick identification.

The first design (Figure 5.3a) was described as neat, clear, and tidy. In particular, it was referred to as intuitive and fluid due to the familiarity of the flowchart-like visuals. The third participant stated that it was almost exactly what they had originally in mind, but more refined. There was mixed feedback about how the design provides contextual information about the current position in the hierarchy. One participant stated that the path system highlighting the current depth wasn't useful enough, but the other two participants commented upon how useful they thought it was for providing context. A suggested solution to this was the addition of an outliner panel, which was previously favored in some form by all three participants in their own designs. Another concern was the ability to identify elements based on color alone, as the design did not use shapes to distinguish node types. While this was only raised by one participant, the concept of using not only color but also shape was present in all of the participant's own designs.

The second design (Figure 5.3b) was preferred by all participants. All participants spoke positively about the inclusion of an outliner due to its ability to provide an overview of the story and given context as to the current position when editing. It was requested by one participant to make this collapsible to further save space for the graph. The most discussed part of the design was the multilayer 'artboard' system which was complemented by all participants as providing a deeper insight into the sublayers of the current parent element, which provides extra context and therefore a better understanding of the story at a given point without the need for further traversal. It was also highlighted by one participant that this system would reduce the number of clicks that they would have to take as they would utilize the artboard preview instead of navigating away. The only criticism of artboards in their current form was that they need more than labels to identify their type, such as by adding shapes and colors, which was found in all of the participant's own designs.

Taking into account the feedback from participants, it was decided that the second candidate design was to be chosen, with feedback from participants being considered for inclusion in the design.

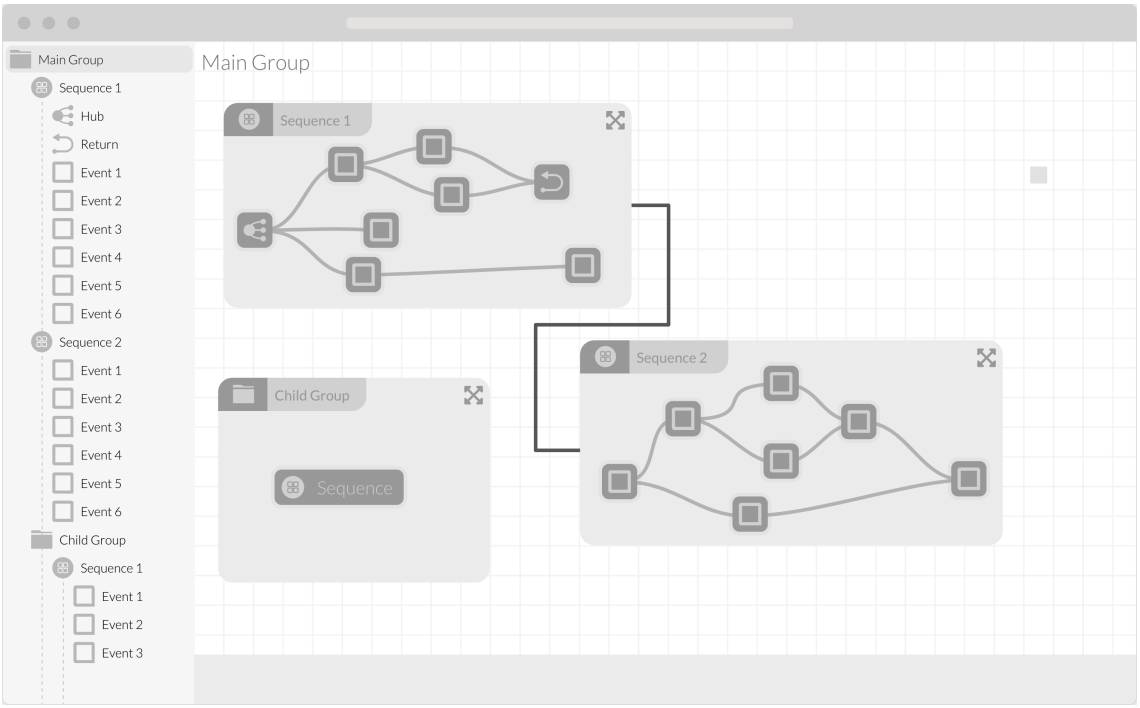
5.4 Refine

The final phase is the *Refine* process. This involves taking the chosen candidate design and feedback from the *Discover* stage, as well as all information gathered in the *Research* phase, and using them to further tweak the prototype. As the chosen candidate design was an approach rather than a complete prototype, it first needed to be expanded upon prior to including participants in the refinement process. This involves iterating upon the candidate design to enhance its features and visual quality, transitioning into a high-fidelity wireframe mockup with limited interactivity. The added features were developed in line with the feedback, constraints, and requirements outlined in previous phases of the pipeline. A design notebook was created and managed alongside the enhancement of the prototype, containing a detailed description of how each feature functions. The final post-study design notebook is listed in Appendix C.5. The prototype was later contextualized by ensuring that the content displayed within the prototype was reflective of an actual game story rather than using placeholder material. This ensures that when participants are involved, they experience a more natural environment when interacting with the prototype. Figure 5.4 shows the enhanced chosen candidate design and the initial high-fidelity prototype used in this refinement process. How the resulting design following RITE refinement implements the principles and follows best practices in design is discussed at the end of this section.

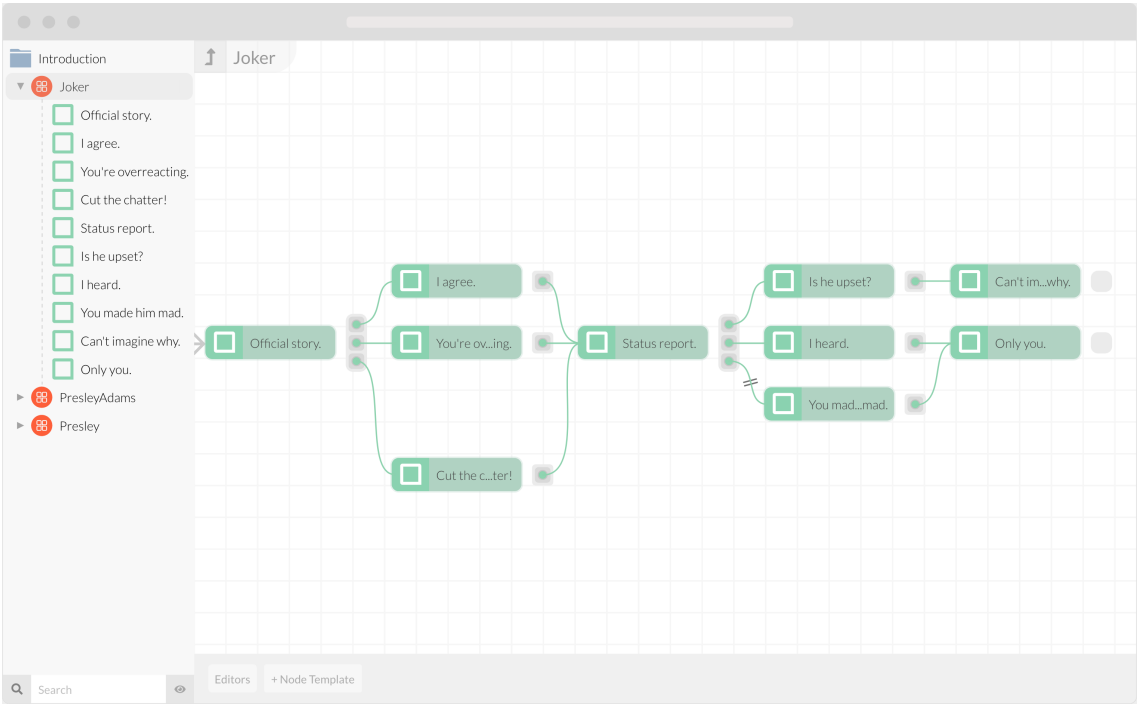
5.4.1 The RITE Method

The methodology used in the design refinement process is based upon the philosophy of the *Rapid Iterative Test and Evaluation* (RITE) method. While this methodology was first formally described by Medlock *et al.* in 2002 [145], it is noted that the approach has been used informally prior to its academic inception for quite some time. RITE is used extensively in the video games industry [144] but also has case studies in more traditional application development environments [144], one example being the refinement of the UI and UX of *Microsoft Digital Image*[55].

The philosophy behind RITE is quite trivial. It states that when a problem is found, it must be fixed as soon as reasonably possible, and calls for those that decide how the fix is made to be part of the research team. This idea places the responsibility of determining the validity of encountered issues on the researchers themselves using observational data and expert knowledge rather than quantitative statistics and other hard data. This kind of thinking is often shunned in academic settings due to the lack of an absolute truth, but conversely is greatly endorsed elsewhere due to its effectiveness and efficiency in



(a) The enhanced chosen candidate design. Minor changes have been made such as adding icons for each node type, modifying the flag shape next to a node's label, and changing the Artboard expansion icon.



(b) The initial high-fidelity contextualized design. Several changes were made including colorizing of node types, detailing connection panels, adding collapsible Outliner items, adding a search area and toolbar buttons, and adding buttons to enhance hierarchical navigation.

Fig. 5.4 The chosen candidate design (a) refined into a contextualized design (b) both created in *Adobe XD*.

finding problems with a system. As explained by Medlock [144], "it is not that [video game] businesses do not care about the 'truth' — it's that they are willing to sacrifice it if it gets them 80% there quickly". The idea is to spend more time focusing on the fix than on the failure, and seeing if the fix worked rather than trying to determine if you have found all possible issues and understood them perfectly.

The core methodology of RITE takes inspiration from traditional usability studies in that participants are actual users who are given tasks, researchers observe and record what participants do and think, and then this information is used to make changes. The RITE method positions itself as a discount usability test to be used in place of a traditional usability test. RITE differs itself from a traditional usability study in that data are analyzed after each participant (or at least after a small batch), changes are made as soon as a problem is identified and a possible solution is drafted, and that the modified system is shown to participants that follow to see if the changes have had the expected impact. Figure 5.5 visually shows the difference between RITE and a traditional usability study. A key component to the success of RITE is keeping a proper record of all troublesome encounters that participants face and documenting each mitigating fix that is made. This is typically captured in a spreadsheet that is updated for each participant. In order to help prioritize and better understand problems, it is useful to categorize issues into *errors* and *failures*. The definitions of these are up to the researchers, but typically an *error* is a non-critical bug that causes confusion but does not prevent completion, whereas a *failure* is a critical bug that prevents completion of the task.

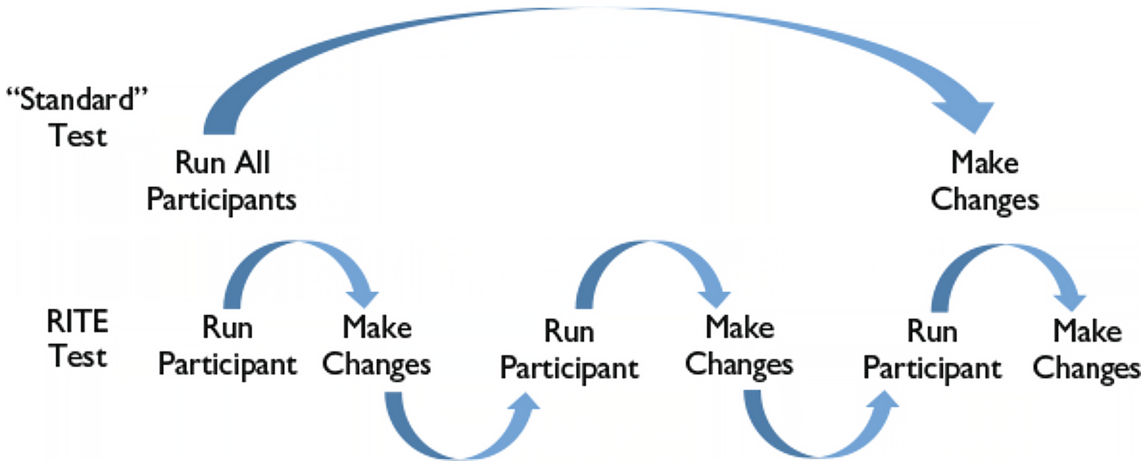


Fig. 5.5 RITE versus a standard usability test. Courtesy of *Games User Research* [144].

Another critical requirement of the RITE method is that the system being tested must be able to change often and quickly. Because changes are made between participants, sometimes between sessions on the same day, the interface needs to be able to change rapidly, ideally without the concern that making changes will introduce unexpected bugs. It is also recommended to save complete versions of the system between participants to make fixes easier to revert if problems are encountered with them. When to use RITE over traditional usability tests is dependent upon the researcher and the particular requirements of the research. In the case of refining the prototype design, RITE is

chosen to efficiently refine the user experience, whereas using a traditional usability methodology for the same task would take much longer. Additionally, the combination of an interactive, high-fidelity prototype goes hand-in-hand with RITE due to the ease of making changes without the worry of introducing technical problems. This is especially true in comparison to the time required and complexities involved with implementing and modifying a functioning prototype stable enough for use within a study.

5.4.2 Methodology

The methodology for this study was approved by Bournemouth University's Ethics department with the approval number 30531.

The goal of this methodology is to produce a prototype with a refined user experience and feature set based on the RITE philosophy which detects and fixes miscommunications and misunderstandings that participants encounter. Following this process, the user experience of the prototype will have been refined such that the problems are removed or at least mitigated. The demographic for this process was students of digital narrative who have any level of experience with existing authoring tools for interactive narrative. Participants were sourced from related courses at Bournemouth University, although this was not a restriction. Invitations were distributed to potential participants by email.

Participant Protocol

Each participant had their own dedicated session. They were first shown an introduction document that prepared them with context for the prototype that they interacted with when completing tasks. The document covers a summary of the persona, an overview of the tool as a whole, a listing of the types of nodes in the prototype and their functionality, and a brief summary of the Logic scripting system from the perspective of a user. Following this document, the participant was then shown and walked through three short video clips from *Mass Effect*. The narrative scenes from these three clips were modeled within the prototype design to provide a contextual and representative experience. The complete introduction document is available in Appendix C.3.

A list of tasks was generated from a collection of mandatory functionality — a listing similar to the MVP but for this particular design — that users should be able to perform in the prototype design. These tasks collectively ensure that the participant is able to perform each mandatory functionality item at least once. For example, one mandatory functionality is that authors must be able to edit the label and description of each node, which is targeted by asking the participant "You want to refine the description for the PresleyAdams Sequence. Walk me through how you would edit the properties of this node". To ensure that participants were not given tasks in isolation from one another, they are ordered such that they iteratively build upon each other while creating a story within the prototype design (e.g., a node is added, then moved, then grouped, and then edited in that order), hence in the above example, the participant is already familiar with

how to find nodes and what the PresleyAdams Sequence is. In some cases, a mandatory functionality item may be targeted by multiple tasks. For instance, there is a mandatory functionality item that requests authors must know how to navigate along a pathway in all directions within the built-in testing harness, which is primarily targeted by asking them "Please navigate and follow the story until **Status report** is the current node", but several other tasks, while chiefly focusing on a different item, include further elements of navigation around this same context. Due to the interactivity limitations of the prototype, some tasks were phrased as 'show me how' or 'walk me through' in which participants use both the prototype and think-aloud protocol to explain their thoughts. This process is semi-structured in that the tasks are core but further questioning and exploration can take place based on how the participant responds. The mandatory functionality and list of tasks are available in Appendix C.4.

If when completing a task a participant required minimal assistance such as clarification of behavior or pointing in a broad direction, then the RITE spreadsheet was marked with an **X**, representing an error in the form of miscommunication of the feature in question. If when completing a task a participant required assistance to the point where they were unable to continue without it, then the RITE spreadsheet was marked with a **Z**, representing a failure in the form of a complete misunderstanding of the feature in question. If a mark was made in the RITE spreadsheet, further context-specific questions were asked to identify the cause, which aided in creating solutions. As participants think aloud, their general feedback on the user experience was also noted and used to aid solutions and idea generation.

For each task, the participant's expected solution was documented and compared with the intended solution outlined in the prototype's design notebook. When taking notes, the participant's answer was repeated back to them with slight rephrasing to ensure it sounds more natural [31]. This has the advantage of confirming that the researcher has understood the participant's thinking, which is especially important if their answer was ambiguous, but it also has the benefit of encouraging the participant to further explain their answer by adding clarifying points and gives the researcher additional time to complete detailed notes that would be difficult in a real-time observation scenario. Actions and expectations of participants that differ from the expected solution were highlighted in order to identify the troublesome areas.

Tweaks and fixes, which are detailed below, were made to the prototype between participant sessions based on RITE spreadsheet entries and general feedback from participants. Changes made due to RITE entries, as outlined by the method itself, may or may not be acted upon depending on whether enough evidence was gathered from the sessions during which the incident occurred to make an informed change, or that the incident was deemed unnecessary, with reasoning, by the researcher. The actual change made was often a combination of user feedback with informed intuition on the researcher's part as to how the design could be improved to avoid the same problem arising with future

participants. Additional changes based on gathered feedback and general observations can also be made between participants as long as changes are documented and shown to future participants. In some circumstances, changes may result in modification of the design document or even the task list that is given to participants to complete. In the case of the design document, changes were made and comments are added to show where each change originated. For the task list, changes were made and can be compared to the previous participant's task list. If the prototype was changed between participants for bug fixes, it was documented in the changelog for the participant they were made after.

Data Gathering

For each participant, a text document was created detailing changes that were made due to RITE and those made from feedback, observations, or bug fixes. The task list that participants were taken through was also stored annotated with notes taken during the session which were used for comparison between expected outcomes and actual outcomes. An updated copy of the design notebook and an incremented copy of the prototype with all modifications made were also stored per participant. The unmodified prototype shown to each participant is the previous participant's modified prototype. All of this data is anonymous in its entirety.

5.4.3 Results & Discussion

Seven participants were included in the refinement study with all sessions taking place in person. For usability studies such as this, it is established that only a small number of participants are required to uncover the majority of usability problems [167]. Each participant experienced and interacted with the prototype using an *iPad Pro* acting as a real-time preview of the *Adobe XD* project. The host device is able to control which screen is displayed to the participant, which helps in managing tasks that may jump between different views without the participant noticing, resulting in an easier experience for them. Given that the study required switching between screens for most tasks, this method was the least intrusive. If this study was performed with a laptop alone, then each time a screen needed switching, the participant would have to move and look away while the screen was switched to avoid seeing parts of the project in *Adobe XD* that could reveal answers to upcoming tasks and influence their responses. Another benefit of this setup is that notes can be taken digitally on the host computer without the participant being interrupted, which makes for more detailed and faster notetaking. Participants were informed that regardless of using a tablet device, the prototype application was intended for a desktop environment with a mouse and keyboard, and that their thought process should consider this, to which no participants had any trouble. Due to the task-focused interactivity offered by the prototyping software, the act of using a tablet did not result in the loss of any interaction methods versus using a desktop preview. The only exceptions to this are tasks 17 and 33. The former required the participant to temporarily

Task	Problem Description	P1	P2	P3	P4	P5	P6	P7	1st Change	2nd Change	3rd Change
4	Couldn't figure out how to create a Frame.			X					Added a Frame icon to the toolbar at the suggestion of participant 4.		
8	Struggled to identify the nodes in the Canvas.		X						Added "Main" Group to the Outliner to assist with identification of elements.		
6	Thought that double clicking a node edited its label.		X	X					Made double clicking the node edit the label.		
10	Unable to navigate into a node in Artboard mode.	X							Replaced FontAwesome's `expand-arrows-alt` with `expand-alt` and darkened color.		
13	Unable to add Node Templates to the Canvas.	X							Changed `+ Node Template` to `Insert Node Template`, moved buttons to right side, reordered buttons, darkened text.		
13	Unable to create Node Templates from the editor.	X							Added labeled headers to the listings in all Editor panels, except the Variable Editor.		
17	Unable to add links to nodes.		X	Z					Modified connection panels to animate in a + button on hover to make it more obvious.		
28	Didn't understand which scripts are inside the Scripts window.	X	X	X					Added a label to the left of the script dropdown and hidden the explanatory text in an information popup.		
32	Couldn't tell that variables highlighted yellow are related to the outputs.		Z								
33	Struggled with states in the Simulation Mode.		X	X					Added an explicit save button that mimics the submitting behavior, and moved delete behavior into a popover no longer requiring applying states to delete them.	Updated "Delete" button to be "Delete..." to signal that it's not the current selection that will be removed.	Renamed "Delete..." to "Manage".

Fig. 5.6 The RITE spreadsheet that details changes due to errors (**X**) and failures (**Z**). Columns P1–P7 are participants. The last three cells are color-coded changes made in response to **X** or **Z**. Colored cells in P1–P7 mean the corresponding change was present for that participant.

use a mouse, as the feature relied upon hover states, which are not available on a tablet, although this was added during the study as a refinement rather than an initial constraint. However, no other tasks required hover states, and as such this was deemed to be an outlier in the methodology. The latter required the participant to use a simple desktop application developed in *Xcode* that mimicked a small segment of the interface as shown in Figure 5.13. This was due to limited interactivity methods available in the prototype, and that the feature in question required typing into an editable combo box, which is not possible in the prototyping software regardless of the medium used for participant interaction. This was faithful to the interactive prototype's design and no participants had trouble recognizing the resemblance.

The RITE spreadsheet displayed in Figure 5.6 shows the problems that participants **P1** through **P7** encountered and changes made to mitigate them. A present **X** or **Z** represents errors and failures as discussed earlier. If a given cell is empty, the problem was not encountered by the participant. Shaded cells represent their corresponding change being implemented and used. Sometimes other changes were incorporated in an attempt to improve the user experience based on observations on participant expectations and feedback. These changes, of which all are not discussed below, were not included in the RITE spreadsheet, but were documented.

Node creation. Creating nodes was originally done only via the Canvas' context menu. This was tested in task 3, during which the third participant suggested adding additional physical buttons. The suggestion was implemented by adding an array of icon buttons to the Toolbar at the bottom, as shown in Figure 5.7. The primary function of these buttons was to click and drag from the Toolbar onto the Canvas. To distinguish them from

regular buttons, a dotted outline was placed around each button to suggest an alternative interaction method. After this change, all participants used the Toolbar buttons to create the nodes and correctly identified the drag-and-drop interaction method.

Fig. 5.7 Toolbar icons for all node types and Frames, each with a dotted outline to suggest an alternative interaction method from standard buttons.

Artboard button. In task 10, the first participant incorrectly assumed that the icon on nodes in artboard form was for closure rather than expansion and only clicked it after guessing. This icon was updated to be less ambiguous, and no following participants experienced any troubles. The old and new icons are shown in Figure 5.8.

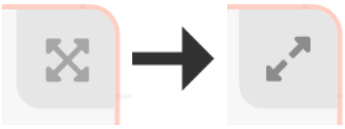


Fig. 5.8 The Artboard expansion icon was changed to something more commonly used.

Frames. Frames, explored in task 4, underwent a variety of changes. Initially, Frames were only able to be created through the Canvas context menu and did not take into account any existing selections. While the first participant had no trouble working with them, they did suggest making the Frame automatically fit to an existing valid selection. This was added, the design notebook updated with the new behavior, the prototype modified to show a selection prior to Frame creation, and the corresponding task rephrased to include the existing selection. The third participant did require assistance with the creation of a Frame, but no change was made as their thought process and suggested approach greatly increased complexity and was contradictory to established paradigms of similar features in existing software. A further change was made after participant four, who suggested adding a Frame button to the Toolbar, as it was where they instinctively looked. This was implemented, and all participants that followed naturally used this button over the Canvas context menu creation method.

Adding links. The way in which links were added to a node's connection panel, tested in task 17, underwent significant change. Originally, links could only be added by using the connection panel's context menu. Participants two and three struggled with creating links, although both understood the functionality once it was explained to them. A fix was introduced after these two participants which modified the behavior of the connection panel to reveal a hidden "+" button upon hover, complimenting the context menu approach as an explicit and tangible alternative. This is shown in Figure 5.9. This change required future participants to use the host system for this task instead of the tablet device as it

relied on hover states which the tablet device cannot do. The remaining participants all initially gravitated toward the hover functionality for adding links, mentioning the context menu as their second guess, demonstrating its success.

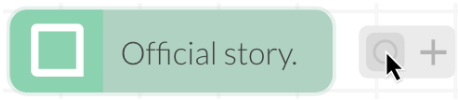


Fig. 5.9 When hovering over the node's connection panel, a hidden plus is revealed to quickly add new links as a visual alternative to using the panel's context menu.

Outliner. Originally, the Outliner only displayed nodes created by the user. In task 8, participants were asked to identify the types of the nodes on the Canvas. The second participant used the Outliner hierarchy to aid them but incorrectly identified the node types. To help distinguish the types, the "Main Group" was added back into the Outliner hierarchy. After this change, which is shown in Figure 5.10, participants were able to identify the node types without trouble due to the increased context.

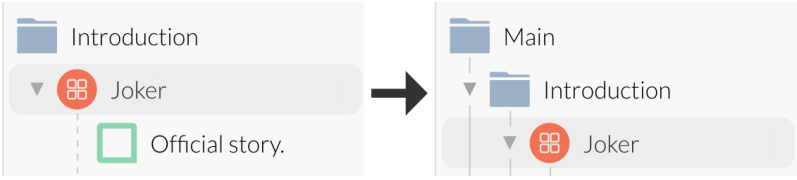


Fig. 5.10 Adding the 'Main Group' to the root of the Outliner for enhanced context.

Node properties. How properties of nodes are edited was refined in task 6. Initially, double-clicking a node would bring up a floating window for the node's properties that is dismissed when interacting outside of its bounds. This approach was chosen based on the participatory design process to save on visual real estate yet still maintain a dedicated editing panel. The first participant suggested that the popup window should have the option to be detached, which would be topmost and permanent unless manually closed. This behavior does not impact the original approach and instead provides an additional way of handling editing of node properties. The double-click interaction to open the properties panel was a source of light confusion for the second and third participants, as they both interpreted the interaction as editing the node's label. To remedy this, the double-click behavior was modified to edit the node's label in-place (this also applies to the node's description if it is visible). A similar change was made such that when a node is initially added to the Canvas, its label becomes editable until the author confirms the label by either submitting or interacting outside of the node's bounds. Two of the four remaining participants double-clicked the label to edit it, which demonstrates that this fix was received well. The other two preferred to use the context menu to open the properties panel rather than double-click, but this was just personal preference.

Node Templates. The way Node Templates are created and inserted was refined in tasks 13 and 14. The initial configuration caused confusion for the first participant, as

the button labeled "+ Node Template" for insertion was mistaken as creation. This was remedied by renaming the button to explicitly read "Insert Node Template". An additional context menu entry for insertion of Node Templates was also added to the Canvas. After these two changes, no participants had troubles inserting Node Templates. Creation of Node Templates was done from the corresponding editor window by context clicking on a listing and choosing the appropriate item. This was problematic for the first participant as it was unclear that the listing was for user-created Node Templates. This was resolved by adding a labeled heading to the listing which helped to clarify its purpose. This was also added to all other editor windows that used a similar listing. The second participant suggested that a "+" button could be added to the labeled headings that when clicked provides a context menu as an alternative method of creation. After these changes, as shown in Figure 5.11, no participants had trouble creating any elements in the modified editor windows, and most defaulted to the "+" button in the labeled header.

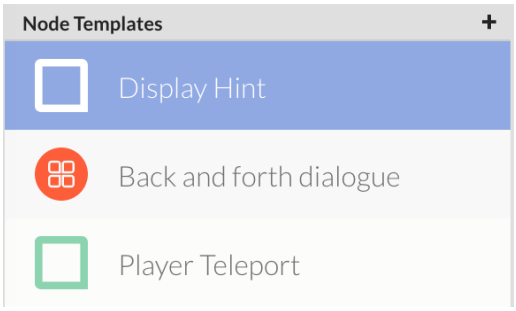


Fig. 5.11 An excerpt of the Node Templates listing with the added header which includes a darkened background, a bold label indicating its purpose, and an add button supplementing the panel's context menu.

Scripts. Changes were also made to the script editing process. All script editor panels have a collapsible error listing, which was touched upon in task 23. While the first participant did not have trouble identifying this panel, they made the suggestion that clarity of its function could be improved by putting a colored bar and error icon to the leftmost side, which reinforces that the panel is for errors. Following this change, all participants described the panel's functionality correctly. Task 28 investigated the process of editing a node's custom scripts. An area of confusion was changing which of the node's scripts was currently being edited and configuring the node to use the custom script. Initially, the top-right of the window had an unlabeled dropdown showing the active script, with some text to its left describing how to assign it for use. The first three participants found this confusing and did not know how to change or activate the script without help. This was fixed by labeling the dropdown to read "Current Script" which better indicated its function. The instructive text previously to the dropdown's left was removed, and a help button was inserted to the right of the dropdown that when clicked showed an expanded version of the help text. With these changes in place, all following participants read the help text and had no trouble selecting and assigning custom scripts to a node. Both the error panel and the popover help text are shown in Figure 5.12.

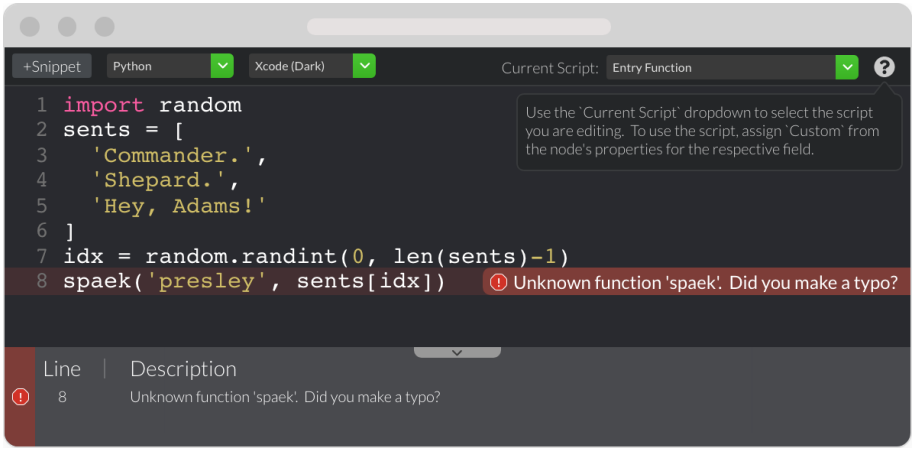


Fig. 5.12 The Scripts window showing a popover hint (top-right) and error panel (bottom).

Simulator variables. In the Simulation window, variables that impact the outputs or sibling triggers are highlighted in yellow, the identification of which was tested in task 32. The second participant was unable to identify this relationship, but upon further discussion, this was a misinterpretation of colors. It was decided to not make changes until a further case occurred, but no other participants encountered this trouble, and as such this case was deemed an outlier.

Simulator variable states. In the Simulation window, the state management of the variables caused confusion and underwent significant rework. In task 33, participants had to assign an existing state, create a new state, overwrite a modified state, delete an existing state, and restore variables to their default values using a small desktop application that mimicked the states panel of the prototype. Initially, assignment, creation, and overwriting were all done using a single editable combo box, with deletion requiring a button press after a state was assigned. The second participant found the multifunction combo box confusing and struggled to create and overwrite states, but action was not taken due to insufficient information on how to make an informed change. While the third participant did not encounter such problems, they did suggest adding an explicit button for saving, and highlighted that deleting should occur without having to first apply a state. A save button was added to the left of the combo box and the behavior of the delete button was modified to show a popover where the user can choose which state to delete rather than having to apply it first. This save button, while functional, is largely placebo, as submitting text into the combo box performs the exact same function as pressing the save button. However, this fix worked, as all following participants were able to save and overwrite without trouble, even though the underlying functionality did not change. Although all following participants were able to also successfully delete states, there was frequent hesitation, which suggested that the functionality of the delete button was not as clear as it could be. Discussion with participant four resulted in the delete button's label being appended with ellipsis, which is often used to indicate further that there are further steps [82], in this case that the button does not delete the currently

applied state upon press. A final change was made following participant six due to further hesitation, where the delete button's label was changed from "Delete…" to "Manage", after which the final participant had no troubles. This modification also allows for further expansion of state management rather than just deleting while still maintaining a low visual real estate, therefore being preferable to the previous labels. The final desktop equivalent of this design used for testing interactivity can be seen in Figure 5.13.

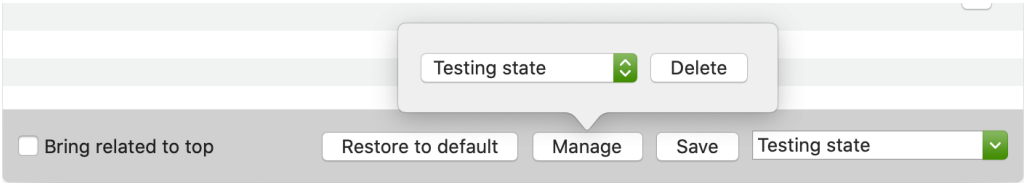


Fig. 5.13 Finalized desktop equivalent of the Simulator Variable States panel.

Recordings. Following the sixth participant, the Simulation window was modified to add a Recordings feature, which was documented in the design notebook, and two tasks were added to the task list that investigated creating a recording and playing it back. This feature was added not because of a problematic encounter, but instead to better support the underlying design principles, particularly the *Principle of Fast Track Testing*, as it enables users to repeat focused testing sessions with a reduction in human error. The RITE process is flexible enough to allow for these modifications to be made during experimentation, as long as there is a follow-up to the change. The Recordings panel, as shown in Figure 5.14, was inserted into the Simulation window below the Entry/Outputs/Triggers panel and did not require modification of any other interface elements. The selection, creation, overwriting of, and deletion of a recording used the exact same interaction paradigm as the refined variable states, including the manage button, save button, and editable combo box. Additionally, for control of recording playback, buttons with standard media control icons were used for familiarity. This Recordings feature was tested with the final participant who explicitly commented upon its visual similarity to the variable states panel and had no trouble explaining and operating the feature.

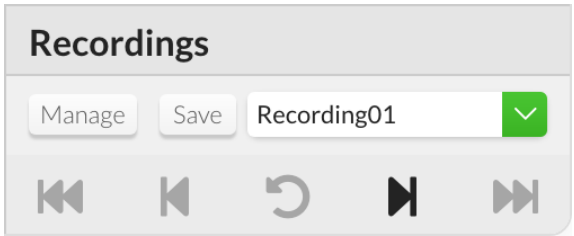


Fig. 5.14 The Recordings panel from the testing harness with a similar layout to the Variable States panel and a familiar media control panel for playback of loaded recordings.

5.4.4 Refined Design Principle Support

As the refined design fundamentally builds upon the chosen candidate design, all points regarding principle support and best practices for design as discussed in §5.3.1 remain true. Moreover, during the RITE process, each alteration to the design and each newly added feature adhered to the same best practices for design.

The final refined design and accompanying design notebook (Appendix C.5) support the concepts of each principle in several ways beyond the chosen candidate design. The relationships between the principles and the refined design discussed below are examined in finer detail as part of the preparation for the study that follows in §6.2.1.

The principle of *Metaphor Testing* is better supported through refined visual attributes of nodes which help to identify specific instances, which in turn could aid in making sense of story structure through enhanced context and clarity. Moreover, connectivity between nodes is colored based on the target, which helps to identify what kind of node follows at a glance, further enhancing contextual understanding of a node's place within the overall structure. As described in the design document, connections are also improved over the candidate design in two major ways. Firstly, a special intersecting symbol is inserted between conditional connections to help distinguish them from unconditional connections, which helps to increase the clarity of the dynamic nature of the structure at a glance. Secondly, when selecting a node, its outgoing connections highlight and animate, and when selecting two nodes, all paths between them highlight and animate. These two animation techniques draw attention to the connections in question and provide a better visualization of the story structure within a specific context, particularly in densely connected or otherwise messy graphs.

The principle of *Fast Track Testing* is further implemented by adding the ability to begin testing from any node via its context menu or through its item in the Outliner, which means that unimportant content for a particular testing session can be skipped over, reducing the need to test from the beginning of the story each time. This can enable more focused testing sessions which allows for more rapid testing as previous unrelated content need not be traversed each time. Moreover, allowing for manipulation of the story state as well as management of predefined states enable more focused and rapid testing, as multiple scenarios can be examined within one testing session. The addition of recording and playback of testing traversals also supports implementation of this principle as users are able to repeat focused testing sessions quickly with a reduction of human error in comparison to manually repeating traversals, particularly as they grow in structural complexity, size, and number of iterations required to fully test.

Support for the principle of *Structure* is improved over the candidate design with the introduction of visual grouping tools to collect arbitrary nodes together within a common region. These grouping tools do not impact the structure of the narrative, but instead provide further ways for authors to personally organize their graph contents beyond placing nodes in close proximity or using shared node attributes to express relatedness.

The principle of *Experimentation* is better supported with the expansion of unique node shapes and colors for each type of available node, as well as other exposed attributes on the nodes, which help to prevent focus being spent on identification of types of nodes which may slow or otherwise hinder the experimentation process. The improvements to connections and structural organization mentioned above also help this principle, as they similarly prevent focus from being diverted to figuring out and understanding the connections between nodes, allowing for experimentation to remain at the forefront. Furthermore, the ease of creating and managing connections was significantly improved prior to and throughout the refinement process, which helps to support the principle by minimizing the cognitive friction required for experimentation, meaning that the author can focus on the actual restructuring and reorganizing of connectivity in the experimentation process rather than have to deal with tedious connection management.

Support for the principle of *Branching* in the refined design benefits from the reduction in complexity and general usability improvements of creating and manipulating connections, as well as the visual improvements made to help with understanding of structure and relatedness of contents at a glance. These two factors are important when working with branches, as in calligraphic graph-based systems, branches fundamentally come from the authoring of links between nodes. If such a process is tedious or difficult to understand, then branching quantity and quality may also be impacted.

The principle of *Contented Authoring* is targeted within the refined design in two ways. Firstly, individual nodes do not associate themselves with specific functionality beyond providing structural positions for Logic to be executed. That is, there is no concept of a dialogue node, or a text presentation node, but rather structural hooks at which any code can be executed. One of the benefits of generalizing this is that authors may be less likely to become locked into repeatedly using the same kinds of nodes, as each node's actual functionality differs based on what code it runs. The design also provides several ways of completing each task, which could help to break repetitive workflows by providing variation whilst still achieving the same outcome. For example, a node can be added to the graph by using the graph's context menu, by clicking the node toolbar buttons at the bottom of the interface, by dragging from the node toolbar buttons into the graph, or by copying and pasting an existing node.

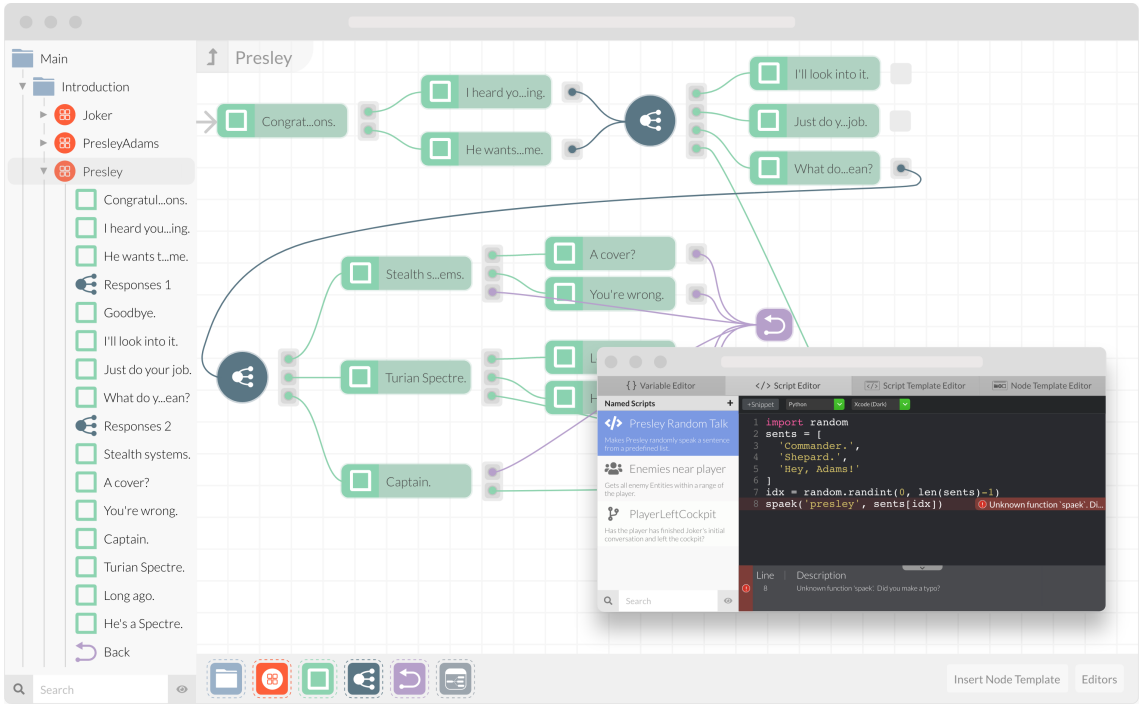
5.5 Conclusion

In this chapter, a gap was identified in existing authoring tool research concerning formal UX design pipelines to accommodate the design and development of user-centered authoring tools. An informed design pipeline was then presented that focuses on authoring tool UX design. Following this pipeline will cover identification of potential users and requirements of the tool, exploration and ideation of early designs, and refinement of the designs to be ready for implementation, at all stages considering the user. The potential

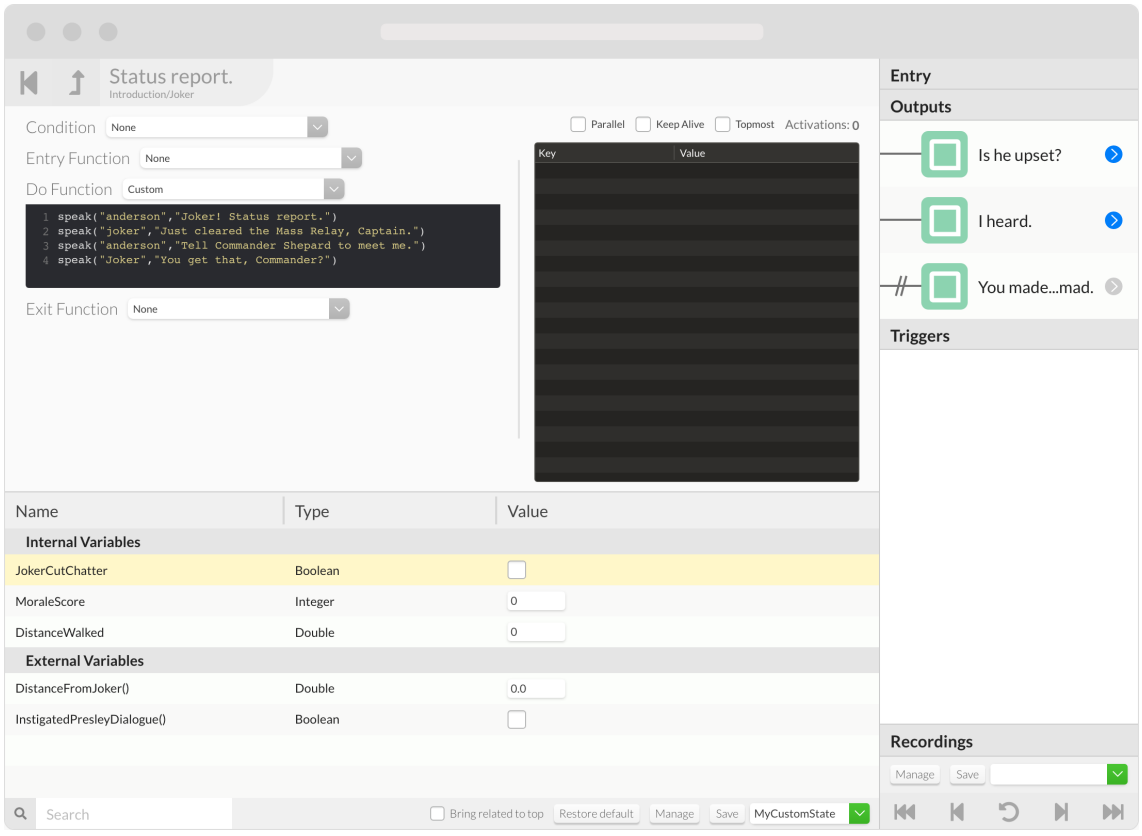
of this pipeline was demonstrated with a case study of creating a design, from scratch, for the *Novella* authoring tool, based on the *Novella Narrative Model* from §3.4. Both the early candidate designs and the refined design followed established design practices and implemented the concepts of the principles from §4, as was discussed in the chapter. The relationships between the principles and the final refined design are examined in finer detail as part of the preparation for the study that follows this chapter in §6.2.1. Figure 5.15 shows two screenshots from the refined output from the pipeline demonstrating the main user interface with an editing window open (a) and the testing harness (b).

Among other benefits, this pipeline enables the identification of potential usability problems and often provides enough information to make rationalized fixes. These problems are usually those that would otherwise go unnoticed, particularly if creating the design without involving users. For example, the *Scripts* window in the *Novella* authoring tool is core to the underlying model and therefore should be clear. To the researchers, the initial design appeared acceptable, but researchers (and developers) are not the end-users. The *Refine* stage revealed that users found the design confusing, which ultimately resulted in the user experience of this particular feature being refined based on user feedback and expectations. Without this process, it is likely that the unclear interface would have gone unnoticed, resulting in a poorer user experience.

The pipeline presented here is composed of various research methodologies that I believe are generally best suited for most authoring tool development processes. However, as established earlier and demonstrated in the existing pipelines analyzed, the specific methodologies used will depend upon the context of the tool being developed. The intent of this pipeline is not to insist on a single UX design pipeline but rather to confirm their value and promote user-focused design processes in the design and development of interactive digital narrative authoring tools to make for a better user experience.



(a) The main authoring tool user interface with an editor window open.



(b) An in-progress testing session in the built-in testing harness.

Fig. 5.15 Two screenshots of the authoring tool design pipeline output showing the main user interface with an editor window open for editing reusable Named Scripts (a) and the built-in testing harness showing an in-progress testing session (b).

CHAPTER 6

Principle Refinement & Validation

In §4, several design principles were identified based on observational evidence and interviews with authors using a representative selection of interactive narrative authoring tools that presented a variety of interface paradigms. As detailed in the corresponding section, steps were taken to ensure that the representative selection of authoring tools used in the study was reflective of the greater space of available authoring tools in terms of features and interface paradigms. Despite the careful selection process, it remains possible that the design principles were reflective of the specific authoring environments used within the study.

This chapter describes a study to validate, further explore, and expand upon the design principles outside of the environments that they were initially discovered in by employing them in the prototype authoring tool design from §5 and evaluating them with a bespoke user study. By doing so, the principles can be evaluated in new contexts distinct from yet related to the original, which provides the opportunity to investigate the robustness of the principles and to appropriately refine and iterate upon their supporting evidence where required based on new data gathered from the study. This refinement process also works to mitigate the chance that the design principles were particular to their specific authoring environments and demonstrates their value as guidelines by implementing and evaluating them in new contexts.

This study, being an extension of the work on the principles identified in §4, targets the third research question of this thesis by further exploring and refining the proposed principles for video game narrative authoring tool design into a referenceable listing.

6.1 Principle Refinement & Validation Framework

Further exploration and expansion of the supporting evidence for each principle can be achieved by applying the principles to a newly created authoring tool design. Incorporating the principles' concepts into a new authoring tool design developed from scratch permits the collection of many principles into a single artifact to be presented to participants. Additionally, as the concept of each principle flexibly describes interaction paradigms and potential author responses rather than a constrained and tightly controlled

specific interaction, applying them to an entirely new design allows for the emergence of new supporting features for each principle's concept that may not have been present in the original three authoring tool environments. This is in line with other heuristics and design guidelines as described in §2.5 which all present general concepts that can be applied to a wide variety of design scenarios beyond the contexts within which they were originally identified. The refinement of these principles and their supporting evidence comes from mapping the principle support onto testable queries that can be evaluated with experimental data.

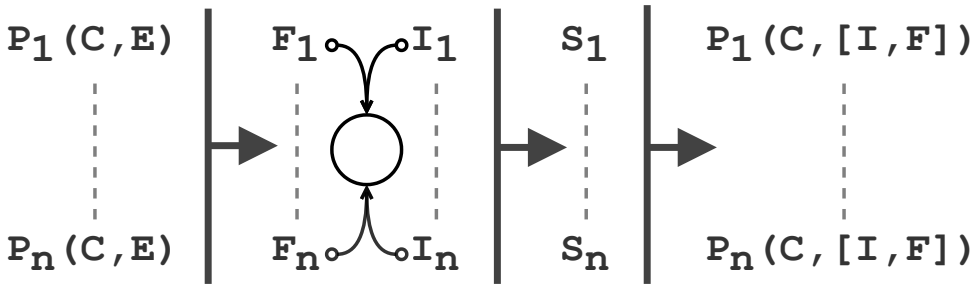


Fig. 6.1 An overview of the principle refinement and validation framework which takes a set of initial principles (**P**), derives pairs of interactions (**I**) with features (**F**) of the design, maps testable statements or queries (**S**) between them, and outputs refined principles.

A general form of this process is displayed in Figure 6.1. The ranges from $1 \ldots n$ are separate for all blocks within the diagram. The initial set of principles (**P**) are each composed of a description of the principle's concept (**C**) and existing evidence (**E**) supporting that principle as described in §4.

The user experience of an authoring tool designed using the principles can be broken down into pairs of interactions (**I**) and features (**F**). A feature is an element or aspect of the design, and an interaction is the way in which the user interfaces with it. Features may be something explicit that authors choose to interact with, such as a button, but may also be implicit and passively utilized, such as an assistive visual indicator that suggests an affordance. Similarly, interactions can be defined at varying levels of granularity and do not necessarily have to describe an isolated action. For example, we may consider a graph feature and all of the various actions that it affords as a singular interaction-feature pair, where in this case the interaction is a compound of the possible actions we are interested in. The way in which an interaction-feature pair supports a given principle is largely down to the informed judgement of an individual as to whether the combination supports the concept that the principle represents. This informed judgement should always be reasoned based on either the existing evidence of the principle, or where not possible, such as when incorporating something that is not included in the existing evidence, reasoned based on how the pair relates to the concept of the principle (e.g., noting how an existing technology or UX axiom supports the concept, or how a particular design allows for a related behavior). A reasoned mapping is created between each principle and a number of interaction-feature pairs that potentially demonstrate

a concrete example of how the principle can be implemented or supported within an authoring tool environment.

Sensors (**S**) are statements or queries that examine an interaction-feature pair in relation to one or more principles that underpin it. When populated with data gathered from the study — such as by surveys, interviews, and observations — they may support, refute, or otherwise demand alteration of the individual interaction-feature pairs with respect to their mapped principle. As data populates the sensors, a principle's collection of interaction-feature pairs is manipulated to refine the individual relationships and overall mapping. This results in an evidenced collection of interaction-feature pairs for each principle. Consequently, once all alterations have been made, the principles' overall concepts may require altering to accommodate the newly refined evidence.

The output of this process is a set of principles (**P**) made up of a refined set of interaction-feature pairs (**[I,F]**) that demonstrate, with evidence from the study, a set of concrete ways in which the principle's concept (**C**) can be supported within an authoring tool. Each principle's concept is also potentially altered to reflect the new supporting evidence where appropriate.

6.2 Study Preparation

The principle refinement and validation framework requires several artifacts beyond the already proposed principles — an existing authoring tool design that implements the principles, a mapping of user experience interactions between the authoring tool's design and the principles, a new authoring tool design demonstration that enables participants to experience the principles in a suitable context, and the methods of gathering data to populate the sensors for analysis. This section will discuss the creation of each of the individual artifacts that were involved in this study. How each artifact was utilized in the study proper is deferred to the methodology discussion in §6.3.

The prototype authoring tool design created and refined in §5, which had the principles considered during its development, was used in this study to generate the user experience interaction mappings. However, given that the prototype authoring tool design was highly tailored and contextualized in both its narrative contents and its intentionally constrained interactivity, it could not be directly reused within this study. Instead, the resulting design notebook (Appendix C.5) and prototype authoring tool design were used as a base to build derived interactive authoring workflow experiences that better demonstrate the principles in a context suitable for this particular study. It is important to note that this process does not alter the original design, instead changing the contents of the design and the specifics of the interactivity within the prototype.

To gather data for this study, the interactive authoring workflows were evaluated within a user study consisting of a survey and interview for each participant as dis-

cussed below in §6.2.3 and §6.2.4 respectively. This provides the sensors in the principle refinement and validation framework with data for analysis as presented in §6.4.

6.2.1 Interaction-Feature Pair Mapping

The original principles identified in §4 were defined with a broad concept that captures the essence of their supporting empirical evidence. When designing an authoring tool user interface based on these principles, the interaction-feature pairs attributed to a principle are based on the designer's own interpretation of a principle's concept. In this study, the supporting interaction-feature pairs stem from existing empirical evidence from the study in which they originated (such as how spatial hypertext graphs complement testing from the *Principle of Metaphor Testing* or how jumping to any state of the story enables rapid testing from the *Principle of Fast Track Testing*), and any additional supporting combinations that are not derived from this evidence are reasoned (such as by relating the pair to existing theory or describing how a particular combination allows for a related behavior) and maintain a close relationship to the principle concepts. These additional supporting combinations are a result of the emergent process followed when designing and refining the prototype authoring tool design in context of the principles which enabled new interaction-feature pairs to become apparent. This is because the design introduces new features that were not present in any of the three authoring tools used in the original study and the design often builds upon features that were present, meaning that new supporting methods for the principles emerged that were not discovered in the original study. This mapping is described below, where each principle is broken down into supporting interaction-feature pairs alongside detailed descriptions of how they are enabled within the prototype design, each accompanied by a generalized statement in italic that describes it beyond its expression within the context of the prototype. These individual mappings are referred to as a Principle's *points*. Naturally, this involves terminology specific to the prototype design, which can be found in either §5 or in the post-study design notebook listed in Appendix C.5. It is important to note that these points do not represent a closed listing upon which no further points could be added, but instead serve as a practical listing of points based on the prototype authoring tool design. It is possible and expected that future researchers could expand upon these principles and their points.

Metaphor Testing

Point 1

Spatial hypertext graph design enables authors to visually make sense of the story structure.

The Canvas uses a spatial hypertext graph design with nodes and lines, which was found in the original principles study to be a major contributing factor to this principle due to the intuitiveness reported by participants, as well as empirical evidence of participants

using node and line graphs to read through the structure without explicitly testing. This is a foundational aspect of spatial hypertext and is already understood to provide a solid overview of the underlying structure, making it an ideal base for this point [18, 190].

Point 2

Visual attributes of a node assist with identifying the instance and function of the node, which can aid in making sense of story structure through enhanced context.

Another factor in the original principles study was the content of the nodes providing contextual information which aids in understanding the structure. Most nodes in this prototype have three different display modes, allowing the user to choose how much contextual information is provided to them at the expense of visual real estate. All node display modes contain a type-specific icon, a color scheme, and a user-specified label which help to identify the function and instance of the node at a glance. One of the display modes adds a user-specified description to the node, providing additional context given that the user has entered a descriptive text. A special display mode is available to non-leaf nodes which displays them in Artboard form, meaning that a read-only preview of their contents is displayed, making it easier to observe the potential impact that a particular node has upon the flow of their neighboring structure.

Point 3

Clarity of connectivity between nodes helps with visually following and making sense of the story structure.

Clarity of connections is also of importance, as if users cannot determine connectivity, then the effectiveness and intuitiveness of a spatial hypertext graph design can be impacted, particularly as the link is a foundational element of hypertext which spatial hypertext is built upon. An important factor of the original principles study was that participants frequently complemented or otherwise desired graphs for visualizing connectivity. In this prototype, nodes have a distinct floating connection panel with thick connecting lines, showing clear connectivity between elements. The connection panels, while floating, make use of the Proximity law from the Gestalt Laws [147] to ensure that users determine them as related regardless of having physical separation.

Point 4

Coloring connections based on their target helps to identify what follows without seeing it directly and therefore can enhance local contextual understanding of the structure versus plain and uniform connections.

Lines between nodes are colored based on their target node type which helps to further identify what type of node comes next without actually observing it. This is especially useful when a node's outputs do not all fit on the screen. For instance, if a link that goes off-screen is purple, the user will be aware that purple links always result in a Return

node, and therefore additional context is provided as the user does not need to scroll to find out what type of node is at the end.

Point 5

Distinguishing dynamic connections from regular connections helps to understand the potential pathways that can and cannot be taken from a node, which in turn increases local contextual understanding of the story structure.

Connections between nodes that are conditioned (either on the link itself or the target node) are overlaid with double slashes to indicate that a connection may be dynamically activated, providing more context about potentially blocked pathways through the flow of the structure.

Point 6

Highlighting and animating outputs toward their target nodes helps to better visualize, follow, and understand structure and flow when dealing with densely connected or otherwise difficult to read graphs.

When a node is selected, all of its output links become highlighted, dashed, and animate toward their target, which helps to identify connections between nodes. This feature is especially important to mitigate dense graphs where many nodes and connections overlap, making it difficult to follow the flow, which is a common occurrence in flowchart-like designs. With this method, selecting a node means that its connections are clear regardless of node density, connection density, or any complex overlapping.

Point 7

Visualizing all pathways between two nodes helps to better understand the structure and relatedness between the two nodes and also helps to identify problematic pathways.

When two distant connected nodes are selected, all pathways between the two nodes become highlighted, dashed, and animate toward their target also. This helps in identifying all pathways from one node to the other, and can help to not only understand the structure better without testing, but also to identify any unwanted or desired pathways based on the state of the lines (e.g., if a pathway is highlighted but should not be a valid method of traversal between the two selected nodes, then the user can investigate why the connections exist).

Point 8

Support for traversal methods is essential to being able to make sense of the story structure, especially with large projects where visual space is limited and elements are frequently off the screen.

Empirical evidence in the original principles study showed participants traversing graphs that didn't fit into view when visually testing their story, and as such, supporting common traversal methods is vital to let users see the complete extent of their structures. This

prototype design supports panning, zooming, and fitting the view to a selection, which follows established traversal paradigms. Within the underlying model is also the ability to traverse up and down the hierarchy, which is enabled in the prototype using buttons on Artboards, context menus, buttons on the Canvas root, or the Outliner panel, each of which were refined in the RITE phase and further support efficient traversal.

Fast Track Testing

Point 1

Being able to begin testing from any node means that unimportant content for a particular testing session can be skipped over, reducing the need to test from the beginning each time, which means that testing sessions can be rapid and more focused on a particular point as prior content is not traversed.

Within the original principles study, participants were frequently observed initiating testing from arbitrary locations and focusing their testing around a particular part of their story. In the prototype design, testing can be entered into at any point by context clicking on a given node and selecting the appropriate menu item. Alternatively, context menus of items in the Outliner can be used in the same way. This ensures that testing can begin at a given point (in contrast to from the very beginning) and can be instigated in multiple ways. As testing can be started at an arbitrary node, the sessions can remain more focused and be quicker as prior unwanted content does not have to be firstly traversed.

Point 2

Allowing users to control the story state during testing means that testing can be rapid and focused as they can test multiple scenarios in one session.

The Simulation Mode variables panel allows for arbitrary configuration of internal and external variables. This is important as it means that the user can simulate modifications to the story state to test for specific setups and their impact upon the story flow. This directly complements testing from a point other than the beginning, as previous choices that will no longer be executed can be simulated instead. Letting the user control the story state enables them to test multiple potential scenarios within a single testing session, making the testing sessions more rapid and focused.

Point 3

Supporting proper management and application of custom variable state configurations allows for more focused testing of specific states and their impact upon the story flow.

Management of variable states is implemented in the prototype and was refined during the RITE phase. It is important to let users properly manage and apply preset state configurations so that they can test the impact of specific setups without having to configure all included variables each time. Doing so lets users focus on testing rather than spending time setting up variables for each testing session. This complements testing

that doesn't start at the beginning, as a particular state can be applied to better focus testing. This differs from point 2 in that it concerns itself with bulk state application rather than experimental manipulation of individual variables.

Point 4

Supporting recording and playback of traversals means that users can repeat focused testing sessions quickly with a reduction of human error in comparison to manually repeating the same traversals.

The Recordings feature allows users to capture and replay specific pathways between two nodes with variable state alterations included. They are able to manage saved Recordings and create new ones. Playback controls are provided to allow traversal through an active Recording, including a reset button for each step if a participant wishes to experiment. During playback, participants are not restricted and can deviate from the Recording's steps, although this will terminate the recording and resume a manual traversal. This behavior is essential to this principle as it enables users to create repeatable testing patterns with desired variable configurations, which allows for highly specific quick and focused testing as they do not have to get to the desired stage nor remember the traversal pathways each time, mitigating human error when repeating testing sessions.

Structure

Point 1

Spatial hypertext graph design helps with personal organization due to its intuitiveness and lack of restrictions in regard to positioning of content.

The Canvas uses a spatial hypertext graph design with nodes and lines, which was found in the original principles study to be a major contributing factor to this principle due to the intuitiveness reported by participants and comments upon its ease of use for personal organization. An important factor of this is that the author has the freedom to place nodes where they want with minimal or no constraints. This is also an aspect of spatial hypertext and is understood to be used for idiosyncratic organization [18, 138, 190].

Point 2

Supporting arbitrary visual grouping tools that do not impact the structure aids personal organization.

Frames let users create arbitrary groupings of nodes that do not impact the hierarchy. This feature, while not present in any of the authoring tools used in the original principles study, further enhances the ability to organize content into pseudo-structures, as participants were observed to do. Additionally, they are prominent in many other node-based software packages and were welcomed by all participants in the RITE study.

Point 3

Support for traversal methods directly complements the ability to personally organize spatial hypertext graph contents.

Supporting personal organization must be complemented with efficient methods of traversal around the spatial hypertext graph, otherwise the methods of organization would become cumbersome due to limited visual space. This prototype design supports panning, zooming, and fitting the view to a selection, which follows established traversal paradigms. Within the underlying model is also the ability to traverse up and down the hierarchy, which is enabled in the prototype using buttons on Artboards, context menus, buttons on the Canvas root, or the Outliner panel, each of which were refined in the RITE phase and further support efficient traversal.

Point 4

Being able to identify types and unique instances of nodes helps with personal organization as it lowers cognitive friction and helps users focus on organization rather than identification.

Organizing nodes into a personalized structure cannot be done effectively if the nodes cannot be easily identified. This prototype uses a variety of attributes on nodes to aid in identifying the type and instance of a node, and each node has multiple display modes to provide more or less local contextual information. This is important as without being able to identify what each node is quickly, organization can become challenging.

Experimentation

Point 1

Spatial hypertext graph design provides greater context within which experimentation can take place.

The Canvas uses a spatial hypertext graph design with nodes and lines, which was found in the original principles study to be a major contributing factor to this principle due to the clarity of structure and connectedness such an approach brings. Spatial hypertext graphs provide visual context and can present a vast amount of structure, especially in contrast to other less visual methods that give a more restricted picture of the story. Spatial hypertext graph design is already known to provide clarity to structure and connectedness in the form of context around individual content, making it an ideal base for this point [18, 190].

Point 2

Being able to identify types and unique instances of nodes helps with experimentation as it prevents focus being spent on identification which could slow or hinder the experimentation process.

Experimenting with structure and connectedness cannot be done efficiently if the nodes cannot be easily identified. This prototype uses a variety of attributes on nodes to aid in

identifying the type and instance of a node, and each node has multiple display modes to provide more or less local contextual information. This is important, as without being able to identify what each node is, the process of experimentation would be interrupted, and focus diverted to the task of node instance identification instead.

Point 3

Support for traversal methods directly complements the ease of experimentation.

Experimenting will often require navigation around the Canvas to understand context or to access nodes, so if traversal methods are poorly supported, the ability to experiment will be impacted. This prototype design supports panning, zooming, and fitting the view to a selection, which follows established traversal paradigms. Within the underlying model is also the ability to traverse up and down the hierarchy, which is enabled in the prototype using buttons on Artboards, context menus, buttons on the Canvas root, or the Outliner panel, each of which were refined in the RITE phase and further support efficient traversal.

Point 4

Being able to identify the connected state of links between content helps with experimentation as it prevents focus being diverted to understanding the connections which could slow or otherwise hinder the experimenting process.

An important aspect of experimenting with structure and connectedness is being able to correctly identify the state of existing connections, otherwise focus is diverted to understanding the connectivity rather than exploring structural options. In this prototype, floating connection panels accompany nodes, and each outgoing connection has a pin within the panels with different states based on its status. Connected pins are filled with a color based on the target node type, and disconnected pins have no fill. This makes identifying the connection status easy, therefore benefiting the ability to experiment.

Point 5

Being able to easily create and manage connections between nodes with minimal cognitive friction ensures that the user can focus on the actual restructuring and reorganizing of connectivity between nodes rather than focusing on the details of managing connections.

An important part of experimenting with connectedness is the ability to quickly and easily assign and reassign connections between nodes, as well as create new links and delete existing links. Being able to manage connections between nodes with minimal cognitive friction means that the author can focus on the actual restructuring and reorganizing of connectivity between nodes rather than on the connection process itself. In this prototype design, connections can be easily made and rerouted using drag and drop interactions from the source pins. New connections can be added in multiple ways (which was refined during the RITE phase), and they can be similarly edited or deleted easily.

Branching

Point 1

Authors tend to organize nodes to visually mimic story flow which makes it easier to identify the impact of branches before or after they are created due to increased local context.

The Principle of Structure is important as it enables users to organize their story such that it visually mimics the story flow, including branches. This was observed in the original principles study and other studies have shown that authors will organize their nodes in this mimicking format [111]. This presents the user with additional contextual information about the relationships between their nodes and therefore allows for visual inspection of the impact of a branch before or after creation.

Point 2

Reducing the complexity of creating and manipulating connections enables easier creation and experimentation with branches.

The Principle of Experimentation is important as it enables users to quickly create and manipulate connections, consequently meaning that branches can be readily created and experimented with, whereas if creating and manipulating connections is tedious, working with branches is also impacted.

Contented Authoring

Point 1

Including nodes with generalized functionality can reduce the chance of authors becoming locked into repeatedly reusing the same kinds of nodes.

This prototype design, which represents a generic authoring tool without ties to a particular runtime environment, has various nodes but none are particular to a specific type of interaction, such as dialogue or other predefined forms of content delivery. The behavior of each node, even Events, is deferred to the contained Logic instead of being fixed with open parameters as a particular node type. This lack of fixed functionality through generalization and deferral means that users could be less likely to lock themselves into reusing the same type of predefined node over and over, as there are no nodes with fixed functionality. Node Templates do not violate this as they are consciously created by the authors with the intention to repeat.

Point 2

Providing multiple ways for users to complete tasks can help to break a repetitive workflow.

This prototype design often provides multiple ways of solving the same task. For example, being able to insert nodes by dragging from the toolbar, clicking the toolbar, or using the context menu on the Canvas. Another example is making a node the root of the Canvas, which can be done from its Artboard, from the node's context menu, from the Outliner's

context menu on that item, or by clicking on the item in the Outliner. This philosophy is followed elsewhere throughout the design. Providing many ways to complete the same task means that the author is given variation as to how they can solve an issue which has two benefits. Firstly, the author can choose a method that is most practical given the context of their current actions, and secondly, it can potentially unlock them from a repetitive workflow by allowing them to take different options each time (although authors may tend to prefer one method over others due to personal preference).

6.2.2 High-Fidelity Interactive Authoring Workflows

As described in the introduction to this section, the refined interactive authoring tool design from §5 cannot be used verbatim due to its intentional constraints and specific narrative content, but it can be used, along with the design notebook (Appendix C.5), as a base upon which to build new contextualized examples specific to this study. To reiterate, this does not equate to changing the design, instead altering its contents.

Consequently, a set of two interactive authoring workflows were created in *Adobe XD* based on the original design, together covering the entire range of principle points declared earlier. Two authoring workflows were chosen over one to allow for different examples to be demonstrated and to break up the size of the workflows to avoid participants becoming overwhelmed or fatigued which could impact the results of the study.

It is important to address the choice to create high-fidelity interactive prototypes instead of actually developing the software from the design for use within this study. Ultimately, the interactive prototype approach provided several tangible benefits over developing the software which was a comparably higher risk to complete.

The sheer time and resources that it would take to implement an authoring tool to a level of stability suitable for use within this study, even for a small dedicated team, introduces a high risk without an equal return on investment. Consider, for example, the *Twine* authoring tool, whose user interface and testing harness are not as complex as the design used in this study, which has nine core developers and at least 34 overall contributors, yet still has almost 700 bug reports, 255 of which are currently open[56]. The authoring tool design contained several advanced features that must be representatively functional for use within this study (e.g., nodes with nested interactive views, an internal simulation engine for the testing harness, a custom scripting system), which further increases both the time for development and the risk of introducing bugs that could disrupt the study and the quality of data gathered. While bugs are inevitable (even in *Twine* with its numerous contributors), in the context of this study, it introduces the risk that participants will focus on the bugs instead of the tasks at hand, especially if they occur during task completion. Despite these disadvantages, using a developed authoring tool does present one significant benefit over an interactive prototype, which is that participants have the freedom to interact with the software in its entirety, whereas in an interactive prototype, they are scoped to a wide yet limited set of interactions. While

this is an acknowledged limitation of interactive prototypes, it does not mean that the collected data will be of poorer quality, and in fact the interactive prototype approach addresses several of the disadvantages associated with actually developing the software.

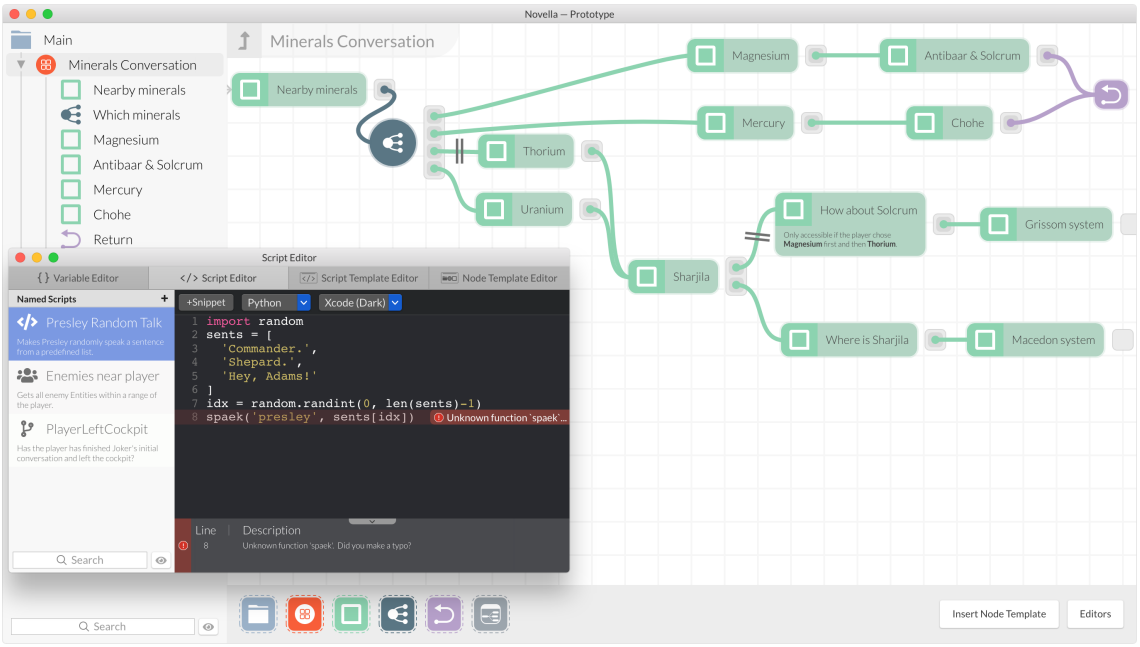
As the interactive prototype created in *Adobe XD* does not involve actually implementing features, adding new features and content is as simple as mocking them up within the program rather than spending time planning and actually implementing them in code. This allowed for more experimentation with the features included in the prototype design as the risk of introducing bugs is heavily mitigated. If bugs such as incorrect interactivity or visual errors did arise, then they were able to be quickly fixed without a high risk of introducing further bugs, as the turnaround for modifications is short as already discussed in §5.4 in the context of the RITE study. It is also worth noting that because these prototype designs are now in a high-fidelity format, they can be considered indistinguishable from or at least highly alike a developed software package, with the exception of complete freedom of interaction. See, for example, the upcoming Figure 6.2, which shows two screenshots of the highly convincing interactive prototype in action. Another tangible benefit of creating these high-fidelity prototype designs within *Adobe XD* is that they can be deployed to the Web and be embedded within a webpage. This allows for the study to be conducted remotely with students and experts anywhere in the world, whereas with an implemented authoring tool solution, this becomes more convoluted and introduces a higher risk to the study (e.g., some participants may be uncomfortable running an unsigned executable on their personal machine, and if done in a cross-platform web framework to alleviate this, the issues of cross-browser development become a factor in stability and equality of each experience in different browsers).

Overall, there are clear advantages and disadvantages to both a developed prototype and a high-fidelity interactive prototype, but in the context of this study, the latter provided a greater return on investment and presented a lower risk without sacrificing the quality of the artifact that participants use.

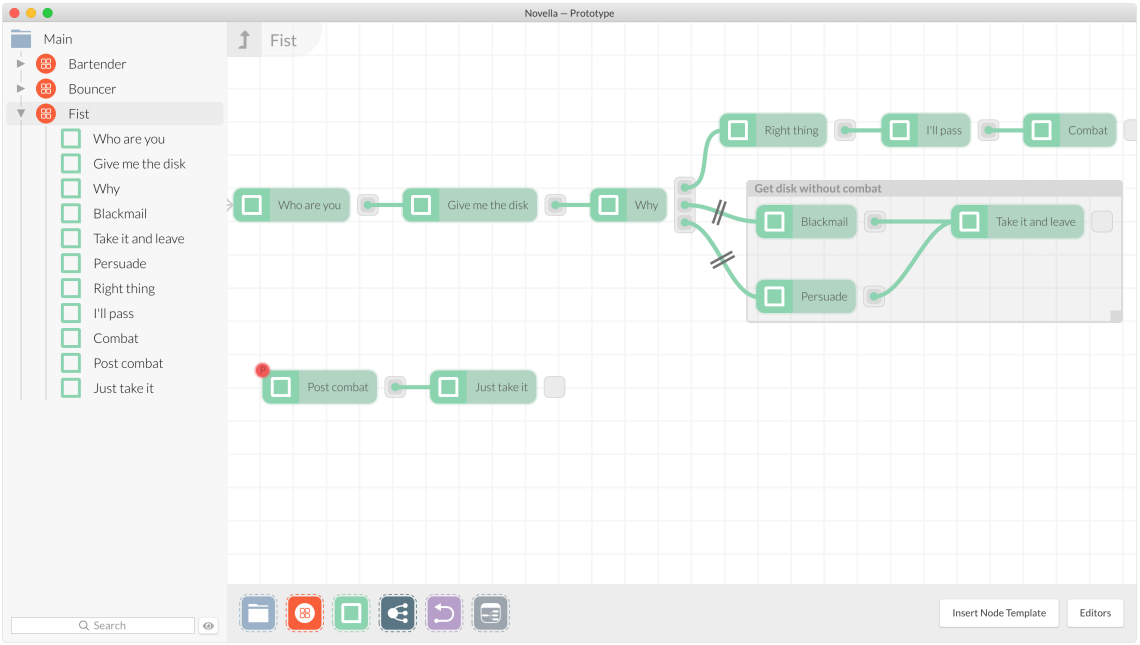
Authoring Workflows

Two separate interactive authoring workflows were created in *Adobe XD*, each targeting a different subset of the principle points described above. As mentioned, these designs inherit from the original prototype authoring tool design with the only differences being the narrative content displayed, the connectivity between screens of content and the hotspots that enable interactivity, and that the user interface elements (the main window, clickable buttons, text boxes, and so on) were altered to mimic an application rather than remain in blockout form, as this design was no longer being actively refined. A single screen from each of the authoring workflows is displayed in Figure 6.2.

The two implemented authoring workflows were abstracted and simplified sequences from *Mass Effect* to ensure that the narrative content that participants were being guided through was authentic and that consequently the interactions with the authoring tool



(a) A single screen from the first authoring workflow which implements a fabricated discourse between the player and an information broker in regard to planets and mineral gathering.



(b) A single screen from the second authoring workflow which implements an abstraction of an actual in-game sequence where the player must gain access to an enemy's hideout.

Fig. 6.2 A single screen from both authoring workflows created in *Adobe XD* implementing contextualized narratives abstracted from the *Mass Effect* video game.

were reflective of video game narrative authoring. The exact semantics of the narrative content was not of significant importance, as it primarily serves as a contextualizing mechanism within which participants can perform guided authoring tasks.

The first authoring workflow, shown in part in Figure 6.2a, implements a fabricated discourse between the player and an information broker, styled both thematically and after typical conversational logic within the game (hub-like conversation structures, dynamic conditioned conversation pathways). Within this story, the player is asking an information broker about nearby planets for mining minerals (represented by the *Magnesium*, *Mercury*, *Thorium*, and *Uranium* nodes). If the player chooses either *Magnesium* or *Mercury*, they eventually return back to the choice of asking about the four minerals again. If they instead choose either *Thorium* or *Uranium*, then they will continue the discourse without having the four choices presented again. The player's access to the *Thorium* node is conditional based upon the player's in-game level which is controlled through variables and scripts, and the *How about Solcrum* node requires the player to already have asked about *Magnesium* and then have chosen *Thorium* for it to be accessible. When a participant interacts with this workflow, it guides them through several authoring tasks to expose them to various parts of the interface. This includes using the built-in testing harness to preview the story and understand the impact of scripts during gameplay, modify scripts and repeating testing to understand the impact of those changes, saving and loading of playthrough Recordings and variable States, making alterations to the story while Recordings are playing back, and previewing animated pathways.

The second authoring workflow, shown in part in Figure 6.2b, implements an abstracted sequence from within the game where the player must gain access to a character guarded by a bouncer with more than one possible solution. Figure 6.3 shows a structural overview of the implemented story which involves three characters — a *Bartender*, a *Bouncer*, and an evil mastermind named *Fist* — with the player's goal being to get to *Fist* to relieve him of a data disk, but the *Bouncer* prevents this without a password. The player can obtain the password by either charming or blackmailing the *Bartender*. This story makes use of three variables utilized within scripts — a `knows_password` Boolean representing whether the player currently knows the password, a `knows_meetings` Boolean representing whether the player knows about *Fist's* secret meetings with the *Bartender* which is used for blackmailing, and a `charm` integer used to track the player's in-game charm level used for persuading characters.

As mentioned, this story is implemented to have more than one solution. The first solution sees the player start by visiting the *Bartender*. Blackmailing isn't yet available (`knows_meetings` is `false`), but assuming the player's `charm` ≥ 50, they can use persuasion to get the password (setting `knows_password` to `true`). The player then visits the *Bouncer* who lets them in (`knows_password` is `true`). When encountering *Fist*, the option that requires `knows_meetings` is disabled as it is `false`, so the player must instead choose either remaining option to obtain the data disk. The second solution sees the

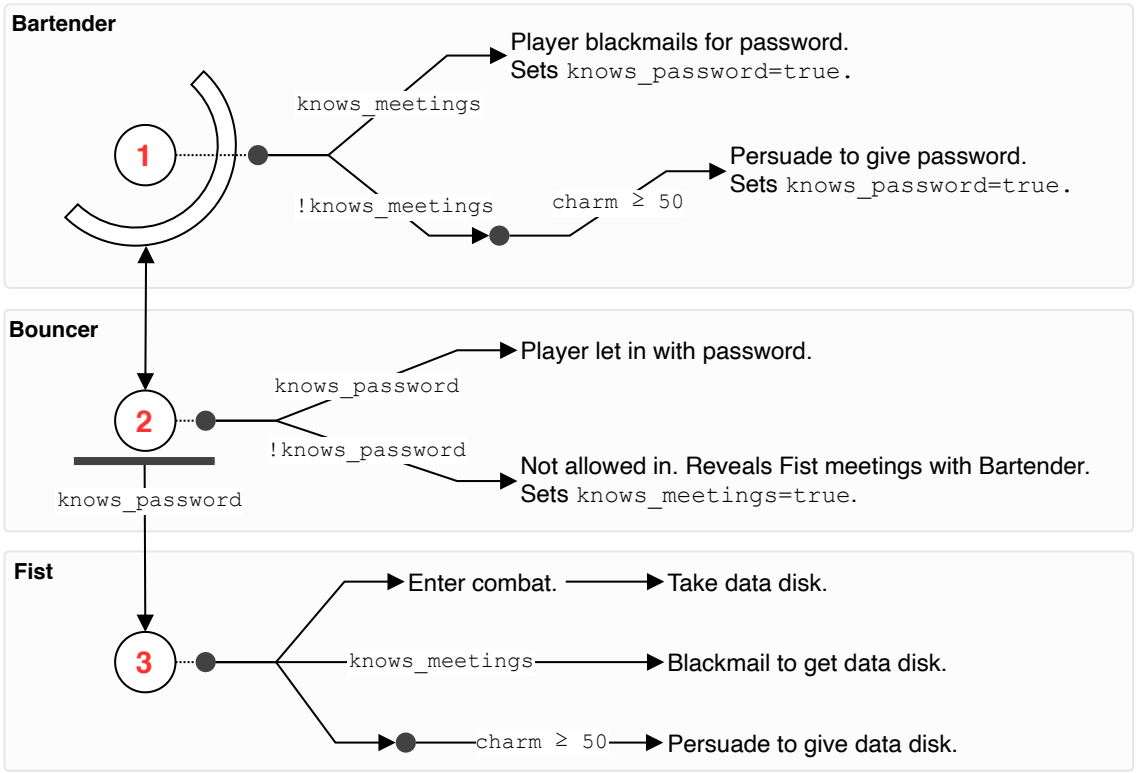


Fig. 6.3 A structural overview of the story implemented in the second authoring workflow based on an excerpt from *Mass Effect*, demonstrating use of and interaction with variables, conditional dialogue options, conditional guarding of progress, multiple endings based on player choice, and repetition of interactions with options differing as choices are made.

player start by visiting the *Bouncer* who blocks them (`knows_password` is `false`) but reveals secret meetings between *Fist* and the *Bartender* (which sets `knows_meetings` to `true`). The player then visits the *Bartender* and is able to blackmail them for the password, then return to the *Bouncer* who in turn lets them in. When encountering *Fist*, all options are available (assuming the player's `charm` ≥ 50). When a participant interacts with this workflow, it guides them through several authoring tasks to expose them to various parts of the interface. This includes navigating around the project, creation of nodes and managing links between nodes, organization of nodes with Frames, modification of scripts, and testing through a prepared Recording.

Workflow Guides

In addition to the authoring workflows, supplementary guides were created for each, serving as informative and instructive text for participants alongside the interactive element. These guides walk participants step-by-step through tasks within the authoring workflows to ensure that they practice a range of authoring activities exploring different parts of the interface. Including a text-based guide alongside the authoring workflows has the benefit of being able to provide extra contextual information and explanation as participants make their way through the content, as well as enabling semi-autonomous participation in that a moderator need not be present for each participant.

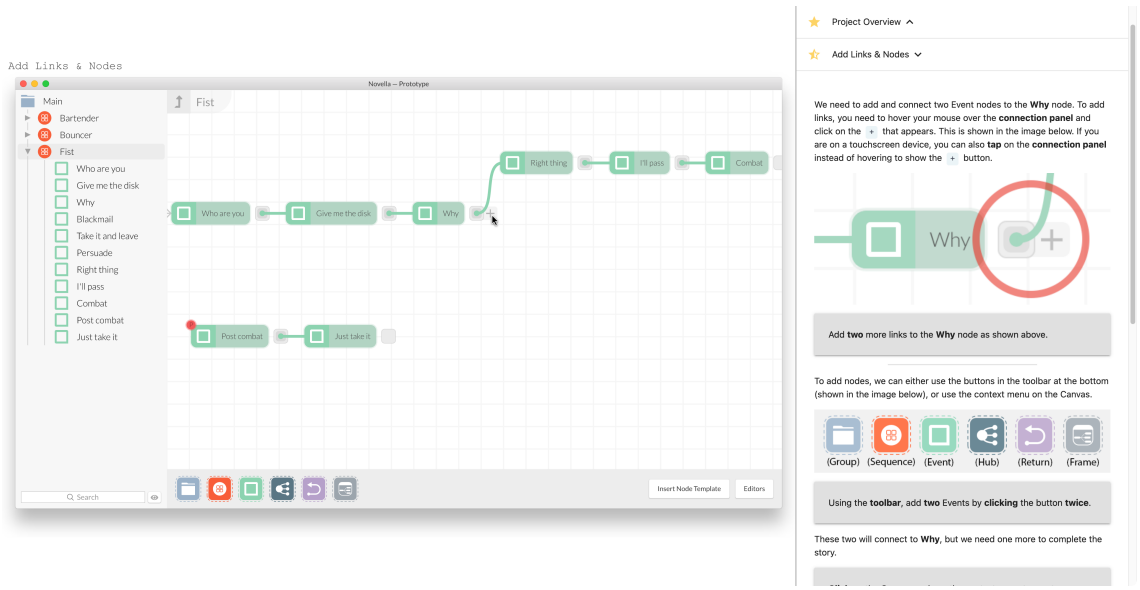


Fig. 6.4 An example of the second interactive authoring workflow (left) and its corresponding sectioned instructive walkthrough (right) embedded in a webpage.

Figure 6.4 shows an example of an interactive authoring workflow and its corresponding sectioned instructive walkthrough used in the study, implemented as a webpage. Each webpage was vertically split into two panels. The left panel embedded the appropriate interactive authoring workflow published to *Creative Cloud Assets*, allowing the prototype to be interacted with in a web browser. Due to the data size of the published authoring workflows, a preloader was added to this panel to indicate that the page was loading. The right panel hosted the corresponding guide containing information and instruction on how to work through the neighboring authoring workflow. The guide panel, implemented as an accordion, was broken up into several sections, each detailing a specific set of actions that the participant must complete. The accordion was configured such that the previous section recently completed was always visible (but collapsed) for reference. Several visual indicators were added to the accordion to assist with understanding the state of each section. For example, to the right of each accordion label is an arrow indicating the collapsed or expanded state of the section. To the left of each accordion label is a star that indicates the completion status of the section, where a hollow star represents an incomplete section, a half-filled gold star represents a section in progress, and a filled gold star represents a completed section. The body text of each section also has distinct elements indicating when the participant should interact with the authoring workflow and in what way, which was explained to them at the start of the study. Due to protection against cross-site scripting attacks, it was not possible to automatically synchronize the authoring workflow and the guide, as the authoring workflow can only be embedded from an external source. To mitigate the potential desynchronization that participants may experience between the authoring workflow and the guides, a label was added to the top-left of the window within each authoring workflow that corresponded with the section headings within the guide accordions. Therefore, if a participant notices

that the labels do not match, they can identify which section they should instead be reading. This was also explained to the participants at the start of the study.

6.2.3 Survey Questions

Based on the principles, their supporting points, and the guided authoring workflows mentioned above, a list of survey questions was created for participants to complete after finishing each workflow. In the principle refinement and validation framework, the survey questions provide, in part, chiefly quantitative and some qualitative data to aid in populating and analyzing the sensors. Due to the large number of principle points, it was unfeasible to simply ask a question about each interaction. Consequently, select points that were closely related were merged to be covered by a single question to reduce the overall total number of questions. This helps to mitigate the risk of fatiguing participants due to an overwhelming number of survey questions following already extensive interactive authoring workflows which could have a negative impact upon the gathered data such as survey satisficing [117][57]. In addition to survey questions targeting the principle points, four demographics questions were added to determine the experience of each participant in creative writing, games storytelling, and use of interactive or game narrative authoring tools. The resulting survey questions were then split into two groups depending on which authoring workflow they were chiefly demonstrated in, which is further described below in §6.3. An optional free text entry field was appended to the end of each group for participants to provide clarifications or additional comments. A complete listing of all survey questions and a mapping to the points that they cover is shown in Appendix D.1.

All but two of the survey questions use a unipolar scale as opposed to a bipolar scale. Those questions are providing data to answer, in part, the extent to which that a particular interaction-feature combination supports the overarching principle. Testing this comes in the form of importance to that principle, effectiveness at supporting the principle, or usefulness toward the principle's concept. Unipolar scales are ideal for this kind of query, as they are designed to prompt respondents to think about the presence or absence of something. They use a zero-based scale with no true midpoint, where zero represents no effect, and the highest number of the scale represents the maximum effect, which maps well to testing an extent. In the case of this study, the five unipolar scale degrees used are *Not at all* *, *Slightly* *, *Moderately* *, *Very* *, and *Extremely* *. In contrast, a bipolar scale, with positive and negative axis set around a central midpoint, does not map as well, as this would instead measure a scale of extremities which does not provide enough information to determine the extent to which an interaction-feature pair supports a principle. On the other hand, unipolar scales naturally measure extent.

There are several other benefits to using unipolar scales over bipolar scales, particularly when surveys contain a moderate amount of questions using the same scale. Studies have shown that repeatedly using bipolar scales increases the likelihood of forms of

survey satisficing such as acquiescence, where respondents incorrectly or untruthfully answer for a variety of reasons including the tendency to be polite or to avoid social friction by becoming more agreeable when answering bipolar scales [36, 118, 188]. Moreover, the inclusion of a midpoint was often found to increase uncertainty in respondents, with a tendency to be more agreeable as a result, or to develop a predictable pattern in response to the uncertainty. One concern with unipolar scales is that respondents may have a different understanding of what each degree means, although it has been found that when each option is fully labeled and that the labels are evenly distributed, respondents will build a reliable understanding of a particular scale degree based on the context of the other labels present in the same scale [118].

The two exceptions to unipolar scales in the survey are questions for the *Principle of Contented Authoring*, which are phrased as agree/disagree bipolar scales, as they are specific examples particular to a user's behavior rather than a measurement of extent of support, and therefore they do not represent effectiveness, importance, nor usefulness. Moreover, it is possible to disagree with the way in which these two questions impact your own workflow, whereas with the questions using a unipolar scale measuring extent, explicit disagreement does not make as much sense.

6.2.4 Interview Questions

Interview questions were created based on the principles and their supporting points. In the principle refinement and validation framework, the qualitative results from the interviews provide the majority of the data to aid in populating and analyzing the sensors. The interview is designed to be semi-structured, allowing for further investigation into points of interest particular to an individual participant. To avoid the interviews being too long, all points within a principle were represented by either one or two broad questions. These broad questions are generalized and do not refer directly to the authoring workflows in their wording, instead serving as a topic for open discussion. By doing this, the participant is encouraged to talk about their own experiences in a range of authoring environments, and where necessary the moderator can raise the authoring workflows in discussion if a participant does not do so themselves. In accordance with professional interviewing techniques [94, 137], an additional question was prepended as a warm-up to allow participants to better prepare for the interview proper. A complete listing of all interview questions is shown in Appendix D.2.

6.3 Methodology

The methodology for this study was approved by Bournemouth University's Ethics department with the approval number 32294.

To fulfill this study's goal of further validating, exploring, and expanding upon the initial design principles, they must be employed within a new authoring tool design and

evaluated in this new context with a bespoke user study. Earlier in this chapter, the four components that make up this user study were outlined, including the interaction-feature pair mapping based on the output design of §5, two newly created high-fidelity guided interactive authoring workflows based on the same design, and appropriate survey and interview questions to gather data about the supporting points for each principle. Together, they are used to perform a user study based on the principle refinement and validation framework proposed at the beginning of the chapter.

This study separated participants into three distinct cohorts, although the actual participant protocol for each was similar. The shared demographic for all three cohorts was people that have an interest in interactive storytelling or video game storytelling and have any level of experience in authoring such stories. Individual cohorts then placed finer restrictions on this broad demographic to ensure clear differentiation between the bands. The first cohort had a specific demographic requirement of participants being video game industry professionals or other interactive narrative professionals, and that they must be able to take part in a remote interview. Participants for this cohort were recruited by approaching professionals in connection with Bournemouth University's Creative Technology department and by contacting professionals in related research communities (ACM Hypertext, ICIDS). This included academics, published authors, and game developers. Participants in this cohort are experts in their field because of their rich previous experience, contributions made, and active status within either the industrial or research space. The second cohort had a specific demographic requirement of participants being students that are trained in and have experience with authoring interactive narrative, and that they must be able to take part in a remote interview. Participants for this cohort were sourced from relevant local university courses including Bournemouth University and Southampton University. This cohort was also offered an incentive £20 Amazon UK voucher to encourage participation and as a reward for their contribution, which was provided by email shortly after completion of their participation in the study. The third cohort, targeted at the wider community with an interest in games and interactive storytelling, had no additional demographic requirement and did not require a remote interview due to being anonymous. Participants for this cohort were sourced by posting invitations on relevant online communities including Reddit (*SampleSize*, *InteractiveFiction*, *Inform7*) and a popular Discord server (*Twine Games*). Permission was obtained from respective moderators prior to advertising the invitations.

Three cohorts were chosen over one to better target the recruitment of participants and to better segment the participants during analysis. Each cohort presents a different level of confidence in the data gathered. The opinions of the first cohort — experts in their field — carry the most weight due to their professional status, whereas those of the second cohort provide valuable opinions but are not as impactful in weight, and those of the third cohort represent a broad anonymous and unknown spectrum of talent which

should be treated as such. Moreover, as mentioned above, members of the second cohort were offered an incentive, which requires them to be segmented.

6.3.1 Participant Protocol

For the first and second cohorts (i.e., those not anonymous online), the study protocol was equal. As the study was conducted remotely, invitations were sent by email with an attached information sheet explaining the premise of the study, the process they would go through if participating, their rights when participating, data collection and management, and in the case of the second cohort, the incentive offered to encourage participation. If a participant showed interest in the study, then they were sent a consent form to digitally sign also by email which outlined the terms for participating in the study. Once a participant had read the information sheet and confirmed their participation by signing the consent form, they were assigned both a unique integer identifier and a unique generated token. Participants then received an email linking them to an instance of the online study website specific to them through use of their unique token. This token system was used so that participant responses could be identified and linked to the unique participant identifiers without relying upon personally identifiable information. This also provides a mechanism for destruction of a participant's data without relying upon personally identifiable information should they wish to withdraw from the study. When initially accessing the study website in a web browser, the participant was greeted by an introduction page which provided a brief overview of the study, a summary of the authoring tool design with highlighting of keywords used throughout the survey that followed, and an optional link to keep the guide open in a separate tab. This was followed by the demographics questions to gauge the participant's relevant experience. The survey proper was then divided into two sets of questions as described in §6.2.3. At the beginning of each section, the participant was linked to the corresponding interactive authoring workflow and instructed to complete it prior to answering the survey questions. Upon completion of the survey, further correspondence with the participant took place to schedule a follow-up interview shortly thereafter. This allowed time for preliminary analysis of the participant's survey responses in order to identify potential questions specific to the participant, but to also served as a break to reduce participant fatigue and impact upon their personal schedule. For the second cohort, their incentive was delivered by email upon completion of their participation.

The online cohort mostly followed the same structure but without an interview, although there were minor differences for survey invitation, dissemination, and content. As described above, due to this cohort being anonymous, individual invitations could not be created. Instead, an invitation template, much alike the email invitations, was created and sent out, with permission, to various relevant online communities. Within these invitations was a link to the study website, but this was open to all and was not guarded by token access. Due to this cohort being entirely anonymous, the introduction page was

modified to include a copy of the information sheet and consent form, and progression to the survey questions was gated by an agreement checkbox which participants must have checked in order to continue. Additionally, a seriousness check question was added at the end of the survey, which has been found to successfully filter out non-serious participants [10], therefore increasing data validity.

Pilot Study

A short pilot study was run with three participants using the above participant protocol described for the second cohort. Note that none of the participants in the pilot study took part in the final study. Although no methodological changes were required, several minor alterations were made to the content of the study. When using the links embedded within the survey to access the interactive authoring workflows, the second participant found it difficult to see the links as they were using the same font and size as the rest of the survey text. This was remedied by making the font for the links significantly larger and colored differently to make them stand out from the rest of the survey text. Similarly, when traversing the interactive authoring workflows, the same participant had begun by interacting with the authoring tool design rather than by reading the instructive guide. This was resolved by adding an introduction screen to the start of each authoring workflow that directed the participant to the instructive guide before interacting any further. The third participant found that the animated lines would sometimes not trigger correctly. Additional text was added alongside the corresponding instructions within the guide to highlight that they may not function, and a video demonstrating them was embedded below in the event that they did not function. Several other spelling and grammar fixes were made, as well as solving a survey timeout problem with *LimeSurvey* caused by the default timeout configuration variable of a PHP session being too low.

6.3.2 Data Gathering

Data was gathered from multiple sources to maximize output of the study. As mentioned above, each participant was assigned a unique identifier which was used to collate all data gathered about them during the study and served as a way of retaining the grouping once anonymization had taken place. All information collected about participants was kept in regulation with Bournemouth University's Data Protection Legislation.

As participants went through each interactive authoring workflow prior to the survey and interview questions, they were exposed to the implemented principles within the design without being explicitly informed beforehand about what the principles or their constituent points are. Participants were therefore adequately prepared for the survey and interview questions that followed having been given reference material in the form of the interactive authoring workflows to better contextualize the questions being asked. However, when answering the survey and interview questions, it was expected that participants would also draw upon their own previous experiences when answering the

questions, particularly in the interviews where they are given more freedom to express their thoughts. If we were to instead restrict the survey and interview questions to exclusively discuss the interactive authoring workflows in isolation, then we risk losing out on otherwise valuable related data about how the authors typically work in authoring tools of their choice, which can provide a great insight into their relevant authoring interactions without the need for direct observation.

The survey that participants complete alongside the interactive authoring workflows provided chiefly quantitative and some qualitative data. As all cohorts performed the same set of main survey questions, all data was merged to increase statistical rigor. None of this data allows for identification of the participant and all aspects of the responses are anonymous. For the first and second cohorts where individuals were directly contacted, a unique token was generated and attached to their unique identifier, which participants had to enter to gain access to the survey, allowing responses to be anonymously traced back to the unique identifier of the participant. To differentiate between the first and second cohorts (i.e., between professionals and students), a custom attribute was attached to each token to determine which group it belongs to. For the third cohort, no anonymization was required as all submissions were inherently anonymous. Both the token system and group attribute are built-in features of *LimeSurvey*, the software used to create and deploy the survey online. The survey data was stored in a protected MySQL database as part of the *LimeSurvey* installation on a private secure and fully patched web server, which was purged once the data had been exported and stored on an encrypted hard drive.

The interviews were conducted by either *Skype* or *Discord* depending on the preference of the participant, with audio being recorded for later transcription. In the case of *Skype*, the audio was recorded using the built-in call recording functionality which makes the recording available to all members of the call and is stored securely on *Microsoft's* servers for a maximum of 30 days at which point it will be permanently deleted. In the case of *Discord*, the audio was recorded using a channel recording bot called *Craig*[58] which temporarily stores the recordings on secure servers, being automatically deleted after seven days, although the recordings were manually deleted as soon as the audio was downloaded. All of the audio files were stored on an encrypted hard drive. Even though the information sheet and consent form for this study outlined the audio recording policy, participants were reminded prior to starting the recording how and why their audio would be captured and what its purpose was. Participants were also offered the chance to withdraw if they were unhappy with this, although none chose to do so. As interview audio data is identifiable, all corresponding audio files were transcribed as soon as possible so that the source material could be safely deleted.

Participant contact details for the first and second cohort were stored only for the purposes of initial communication, sending of a survey link and token, and where applicable, sending of the incentive. Once the participant had either completed or withdrawn from the study, their contact details were erased.

6.4 Results & Analysis

For the first cohort, a total of five professionals were sourced. The first professional is a Narrative Designer with additional writing responsibilities at a well-known UK game studio that focuses on narrative-driven games with minor levels of narrative complexity. The second professional is an established academic and fiction writer with a PhD in interactive fiction research who actively teaches a wide array of students about interactive storytelling. They are also responsible in part for the design and development of a popular web-based interactive fiction authoring tool. The third professional is a prominent hypertext author who has been involved in writing interactive hypertext fiction since the 1980s having produced a number of notable works and having plentiful experience with a wide range of authoring tools. The fourth professional is a Lead Writer at a well-known European game studio recognized for producing extensive narrative role-playing games with great levels of narrative complexity. This professional expresses great interest in the use of authoring tools both in their professional work environment and as a personal hobby. The fifth professional is a prominent figure that has been actively involved in the development, writing, and research of interactive hypertext fiction works since the 1980s, themselves being responsible for several hypertext fiction works and multiple popular hypertext authoring environments. For the second cohort, a total of 15 students were sourced from both Bournemouth University and Southampton University. For the third cohort, which focused on online communities, a total of 14 people had begun the survey, but none had progressed past the demographics stage despite the invitation being repeatedly posted in active communities. While the exact reasons for the lack of participation within this cohort are unknown, it is possible that potential participants were dissuaded by the length of the study, the lack of incentive for this cohort, and that it was administered at the outbreak of an ongoing international pandemic which could introduce stress and consequently less engagement with voluntary community activity. Due to this lack of engagement, the third cohort was removed and the study continued with the data from the first and second cohorts. In total, 20 participants fully completed both the survey and interview.

As mentioned earlier, each participant has a unique identifier. This identifier begins at a value of one and is assigned in order through the cohorts such that the final participant has a unique identifier of 20. In order to easily refer to participants and differentiate between their cohorts throughout this analysis, a respective shorthand label will be used followed by the participant's unique identifier. The five professionals from the first cohort will be referred to as PRO1 through PRO5, such as PRO4 referring to professional four. The remaining 15 participants from the second cohort will be referred to as P6 through P20, such as P9 referring to the overall ninth participant (which is the fourth participant in the second cohort).

6.4.1 Survey Analysis

For the quantitative analysis of the survey data, participant responses were exported from *LimeSurvey* as well as timing information including the start and end timestamps and individual times per section, both to spreadsheets. These sheets were then used as lookup tables based on each participant's unique identifier and token to create a final cleaned spreadsheet ready for analysis. An additional column was added that calculated the total time that the participant took based on the start and end timestamps provided by *LimeSurvey*. The timing data for P7 incorrectly reported a duration of over four days, but after discussion in the interview, it was found that they had left the survey tab page open in their web browser for several days which introduced the error. Based on their individual timings and discussion in the interview, it was confirmed that their time to complete the survey and interactive authoring workflows was actually 30 minutes, and the timings data was manually updated to reflect this. The average time to complete the survey across all participants, including completing both interactive authoring workflows, was 35m 22s. The final spreadsheet was then imported and analyzed using Python.

To better prompt investigation into potentially invalid survey data, a threshold was used to determine whether a participant's completion time in the final study was candidate for inspection. The threshold used was a quarter of the average time that participants took to complete the pilot study (11m 36s). In the survey's demographics and free text blocks, PRO1 repeatedly described how they do not rely upon dedicated authoring tools in their professional work. Moreover, they chose the same option for each of the main 20 survey questions, which strongly indicates survey satisficing[59], and the total completion time was under the minimum threshold by 47s. With their timings removed, the average completion time was 38m, which further indicates a rushed submission. Given that the survey queried both experience with the authoring tool workflow designs and broader experience with authoring tools in general, and that there is a strong indication of survey satisficing, including this data in the survey would be untrue to what was being asked. Their interview data is unaffected by this removal, as they were still able to provide valuable insight into the workflow of a professional writer in the games industry who does not rely upon typical interactive narrative authoring tools. Note that despite this removal, the unique identifiers of participants are unaffected.

Figure 6.5 contains graphs of the demographics included in this survey split between professionals (cohort one) and students (cohort two), showing the estimated experience that each participant has with each topic.

When asked about their estimated experience with creative writing, all professionals reported having more than one year experience, as did 40% of the students. The remainder of the participants mostly reported values fewer than two months, with the exception of one participant who had less than one year but more than two months experience. Given that this study was advertised to students both with backgrounds in games development and media studies including creative writing, this is an expected outcome.

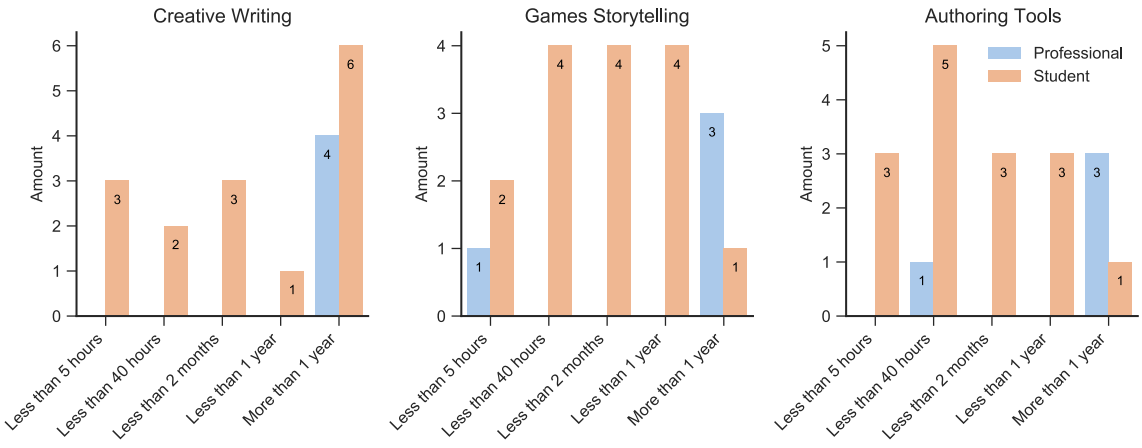


Fig. 6.5 Demographics questions split between professionals (cohort one) and students (cohort two) showing estimated experience that participants had with each topic.

When asked about their estimated experience with games storytelling, all professionals but PRO2 reported having more than one year experience, which is expected as this particular professional is a traditional creative writer who explores interactive fiction as another means of storytelling, but does not necessarily consider their works as a form of games storytelling. Students, on the other hand, chiefly reported less than 40 hours, 2 months, or 1 year, which is adequate and expected given that they are actively studying the topic. The two student participants who had reported less than 5 hours estimated experience were both creative writers, as with PRO2, who had interest and experience in interactive fiction authoring but did not necessarily classify it as games storytelling.

When asked about their estimated experience with authoring tools including those for both interactive fiction and game narrative creation, all professionals except PRO2 reported having more than one year experience, although it is possible that PRO2 underestimated their experience as the subsequent interviews revealed plentiful usage of at least three unique authoring tools. Students were more varied with eight reporting fewer than 40 hours, and the remainder spread evenly between less than 2 months and less than a year, with the exception of one participant who had reported more than one year, although their subsequent interview revealed that this is likely overestimated and is better matched by less than one year experience.

Overall, the demographics of the participants involved in this study provide a range of expertise and experience with creative writing, games storytelling, and prior use of authoring tools for creation of game narratives or other forms of interactive narrative.

Questions within the survey that used a unipolar scale were mapped to integers where the lowest end, "*Not at all* *", was scored as one, and the highest end, "*Extremely* *", was scored as five. For a given question, taking the mean of all of the mapped participant responses provides an average decimal score that indicates the skewness of responses to that question. That is, the closer an average value is to five (as there were five scale degrees), the higher on average it was rated, and conversely, the closer an average value to one, the lower on average it was rated. This single figure provides an overview of the

tendency of participant choice within the survey for a given question and is used later as an initial indicator during detailed analysis.

As the two bipolar questions (agree/disagree) have a defined neutral midpoint, their axes were mapped from zero at neutral to ±2 at the extremes, where negative labels were assigned negative numbers and positive labels were assigned positive numbers. For a given question, taking the sum of the mapped participant responses indicates an average agreement between participants. If the resulting sum is a positive number closer to the maximum positive value ($2 \times 20 = 40$), then this indicates that, on average, participants agree with the statement. If the resulting sum is a negative number closer to the equivalent negative value ($-2 \times 20 = -40$), then this indicates that, on average, participants disagree with the statement. If, however, the value is close to zero, then this can indicate a variety of outcomes where the influence of each axis (positive, neutral, negative) is ambiguous and warrants further investigation. For example, if the resulting sum is exactly zero, then either each participant chose neutrally, or an equal weighting of positive and negative choices was made. If a value is close to but not zero, then this could be the result of either almost equal weighting, a majority neutral with a slight tendency toward one axis, or a mixture of both. This single figure provides an overview of the general agreement of participant choice within the survey for a given question, and like the unipolar mean values, is used as an initial indicator during detailed analysis.

As discussed in §6.2.3, one concern with the use of unipolar scales was that participants may interpret the scale degrees and the relationship between them differently. While evidence shows that this is infrequent and that correct labeling can in fact improve relative understanding of degree, one isolated occurrence of individual attribution of the scale degrees did occur during the study. In their interview, PRO5 explained that they reserved the top end of the unipolar scales (e.g., *Extremely Important* and neighboring) for things that "never occurred to me and would change the field", providing examples of how spatial hypertext or sculptural hypertext were revelations, that the lower end (e.g., *Not at all Important* and neighboring) were reserved for things that were nonfunctional, and the values closer to the center of the scale were used for things considered "pretty useful" for authoring. This has been taken into account when considering PRO5's responses in the analysis. No other participants explicitly described a personal interpretation of scale degree and there is no evidence to suggest that this occurred outside of this instance.

6.4.2 Interview Analysis

As described in §6.3.2, audio from participant interviews was captured and stored. Each audio capture was firstly transcribed and anonymized where necessary before being imported into *NVivo* for qualitative analysis. Within *NVivo*, a Case was created for each participant, and their words from their corresponding transcript were assigned to their Case to make differentiating between my own spoken text and their text easier and to reduce the chance of mislabeling transcribed text as the wrong speaker when coding.

The coding process for the interview transcripts was divided into two phases, with the first phase focusing on making initial sense of the data but without any meta-structure of the coding result, and the second phase further refining and organizing the initial coding result into the final codes used in the analysis.

The first coding phase chiefly uses Exploratory Methods (open-ended investigation of data before a more refined coding pass) as an Initial Coding pass by means of Descriptive Coding (assigning topics to passages of text), Holistic Coding (applying codes to large blocks of text as part of a larger document), and Process Coding (using codes formed of action words), as described by Saldaña [185]. Each transcript was coded in blocks ranging from short phrases to discussions using these methods. Where a block of text demonstrated a concept that had not yet been encountered in the coding process, a Node — an object in *NVivo* for managing codes — was created with a brief title and description, and the block was coded to that new Node. If the block instead fell under the description of an existing Node, then it was coded under that Node, tallying up the various commonalities between the interview transcripts. The block-based coding method also made use of Simultaneous Coding in that the same blocks, or overlaps between blocks, can be present in more than one Node. Additionally during this stage, while progressing through each transcript, if the participant had clarified one of their survey answers or otherwise raised a point of particular interest, an annotation was made on that text block for later reference and use during analysis. The result of this first coding phase is a set of uncategorized trends that are made up of blocks from the interview transcripts, as well as a set of annotations for clarification of survey responses and general points of potential interest to the analysis.

The second coding phase revisited and categorized the resulting Nodes of the previous phase, again using techniques as suggested by Saldaña [185]. This process chiefly used Pattern Coding to identify higher-level similarities between existing Nodes, resulting in the creation of new parent Nodes that summarize topics that contained Nodes relate to (such as all Nodes relating to testing being placed under a new *Testing* Node). There was slight use of Focused Coding during this process by identifying and prioritizing existing Nodes with the most entries (i.e., the topics most referenced throughout the interviews), which influenced how they were subdivided in the Pattern Coding process. Subcoding was also used to reevaluate existing Nodes to see if further Nodes could be derived from their contents, such as by splitting a Node to detail the trend beyond what the initial scope was able to capture. A special case was made for Nodes involving professionals in the event that the Node was unique to them. In this case, a separate Node was created within the parent category that contained all Nodes belonging exclusively to a given professional around a given higher-level topic. This was purely organizational to aid in identifying Nodes unique to professionals during analysis. This second coding phase manipulated the unstructured set of Nodes from the first coding phase into a hierarchically structured and further refined list of trends based on the above processes. The Nodes now followed

a hierarchy where the topmost level denotes broad categories and Nodes nested within increase in granularity based on depth.

For the remainder of this thesis, I will refer to *NVivo* Nodes as *Codes* as it is a more frequently used term in qualitative analysis.

Table 6.1 shows a summary of the Code titles alongside the total number of transcripts involved and the total number of occurrences across all transcripts for each, sorted in descending order by the former. The *Files* column refers to the total number of unique participants who, in their qualitative data, met the criteria for a given Code, and the *Refs* column refers to the total number of occurrences of a given Code across all participants. For more detail of the Codes including their qualifying criterion, please refer to Appendix D.3.

Table 6.1 Hierarchical Code titles alongside the number of transcripts involved (*Files*) and total references across all transcripts (*Refs*). See Appendix D.3 for more details.

Summarizing Title	Files	Refs
Story Structure		
High Level	14	18
Low Level	9	11
Enhanced Lines		
Thinks that animated lines between nodes help to understand connectivity	5	6
Thinks that animated lines between nodes are helpful when zoomed out	2	3
Desire for animated lines to also highlight logical pathways	1	3
Uses enhanced lines to help understand connectivity	1	1
Adding animated lines is technologically taxing	1	1
Users complained when animated lines were removed	1	1
Graphs (Internal)		
Uses a graph to visually follow structure	15	31
Disorderly graphs can be a hindrance	10	18
Uses a graph for a visual high-level structural overview	8	13
Uses attributes to enhance context	8	10
Uses a graph to visually identify logic	6	7
Uses a graph to visually inspect for errors	6	10
Desire to see a graph while editing	4	6
Uses a graph to visually gauge the density of content	4	6
Physical limitations of graphs make them challenging to properly use	3	3
Graphs described as intuitive	1	1
Graphs described as overwhelming	1	2
Nested hierarchy avoids oversized graphs	1	1
Uses a graph to remind them of the structure when revisiting it	1	1
Using a visual graph helps to encourage nonlinearity	1	1
Lack of exposed attributes increases mental bandwidth understanding the story	1	2
Thinks that graph-based systems are efficient	1	1
Uses color to differentiate node function	1	1
Node attributes may desync from their internal content	1	1
Thinks that node attributes allow for idiosyncratic rules	1	1
Using node attributes to represent entity state	1	1

Summarizing Title	Files	Refs
Node Layout & Organization		
Organizes nodes into groups using Frames (or similar)	9	14
Organizes nodes spatially into pseudo-groups	9	10
Organizes nodes in an arborescent structure	7	11
Organizes nodes into nested groups	2	2
Organizes nodes matching their temporal position	2	2
Uses grouping of nodes to indicate a shared status	2	3
Aligns nodes to a snap grid	1	4
Explicitly organizes nodes to mimic hypertext structures	1	1
Organizes node group spacing relative to content size	1	2
Organizes nodes to mimic the environment in the story	1	3
Thinks that node layout is an intrinsic element of the narrative itself	1	1
Using node spacing to represent character relationships	1	1
Mapping (external)		
Plans out the story externally and then recreates it in an authoring tool	6	9
Created an external map due to authoring tool limitations	4	6
Uses an external map to follow structure	4	5
Organizes external map elements into groups	3	3
Organizes external map in an arborescent structure	3	4
Uses an external map to track progress in the authoring tool	2	4
Annotates external map with narrative reminders	1	3
Attempts to mirror the external map layout in a graph-based authoring tool	1	1
Finds discrepancies between external maps and actually implementing it	1	1
Attempts to mirror an authoring tool's graph design in an external map	1	1
Testing		
Testing from arbitrary points of the story	13	28
Testing from the beginning of the story	11	20
Desire to tweak variables while testing	10	12
Testing story by sending to others	5	7
Testing large stories by focusing on specific parts	4	6
Works on small parts and immediately tests them	4	7
Moves variables if they appear prior to desired testing location	3	4
Thinks the Recordings feature is useful for larger or more complex stories	3	3
Playtesting in a game engine	2	6
Thinks the Recordings feature saves time	2	2
Uses built-in grammatical rules for testing	2	2
Wants a clear understanding of the story before starting to test it	2	3
Desire for a way to run automated validity checks	1	2
Desire to observe variables as they change throughout testing	1	1
Thinks that different granularity of testing requires different testing methods	1	2
Uses Recordings to store different potential player pathways	1	1
Uses syntax highlighting to visually check for errors	1	1
Testing the game story by sending it to QA	1	2
Testing the reader's experience with the story	1	3
Finds it difficult to test the use of variables properly	1	1
Would store Recordings for the purposes of historical preservation	1	1
Would use Recordings for debugging and exploration	1	1
Would use Recordings to recover previous playthroughs	1	1

Summarizing Title	Files	Refs
Uses an in-engine interactive debugging tool to test state variations	1	3
Uses built-in automated validation checks to check logic	1	1
Uses in-engine save games to test varying story states	1	1
Uses OBS to record testing sessions for debugging or review	1	1
Using built-in testing loses context	1	3
Usually wants to test from the beginning when working with dialogue	1	1
Considers revising of writing a form of testing	1	1
Runs automated tests for preservation of old hypertexts	1	1
Uses external scripts to automate testing	1	3
Would use Recordings feature to validate the current story revision	1	1
General Authoring Workflow		
Ease of connecting content is important	11	14
Ease of creating content is important	8	9
Works out high-level structures before implementing details	5	9
Inadequate authoring tool functionality limited them creatively	4	11
Being visual and graph-based is important	3	4
Chose an authoring tool based on its genre-specific capabilities	3	3
Ease of content organization is important	3	3
Plans out the narrative in a large Word document	2	2
Tends to work on a detailed section before moving on	2	2
Thinks that connections are hard to follow text-based authoring tools	2	3
Uses writing structures or hypertext patterns intentionally	2	3
Deferring complex tasks can lead to forgetfulness	1	1
Desire for guidance when writing scripts	1	2
Desire for the ability to annotate story structures within a tool	1	1
Desire to know which variables go with which branch	1	1
Ease of access to external media is important	1	1
Ease of navigation is important	1	1
Ease of use is important	1	1
Finds tools with code familiar due to programming experience	1	1
Indecisive authoring methods	1	1
Likes to be able to add custom functionality to script-based languages	1	1
Thinks integration of the story into a game engine is troublesome	1	1
Thinks that natural language tools are like writing in broken English	1	1
Thinks that varying methods of the same task help combat ADHD	1	1
Unclear authoring state causing slowdowns	1	1
Uses autocomplete to speed up making connections	1	1
Uses syntax highlighting to understand structure	1	3
Breaks narrative down into iteratively smaller chunks when planning	1	1
Creates mood boards to assist in ideation	1	1
Narrative content can be impacted by external resources	1	3
Uses internal tracking/documentation over dedicated story authoring tools	1	4
Works on narratives with trivial levels of interactivity	1	2
Being able to visualize links between content is important	1	1
Likes being able to drop into programming for lower level control	1	2
Likes inclusion of high-level programming languages	1	1
Ability for content attributes is important	1	1
Uses a word processor to write a high-level version of the story	1	1

Summarizing Title	Files	Refs
Uncertain of where the story is going during the writing process	1	2
Navigation		
Thinks navigation is more important for large projects	3	3
Large navigation jumps cause confusion	2	2
General Layout & Organization		
Uses colors for content organization	5	6
Organizes Genarrator slides into a custom order	2	2
Separates text-based narratives into different files	2	2
Uses comments in Word documents for story annotation	2	2
Uses styles in Word documents to separate content	1	1

6.5 Themes

Using the qualitative data gathered from the interviews in both their textual form and coded representation, a series of observations was built by performing an inductive thematic analysis on them in isolation from the rest of the experimental data. That is, emerging patterns and relationships — which I will refer to as *Themes* — were identified between the codes. These Themes represent broad and evidenced findings of authoring behavior *exclusively* from the interviews. The Themes serve as a contribution in their own right but also provide a foundation upon which much of the final principle refinement is built, presented in §6.6 below.

Please note that the names of the technologies that participants have used are redacted or replaced below in cases where their identities could be discerned from such information.

Support for external testing is an underexplored factor

Some people send segments of incomplete story to others for testing, but this feature is rarely acknowledged explicitly, which means people must rely on built-in exports which typically package the game as if it were final, which can be a hindrance. By acknowledging that exports are sometimes used for external testing, we can enhance this with better support for debugging, which is currently underexplored.

The *Testing story by sending to others* Code suggests that it is not uncommon for authors to send their work to others during development for external testing. For example, PRO3 discusses periodically sending their story to a small number of testers when "I'm at a point where I think it can benefit", P7 mentions sending "specific bits to people" when testing, and PRO2 explains how they encourage all of their interactive storytelling students to "run beta testing with a target audience" when refining their story.

There are two main ways of testing with an external audience. The first is to have them test the story using a built-in testing harness within the authoring tool, which requires physical collaboration. The second is to export the story so that it can be played external to the authoring tool without the need for physical presence. In the case of the

latter, exporting is typically done by packaging the story for deployment to be run outside of the authoring tool environment (such as in a web browser). By acknowledging that the export process is sometimes used for external testing rather than only for deployment or interchange, we can enhance these specific exports by incorporating essential debugging capabilities beyond that of a story packaged for deployment, giving the tester finer control over the narrative rather than treating it as a finished product.

As this form of testing occurs during development, it is possible that the exported stories will have logical fallacies or incomplete structures. In some cases, this is what is being tested, and the typical export process works well. However, in some cases these apparent errors may be intentional or simply due to the nature of development being an iterative process. For example, if an author wanted to test the content of multiple intentionally disconnected pathways, especially for a creative evaluation rather than testing for logical errors, they would have to, in a worst case, modify the story to have all pathways accessible from within a deployed story, which can introduce human error when undoing the temporary structural changes. However, if we instead provide better debugging support within these exports, such as by letting the tester freely jump between parts of the story, bypassing logical flow, then multiple disconnected pathways could be explored within a single export and therefore does not require the original story to be modified. This kind of behavior is present in design software such as *Adobe XD*, where flows between content can be shared for testing with various export settings, some enabling free traversal beyond logical connections, some enabling comments upon the content, and others locking the tester into the final logical flow.

Built-in testing isn't necessarily representative of the intended story experience

Sometimes authoring tools serve as an intermediary where the story will be experienced in an external framework like a game engine. The built-in testing harness provided by an authoring tool may not reflect this, which results in testing, especially for creative evaluation, being done out of context, which can be detrimental to the author's evaluation of the quality of the story that they are crafting.

While infrequent in the participant sample (PRO4 and P19), sometimes authoring tools are used as an intermediary device to produce content that is ultimately experienced within a game engine. For example, P19 uses the *Inky* authoring tool to write their story and integrates it into *Unity*, and PRO4 uses a proprietary game engine's dialogue authoring tool to create discourse between characters which is ultimately experienced in a game that uses the same engine.

While both of these authoring tools have support for built-in testing, the *Playtesting in a game engine* Code shows that both participants actually test their stories outside of the authoring tool within its final context. P19, for instance, describes prototyping their

story in the authoring tool, and then integrating an exported version of the story into *Unity* for in-game testing. PRO4, on the other hand, explains that due to the complex nature of their dialogues, they "played with having a [built-in] dialogue playback tool", but found that "it just kind of fell out of use". As explained by the *Using built-in testing loses context* Code, the main reason for this was because "in order to really understand how a game is going to feel, we had to play it in the environment that it was going to be played in". PRO4 comments upon how this is down to a lack of context when using the built-in testing feature of their authoring tool, and that how seeing "even just the UI can have a big effect on your perception of the dialogue and the flow of it". They also express that "seeing it in as native as possible an environment to what the audience is going to see is kind of crucial for getting the feeling of it". This is especially true when using a testing harness for creative evaluation rather than for identifying logical errors.

In the case of PRO4 where their team has control over the game engine, they are able to create a strong testing environment with features that would be difficult to implement in a testing harness that is built into an authoring tool in addition to the greater context that comes with being tested in the final environment. For example, the *Uses an in-engine interactive debugging tool to test state variations* Code shows that they implemented an interactive debugging tool where the tester follows a series of prompts to configure and experience the dialogue sequence under different in-game scenarios, and that other team members are able to "give us debug commands that we can enter in a console to trigger certain conditions and try to test the dialogue that way". Similarly, the *Uses in-engine save games to test varying story states* Code demonstrates how testing in the native environment allows for use of in-game debug features such as custom game saves to perform rapid and iterative testing, such as exploring multiple pathways that may have differing activation conditions.

This demonstrates that for an authoring tool that serves as an intermediary where the story is ultimately experienced in an external framework like a game engine, that built-in testing harnesses may not be representative of the final experience, which can have a detrimental impact upon the creative evaluation testing that authors may do, as the work is experienced out of context.

Where authors begin testing from can affect the scale and purpose of testing

The position from which authors start testing (the beginning or at an arbitrary point) reflects the typical scale and purpose of the testing. Regardless of the difference, it is common for authors to use both methods, which means we should not limit where testing can occur from and should strive to aid difficulties that come with starting at arbitrary points.

There is variance in where authors begin testing from and consequently the scale at which testing takes place. The *Testing from the beginning of the story* Code demonstrates

that starting a testing session at the beginning of the story is a popular approach with 11 of the 20 participants mentioning it. The scale of testing in this case depends upon the scale of the story being tested and the point at which the author stops testing. Nine of the 11 participants started testing at the beginning to test the narrative as a whole. P12 described that after finishing a new section they would "go back to the beginning just to make sure it flowed through completely", P6 explained periodically traversing the entire story from the beginning "to make sure it all flows properly", and P8 said how once they had completed a part they would "run through from the beginning to make sure it all worked together properly". Three of these participants expressed concern at the length of the repeated testing, with P20 describing the process as "annoying". On the other hand, two participants said that testing from the beginning was manageable due to the restricted scope of their story content, with P19 describing their work as "short enough to where I was able to play through every route", and PRO4 explaining how their use case was "purely for dialogue" exchanges which are inherently shorter in scope than a complete story.

The *Testing from arbitrary points of the story* Code instead demonstrates the popularity of testing from an arbitrary location within the story, with 13 of the 20 participants mentioning that they do so. The scale of this kind of testing is inherently smaller due to the more targeted approach. One motivation for this is explained by the *Works on small parts and immediately tests them* Code where four of the participants describe testing in conjunction with the writing of small passages as opposed to complete testing from the start. For example, P3 mentioned that they "tend to test as [they] go along" and that they will "often write a branch, test through it, and make sure that is working". Another motivation is suggested by the *Testing large stories by focusing on specific parts* Code where four of the participants describe breaking down larger stories into chunks and testing those individually as opposed to the entire story. When testing from the beginning, the author can achieve states by following the story, but when testing from arbitrary locations, states may never be achieved due to the deferred starting point. Although the *Moves variables if they appear prior to desired testing location* Code only has three participants, it suggests that some authors will attempt to circumvent these limitations in unexpected ways. However, these approaches can introduce human error and do not solve the underlying problem. In the *Finds it difficult to test the use of variables properly* Code, PRO3 expressed difficulty with testing variables in general, and when exposed to this workaround, said that they would "put something visible in the actual text itself" as a reminder, but this solution requires modification of the main story and again introduces the chance of human error. One solution is to support observation and tweaking of states during testing, which ten participants were requesting in the *Desire to tweak variables while testing* Code.

Ten participants were present in both the *Testing from the beginning of the story* Code and the *Testing from arbitrary points of the story* Code, which shows that although the

approaches to testing typically have a difference in scale and purpose, they are both frequently used together by the same author. Therefore, we must strive to support these forms and not limit where testing can occur from. However, enabling testing from arbitrary locations introduces problems with state, and so we should aim to alleviate such difficulties with additional support tools such as state tweaking as is done in the interactive authoring tool workflows in this study.

Authors will find ways of testing their stories that can be unintended or unexpected by developers

Authors will often find ways beyond the built-in testing features supported by developers to test their stories. By understanding and embracing the additional ways that authors seek to validate and appraise their stories, we can further enhance the unexpected or unconventional features used by authors for testing to provide them a greater variety of options for testing their stories.

Authoring tools that choose to implement spatial hypertext paradigms typically do so in the form of a spatial graph which has the advantage of visualizing relationships between content that have a physical representation. As suggested by the *Uses a graph to visually inspect for errors* Code, which contained six participants including two professionals, authors will also use this visualization to inspect their story structure for errors alongside a dedicated testing feature. PRO2, for example, described how their students, in part, use graphs for "diagnostic reasons" in that "it's quite easy to see where a link goes, and if they've inadvertently created a blank frame or a blank screen that they didn't spot, that will appear in the [graph]". PRO3 described a similar concept that they refer to as "proofing the fiction" where they visually inspect the graph to "[verify] that stuff links and goes through the points and that the jumps are there". Other participants made similar statements, such as P12 who said that graphs are the "best way to see your story structure and see whether the links are working", and P7 explained that they use graphs to visually "make sure the links between the nodes are going to correct places" as "you can see if they are connected to the wrong thing". Important to note here is that authors are not only using graphs to understand the structure of their stories but are also using this form of content visualization to actively proof and debug the logical states of their narrative structures. This shows how some authors may adapt an existing feature for testing their story structure, even if it was not intended by the developers of the authoring tool.

Seven of the 20 participants had self-reported experience with text-based authoring tools, and only two of those primarily used text-based authoring tools. Although small in number, the *Uses built-in grammatical rules for testing* Code, which both of those participants were present in, suggests that some authors that use text-based authoring tools rely upon the built-in grammatical structure of the underlying language and how it is integrated within the authoring tool as a form of logical testing. P19 described

"relying on the grammatical rules of the language and syntactical rules of the language" to avoid "any sort of hanging story", praising the authoring tool for raising syntax errors where incorrectly formatted text was entered. P20 similarly complimented the raising of errors based on incorrectly entered syntax. In addition, the *Uses syntax highlighting to visually check for errors* Code, which P20 solely makes up, explains how they also used the coloring of text based on its syntax to visually identify accidental inconsistencies within the structure of the text. These examples further suggest that authors will repurpose a feature to aid them with validation and testing even if it was not intended for that.

In the *Uses external scripts to automate testing* Code, PRO5 explained how they "wrote some unit tests that do Markov walks through the hypertext [that try] to make sure that they don't all loop and that they don't get caught in cycles or places with no links, dead ends, and that they eventually terminate", later confirming that this automated validation runs "as part of the [redacted] build process" when a story is compiled. They explained that the motivation behind this is that "if you misspell the name of an attribute, it may be a syntax error, but if it happens not to be a syntax error, you have no idea" and consequently that "you can't test all of these or you have a combinatoric explosion", hence the creation of predetermined tests. In the *Runs automated tests for preservation of old hypertexts* Code, PRO5 provided an additional use case of automated tests to validate that older hypertext fiction works still run how they originally ran, explaining with a particular example that "there are a couple of other tests that are essentially [validation], often testing behavior that were bugs, edge cases ought not to have existed, but all were used by people who had published hypertexts that depended on them; those tests are there to prevent the bugs from accidentally getting fixed without intending to do it", adding that "we don't want to accidentally break published fiction, and as a result we have various preferences settings where we have something have a behavior that we really wanted to rationalize, but we need to be able to turn off the rationalization for a particular work". While this automation is not a native feature of the authoring tool, PRO4 described in the *Uses built-in automated validation checks to check logic* Code that their graph-based authoring tool has "a debugger installed so that when you save the dialogue it runs through the conditions and makes sure there's nothing missing", which they describe as "handy". P20 explained, in the *Desire for a way to run automated validity checks* Code, that in a text-based authoring tool they would like a button functioning as a "validity check" that would "run it [to check] for any errors" and to raise "things that really don't make sense in terms of programming", which is similar to what PRO4 had described. The presence of automated testing is uncommon in existing authoring tools, as is demonstrated by PRO5 having to develop and run scripts external to their own authoring tool, and that besides PRO4, only P20 requested such a feature. However, the fact that PRO4's preferred authoring tool, which targets video game dialogue trees, contains the feature natively suggests that specific types of author may benefit from such features being built into authoring tools. Moreover, PRO5's additional use case of

historical preservation shows that testing solutions can be used in ways unanticipated by developers, in this case making sure that known bugs that older works may have exploited are not fixed or at least do not alter the behavior of the original work.

Recognize the need for distinct multi-stage planning and building workflows

For some authors, there is a clear distinction between the planning and building of a narrative, and that planning is often done external to authoring tools due to a lack of support for such ways of working, whether intentional or not. Recognizing this separation can help us to build more informed tools that better support authors by integrating in part or in whole solutions to the planning as well as building aspects of the creation of interactive narratives.

Although the *Plans out the story externally and then recreates it in an authoring tool* Code is only made up of six participants, it is able to show that, in part, for some authors there is a clear distinction between the planning of a narrative and the implementation of a narrative. PRO2 explained how they get their students to "create a prototype map on paper first", adding that their students sometimes "do it on big bits of wallpaper, some of them do it with blank playing cards which they move around on the floor and create patterns on the floor until they're happy, and then they take photos of that", and that some "just find free [software] packages online, flowchart creators"; regardless of the method, a prototype structure is created external to the authoring tool. The motivation behind this is to "get them to do all the thinking and [scriptwriting] and playing around with the structure outside of [an authoring tool] first", which they then implement within an authoring tool using the external plan as a guide once they are satisfied with the initial structural draft. PRO4 echoed a similar statement, explaining that "before writing into the actual dialogue editor, my preferred method is to write a flowchart on a whiteboard", with their motivation being that this method is "even faster and even less friction to iterate, copy, erase, and do all that". They further explained that the external prototyping is temporary in that once the dialogue sequence is implemented within the authoring tool, it is permanently erased. P12 also described using "PowerPoint before I even started on [the authoring tool], just so I could organize the different parts of the story and I knew they would all flow in a narrative sense before I then transferred it to the [authoring tool]", further demonstrating the separation of planning and implementation. This concept of planning and creating prototype designs outside of a dedicated authoring environment can also be found in the related field of game design, albeit more formally defined and tightly integrated into the overall game design process, where creators will frequently make use of several advanced techniques for ideation and iterative refinement of designs before properly implementing them in an authoring environment [74].

Four of the six participants that had created external maps did so in part due to limitations of their chosen authoring tool, which can be seen in the *Created an external*

map due to authoring tool limitations Code. For example, P13 explained that they made an external map as "it all got slightly complicated with the one inside of [the authoring tool]". Similarly, P18 described that within an authoring tool, it's not "that user friendly to just start messing around and creating narratives in it because of the connections and lack of visualization", and that in a text-based authoring tool, "trying to type everything out, you'd get lost". However, they also experienced a disconnect between the planned narrative and its implementation within an authoring tool, stating that the narrative may be "all well and good on paper, but when you put it into [the authoring tool], that doesn't actually work". PRO2 also described external maps done by hand to be more efficient for getting ideas down, as "you can do that much more quickly and you can scribble and rub out and chuck away", and that consequently with their students, they "encourage them to do all that planning a bit more along the lines of how you might prepare to make film, script, and storyboard before you start shooting".

Beyond serving as a guide for implementation of a planned narrative into an authoring tool environment, two of the six participants explicitly mentioned using the map to track their progress while testing their narrative implementation, as shown in the *Uses an external map to track progress in the authoring tool* Code. For example, P13 described printing off their map and "crossing off which ways I'd gone" during testing as a way of "seeing where we're going and where we've been".

This evidence suggests that, for certain authors, there is a clear divide between the planning of a narrative and the actual building of the narrative within an authoring tool. On the planning side, authors typically draft structures external to the authoring tools due to inadequacies that prevent them from doing so within that environment. When actually building the narrative, these external plans are often used as guidance or as additional aids when testing. Understanding and respecting that there is sometimes a divide between planning and building can help us to better support developers in future authoring tools. For example, an authoring tool that is focused on building could allow for external references to be imported, so that while plans are still created externally, they can be present within the authoring tool environment for direct reference, or can even be annotated to assist with tracking. A modest inclusion of both planning and building may see scratchpad support for authors to take notes and make simple sketches alongside the primary building functionality. More ambitious authoring tools may seek to fully support both planning and building by incorporating separate modes within an authoring tool where information can be shared between the modes.

Spatial hypertext features that enhance idiosyncratic organization are valued by authors

When working with spatial hypertext systems, features such as visual attributes and the freedom to position nodes enable idiosyncratic rules to emerge. This kind of functionality is valued and widely used by authors. Recognizing the ways that authors create their own rules

will allow us to better our own tool designs, in particular noting that authoring tools are not only a means to create a story, but are in fact a creative space within which authors explore and craft their stories as they are built. Features that may appear unrelated to narrative development, like those that allow for the emergence of idiosyncratic rules and do not impact the story, may actually be more critical than they seem to the authoring experience.

When working with spatial hypertext systems, of which many contemporary authoring tools are, one major problem is the gradual disorder of graphs, making them challenging to work with and understand. The *Disorderly graphs can be a hindrance* Code reflects this as ten of the 20 participants expressed frustration at disorderly graphs impacting their work. For example, PRO4 expressed that they are "anti-spaghetti, anti-spider webs", two terms used to describe the visual appearance of overlapping lines which hinders visual clarity, later highlighting that in a collaborative environment, "[your graph] has to be legible because if people need to debug your dialogue they need to be able to read it as well". They also discussed the addition, upon request of the authors, of specialized "jump nodes" into their proprietary dialogue authoring tool which redirects narrative flow to an arbitrary named point, reducing the amount of required connective lines, which "really helps keep things not spider webby" and "helps with organization". Both PRO2 and PRO5 raised the issue of limited screen space when dealing with larger or more complex stories, with PRO5 expressing that "simply putting everything on one big canvas the way Twine, for example, likes to do, is probably a mistake unless you're writing small things", which further highlights the need for enhancing organizational capabilities as opposed to containing everything in a single view, which many popular authoring tools do. Other participants expressed a more direct impact upon their work as a result of disorderly graphs, such as how P16 explained that "if you just have loads of nodes piling on top of each other, you just can't work out what's going to where", P20 described that a large amount of nodes "would easily just get out of hand" and that they feel an "anxiety of when it gets too overwhelming", and how P7 said that if their graph becomes "a bit of a mess", then "it can also impact how you interact with the narrative when you're testing it if you don't know exactly what's happening or where you're going".

One aspect of spatial hypertext to aid in solving this problem is referred to as visual attributes. The *Uses attributes to enhance context* Code, made up of eight participants, suggests that authors will use these attributes to help make sense of the story structure with the additional context that the attributes provide. For example, P8 stated that when nodes are "all labeled and all had a certain name to them, that makes it a LOT easier than having to open each one to see what I've written", and that when labeling each node based on its internal happenings, that "it makes it a ton easier to go into small sections and edit them and understand where the story is going". In the *Lack of exposed attributes increases mental bandwidth understanding the story* Code, PRO4 explained that when attributes such as titles and descriptions are not exposed or insufficiently represent the node, "the more mental bandwidth I'm putting in to try to understand and remember

what my not exposed information is". PRO5, the developer of a prominent hypertext narrative authoring tool as well as a writer, explained in the *Thinks that node attributes allow for idiosyncratic rules* Code that there are a plethora of possible attributes that can be customized, such as "colors which can be used either as simple tags or as a spectrum or continuum . . . borders which have various properties . . . an iconic badge . . . captions and subtitles . . . a space for flags which is a way of visually representing sets . . . size and shape . . . and positioning". They then went on to explain how authors "adopted their own idiosyncratic custom for what various visual dimensions meant", which demonstrates that attributes are important to authors in a way that may be unique to them despite having no impact upon the actual story. PRO5 provided an example of this personalization in the *Using node attributes to represent entity state* Code where they described in one of their own stories, "when I picked the student that's gone missing, the first thing I did is put a red border around all that link, and then I said with this color scheme that's not obvious enough, and so it became an orange border and that was obvious enough". The *Uses colors for content organization* Code provides further examples of idiosyncratic use of attributes, in this case color, for content organization. P17, for instance, explained that they would "color different answers in different ways . . . if you had aggressive answers or more calming answers, it would sound like something I'd definitely want to color to at least keep separate", later describing similar examples of separating colors based on which character is speaking. PRO4, in the *Uses color to differentiate node function* Code, described a similar use of color attributes that are built into their proprietary dialogue authoring tool in that "if something is an NPC speaking that's going to be a certain color, if it's a potential player response, that's going to be a different color", adding that "you can kind of get a sense of the flow of a dialogue just by looking at the colors of it".

Another aspect of spatial hypertext that helps is the spatial aspect itself in respect to node positioning. The *Organizes nodes spatially into pseudo-groups* Code demonstrates one such usage where nine participants described using spatial positioning and proximity of nodes to create pseudo-groups for personal organization, such as how P16 "had a beginning that had three separate routes" and that they "made sure to put a giant gap between them and then went hierarchically downwards", or how P7 had "arranged the nodes myself into little groups so I could visually distinguish them" which makes it "a lot easier to see where different bits are branching from and where they branch to". Seven authors in the *Organizes nodes in an arborescent structure* Code described their personal layout as tree-like, mimicking the flow of the story visually with variance (but no significance) between horizontal and vertical directionality. PRO4, for instance, described laying out their dialogues in a horizontal tree-like structure, and then using this structure to determine the complexity of the dialogue, in that "the more left to right and skinny a dialogue is, the more conversational it tends to be", and conversely "the more big [vertically], the less a player is actually going to see all of the bits of your dialogue, because these are probably options that they are not getting", concluding that "at a quick

visual glance, the shape of the dialogue can kind of tell you how much reactivity there is or not". PRO2 similarly explained how their students lay out the nodes to "visually look like a family tree", or that they "might look like a wheel with a central hub and lines radiating out from a central point", among various other patterns, so that they can "see their structures in different visualizations". When dedicated grouping tools are provided, such as Frames in the interactive authoring workflows, authors are likely to use them for a similar purpose to pseudo-groups, as shown by the *Organizes nodes into groups using Frames (or similar)* Code made up of nine participants. P12, for example, described themselves as "very visual when it comes to organizing" and states that being "able to see the different chunks and how they all make sense together in their little box as well as seeing it as a whole" is something that they would do. PRO4, on the other hand, uses an authoring tool that does not have dedicated grouping tools, but explains that "it's a feature that we've been wanting", describing it as "really, really cool", and expressing the desire to "clump mutually exclusive bands of information that don't actually interact with each other…particular conditions that all share conditions, those can all go together in a Frame…any kind of information that has something in common". Beyond the high-level concepts of grouping, how authors utilize spatial relationships between nodes for idiosyncratic organization varies greatly. PRO3 explained in the *Thinks that node layout is an intrinsic element of the narrative itself* Code that to them "the organization of the nodes on-screen needs to represent the architecture of the narrative itself" and that "inherent in the narrative is a certain flow or pattern, and I want to see that represented visually when I look at the nodes on the screen", which likewise is achieved through personalized spatial positioning of nodes. The *Organizes nodes to mimic the environment in the story* Code outlines how P18 similarly lays out their nodes "like you're looking at a 2D map of the world", such as "if a room had four separate rooms coming off of that, I would space those rooms as they were in my head and the layout [on the graph] would match that". PRO5 explained in the *Using node spacing to represent character relationships* Code that they sometimes use spatial proximity to define relationships between content found within the narrative primarily as a reminder for themselves as an author. They provided an example from their latest work where they "decide[d] that two of these girls are inseparable and so you put them next to each other and you move them away [together], because otherwise I'll forget", which they describe as "foreshadowing about their inseparability" while the story is still being worked out. They also highlighted that "the reader is never going to see or know about that, but it's a way of reminding myself that there is this issue and it's these two and not some other two".

The evidence above demonstrates three things. First, that spatial hypertext can sometimes become disorderly which hinders an author's workflow. There are many potential solutions to this, but the use of visual attributes and spatial positioning emerged both in spatial hypertext literature [18, 136, 138, 190] and the evidence in this study. Second, that attributes on nodes are used in a variety of ways, and without strict enforcements of

how they can be applied, allow for idiosyncratic rules to emerge regarding the meaning of attributes, which helps with personal organization and is widely utilized. Third, that the freedom to position nodes allows for great levels of personal organization in the formation of pseudo-groups based on proximity or using dedicated grouping tools, and that authors will find unique and creative ways to use spatial properties of nodes to represent complex relationships between the content, either as part of the narrative itself or purely as a personal preference. Many of these parameters have no impact upon the story yet are popular and varied, and consequently it is important to note that we are building a creative space where authors craft their stories, not just a story authoring tool.

Spatial hypertext graphs and additional enhancements assist authors in making sense of story structures

Spatial hypertext graphs assist users in making sense of their story structures, but despite this can sometimes become challenging to work with, especially as stories grow. Additional enhancements to spatial hypertext graphs can bring further clarity, and understanding how authors use them can aid us in refining the enhancements and introducing new ones.

A spatial hypertext graph with content represented as nodes and connections as lines between nodes, as is commonly found in existing authoring tools, is able to help authors to make sense of their story structures. The *Uses a graph for a visual high-level structural overview* Code, which is made up of eight participants including three professionals, demonstrates that authors will use this visual representation of their story to gain an understanding of the high-level structure that they are creating. Both PRO2 and PRO3 described that when zoomed out, they can see the whole structure of their story. Similarly, P18 described zooming out to "quickly visualize the overarching narrative that you're trying to design". The *Uses a graph to visually gauge the density of content* Code further shows the benefit of visualizing story structures at this scale, as some authors visually gauged the density of their story content. PRO4, for instance, explained how "the more left to right and skinny a dialogue is, the more conversational it tends to be", and conversely "the more big [vertically], the less a player is actually going to see all of the bits of your dialogue, because these are probably options that they are not getting", concluding that "at a quick visual glance, the shape of the dialogue can kind of tell you how much reactivity there is or not". Authors also frequently use graphs to follow and understand structure at a much lower level of detail, as demonstrated by the *Uses a graph to visually follow structure* Code which is made up of 15 participants, four of which are professionals. The exact motivations for following the structure vary, but all share a common point of understanding the story at a micro scale and the context around individual elements. As an example, P14 compared following the graph to reading in that you can see that "this links up to there, by this variable, this condition" adding that doing so is "very, very useful", especially "when you revisit it after not looking at it for weeks" as it is easy to

familiarize themselves with the narrative quickly thanks to its visual nature. The *Uses a graph to visually inspect for errors* Code, which is made up of six participants including two professionals, shows an alternative use case for visually following structure to inspect for potential errors. PRO2 explained that their students use a built-in structural map "so they can see what links are going where and whether there's a dead one . . . because it's quite easy to see where a link goes, and if they've inadvertently created a blank frame or a blank screen that they didn't spot, that will appear in the map". PRO3 similarly explained "[looking] at it in the map view and see[ing] that the patterns on screen of movement through the lexia makes sense". Some authors, as suggested by the *Uses a graph to visually identify logic* Code which is made up of six participants including a professional, follow the graph visually to identify logical flow. P10, in response to the interactive authoring workflows displaying double dashed lines across conditioned connections, stated that they "made it more clear that it was a bigger decision than the rest of the decisions" that didn't involve conditional logic. In response to the Return node in the interactive authoring workflows — a symbolized and colored node that indicates a jump without the need for visual connections — P11 complimented the reduced visual clutter making it easier to read, and liked that it reduced the problem to "just a symbol that you can see that says it goes back to this point". This demonstrates how additional visual indicators can enhance context and understanding of the story's structure.

Another enhancement found in the interactive authoring workflows (and a handful of other authoring tools) was that of contextual animated lines between nodes. Five participants, including two professionals, expressed that including animated lines between nodes helped them to understand connectivity, as shown by the *Thinks that animated lines between nodes help to understand connectivity* Code. PRO4 didn't find highlighting outputs from a single node that helpful, but noted that when zoomed out, seeing animated pathways between two distant nodes as "being essential". PRO2 echoed a similar sentiment "especially when you get a much bigger a game or a much bigger narrative and you've got many more nodes and many more links, that would be very helpful". PRO5, the developer of a prominent authoring tool that features animated lines, explained in the *Users complained when animated lines were removed* Code that when attempting to remove the animated lines from the tool multiple times, authors had complained, which highlights the importance of visual enhancements in spatial hypertext that may not have an impact upon the story itself but aid structural clarity.

Additionally, as explained in the *Spatial hypertext features that enhance idiosyncratic organization are valued by authors* Theme, authors will use spatial features, grouping mechanisms, and other visual enhancements like node attributes to create idiosyncratic rules about the structure and layout of their narrative. This enables the authors to enhance the clarity of their story structure in their own way that is unique to them.

The above points demonstrate that the visual aspect of spatial hypertext is used by authors to make sense of their story structure in a variety of ways. The addition of extra

features to the spatial hypertext space such as conditional indicators, nodes that abstract linkage, contextual animations, and use of node attributes further enhance the tools available for authors to increase the clarity of their own structures. By understanding how these aspects help authors to make sense of their stories, we can further refine them or introduce new visual enhancements to the spatial hypertext paradigm.

6.6 Principle Refinement

The main purpose of this chapter was to, through the principle refinement and validation framework outlined in §6.1, further explore and refine the principles that were identified in §4. As part of the principle refinement and validation framework, the initial principles were further explored by assigning interaction-feature pairs to each principle in the context of a refined prototype authoring tool design based upon the initial principles (§6.2.1). These mappings, referred to as the 'points' of the principles, served as a base for the creation of the study artifacts — two interactive guided authoring tool workflows demonstrating the principles and their points (§6.2.2), follow-up survey questions to the workflows (§6.2.3), and semi-structured interview questions to further clarify survey responses and to openly discuss related topics to the principles and their points (§6.2.4).

This section takes the survey data, qualitative coding output, raw interview transcripts, and Themes of the interviews together to appropriately validate and revise the principles and their constituent points. Based on the gathered evidence, points of the principles may be either supported (positively evidenced and consequently unchanged), revised (altered to reflect associated evidence that suggested change), or unsupported (removed due to either contrary or otherwise unsupported evidence). In some cases, points of a principle may be subsequently merged together based on evidence that ties them together into a higher-level concept.

All principles are explored in turn below, each starting with their respective description, followed by a detailed report of each of its points. Similarly, each point will begin with its generalized statement (either original or modified where appropriate), followed by the relevant evidence from the study data. Any changes made to either the individual points or the overarching principles are documented appropriately based on the included evidence. Note that the complete survey questions listing in Appendix D.1 is ordered by principle and lists which points were targeted by which questions. The Themes and refined principles are also presented in a summarized format in §6.7 that follows.

6.6.1 Metaphor Testing

Interfaces that use a visual metaphor to represent story structure and connectedness will result in less testing of non-complex stories.

Point 1 — Supported

Spatial hypertext graph design enables authors to visually make sense of the story structure.

A significant aspect of the *Spatial hypertext graphs and additional enhancements assist authors in making sense of story structures* Theme is in relation to using a spatial hypertext graph to visually make sense of the story structure and flow, which provides significant direct evidence in support of this point.

In the corresponding survey question, 11 participants chose *Extremely Effective* and six chose *Very Effective*. PRO5 chose *Moderately Effective*, meaning that they found it helpful, especially given that during the interview, they referred to the broader inclusion of spatial hypertext as something that they "really need" along with "alternative canvases" for understanding the structure from different perspectives. PRO4, on the other hand, chose *Slightly Effective*, but the interview revealed that this was due to a misunderstanding that the question was about node labels rather than the concept of a spatial hypertext graph interface. Moreover, PRO4 repeatedly praised spatial hypertext graphs throughout the interview, noting that one of their favorite improvements of their company's internal proprietary authoring tool was the addition of a graph, describing the difference as "night and day". Their previous authoring tool, a non-node-based editor that made heavy use of multifaceted controls much alike *Quest*, was described as "extremely difficult to make the kind of conversations we wanted to make". On the other hand, their newer node-based authoring tool was favored largely due to the inclusion of a spatial hypertext graph instead of only multifaceted controls. They explained that creation and deletion of content was a lot easier, and that the inclusion of a spatial hypertext graph made it possible to "see how it's going to affect the flow of your conversation" when doing so, and that they found this is "crucial to iterating so you don't end up with bad material just because it's too hard to change it". They concluded by describing the move to a node-based authoring tool as "[unlocking] a lot of freedom" and noting that they think that "it changed the way that we write a lot and I would never want to go back".

The strong evidence in the relevant Theme and the corresponding survey question results having a high mean of μ=4.42 (unipolar 1–5) strongly suggests that graph-like interfaces as a part of the spatial hypertext paradigm are indeed helpful for making sense of the story structure and flow in a visual way.

Point 2 — Revised; Supported

Exposing attributes of nodes can aid in making sense of story structure through enhanced context, and when customizable can result in idiosyncratic rules which help to personalize how the author understands their own story structure.

As part of the *Spatial hypertext features that enhance idiosyncratic organization are valued by authors* Theme, strong evidence was presented as to how node attributes are used to enhance context, and how they are often part of application-defined or idiosyncratic rules

that further enhance the individual's understanding of the structure and flow of the story. This original point was in relation to node attributes enhancing context, specifically when exposing the internal function of a node, which PRO4 in particular provided strong evidence for. However, this Theme demonstrates that this point is a part of a larger concept of using node attributes, and therefore needs to be modified accordingly.

The corresponding survey question, phrased generically about node attributes, provides further support for this. *Extremely Important* and *Very Important* were chosen by eight and nine participants respectively, with the remaining two, PRO5 and P6, choosing *Moderately Important*. PRO5 thought that this feature was good and throughout the interview highlighted the importance of customizable attributes provided by nodes and their impact upon understanding the structure and flow of the story, as discussed in the relevant Theme. P6, on the other hand, struggled to clarify their choice but ended up describing how node labels are useful for them to grasp the broad structure, adding that detailed attributes like displaying the inner content are less helpful to them.

Given that the relevant Theme revealed that this point lies within a broader scope and that the corresponding survey question data supporting this broad scope had a high mean of μ=4.32 (unipolar 1–5), this point has been refined accordingly to take into account the more generalized findings.

Points 3, 4, 5 — Merged; Supported

Clarity of connectivity between nodes helps with visually following and making sense of the story structure and can be further enhanced by augmenting connections between lines.

The *Ease of connecting content is important* Code, made up of 11 participants including three professionals, indicates that the ease of creating the connections is important to authors. Despite this related point, there was little to no explicit participant discussion on the importance of the clarity or visual appearance of these connections. As the *Spatial hypertext graphs and additional enhancements assist authors in making sense of story structures* Theme demonstrates, authors frequently use graphs in numerous ways to visually make sense of their story structure, which inherently requires connectivity between content to be clear enough to allow for these understandings to take place. This suggests that while the presence of links between content is undoubtedly essential, their appearance and functionality may be of lesser importance to authors, or at least is an under-explored avenue of authoring tools. Point 4 of this principle suggests one such augmentation by coloring connections in the interactive authoring workflows and mentioning it in the corresponding survey question with the intention being that it would stimulate discussion around the topic either in free text boxes or in interviews. However, no participants mentioned the use of colored connections to aid in understanding the story, perhaps due to the granularity of the concept, although the use of color for identification has elsewhere proven to be invaluable as demonstrated in the *Spatial hypertext features that enhance idiosyncratic organization are valued by authors* Theme. Point 5 similarly

describes an augmentation of connections for enhanced clarity by overlaying bars on connective lines to identify them as conditional as shown in the interactive authoring workflows. Two participants, as part of the *Uses a graph to visually identify logic* Code, expressed how this clarification of a conditioned connection helped them to understand the logical flow. For example, P10, in response to the interactive authoring workflows displaying double dashed lines across conditioned connections, stated that they "made it more clear that it was a bigger decision than the rest of the decisions" that didn't involve conditional logic. P20 echoed a similar sentiment, stating that they help to identify "that there is a node there where something needs to happen before you can access that" and that "seeing that at a glance is quite useful because . . . those are the ones that you might have to edit more specifically because they are a bit more complex".

In the corresponding survey question, which asked about the general effectiveness of clear connections upon the understanding of the story structure and provided examples of both points 4 and 5, seven participants chose *Extremely Effective* and nine chose *Very Effective*, with the remaining three, PRO4, PRO5, and P6, choosing *Moderately Effective* without clarification in the interviews.

Although points 4 and 5 were separated as individual supporting points due to their uniqueness in the design which warranted separate investigation, the above evidence has shown that all these points are closely related and can be joined together as a broader point that captures all aspects. Moreover, given the supporting evidence of the importance of clear connections, including through augmentations described in points 4 and 5, and the high corresponding survey question mean of μ=4.21 (unipolar 1–5), points 3, 4, and 5 have been merged and the description altered to include the helpfulness of augmentations to connections upon clarity of structure.

Points 6, 7 — Merged; Supported

Highlighting and animating connections between nodes helps to better visualize, follow, and make sense of the structure of a story, particularly in graphs that have a low legibility such as disorderly graphs or those at a physically reduced scale due to zooming.

In the interviews, five participants, including two professionals, expressed that including animated lines between nodes helped them to understand connectivity, as shown by the *Thinks that animated lines between nodes help to understand connectivity* Code. PRO4 didn't find highlighting outputs from a single node that helpful in their workflow, but noted that when zoomed out, seeing animated pathways between two distant nodes was considered as "being essential". PRO2 echoed a similar sentiment "especially when you get a much bigger a game or a much bigger narrative and you've got many more nodes and many more links, that would be very helpful". The *Desire for animated lines to also highlight logical pathways* Code, although only populated by P19, demonstrates potential extensions to this form as the participant suggests handling logical pathways within the animated context rather than just physical traversals. PRO5, the developer of a prominent

authoring tool that features animated lines, explained in the *Users complained when animated lines were removed* Code that when attempting to remove the animated lines from their tool multiple times, authors had complained, which highlights the importance of this augmentation to connective lines does not have an impact upon the story itself but can aid structural clarity.

In the corresponding survey question, seven participants chose *Extremely Effective*, eight chose *Very Effective*, three chose *Moderately Effective*, and one chose *Not at all Effective*. The *Not at all Effective* came from PRO2, as they reported the feature to not work in the prototype. During the interview, they were shown the backup video included in the guides, after which they appraised the feature as "really, really helpful". With a high mean of μ=4.05 (unipolar 1–5) and accounting for this appraisal, the consensus from the data is that the feature is at least *Very Effective*.

Points 6 and 7 were separated as they presented a different use-case within the prototype design, but in actuality they both share an overarching concept of visualizing connections through highlights and animations to better make sense of structure. While there is favorable evidence from the interview and survey for the general concepts of highlighting and animating node connections, there are fewer points for the specific case of paths between nodes, which in a sense builds on top of the first point. By merging the points together, the concept is further generalized, accepting that animated connections are beneficial, but leaves the particularities of how these animated lines can be used and augmented for future research, with path-based animations being a potential first step. Moreover, as a use-case was identified for when graphs are zoomed out rather than just disorderly, the description has been modified to account for scenarios of low legibility, including both disorder and physical scaling.

Point 8 — Supported

Support for traversal methods is essential to being able to make sense of the story structure, especially with large projects where visual space is limited and elements are frequently off the screen.

In the interviews, ten participants touched upon navigation in some capacity, always in response to a prompt from the researcher rather than on their own accord, and when doing so, discussion was short-lived. This suggests that navigation is either not that important, or conversely it is integral to spatial hypertext and is simply expected as a feature without much thought given to it. Of the ten participants, three expressed that they found navigation more practical when dealing with larger stories, as shown in the *Thinks navigation is more important for large projects* Code. Of those three, two particularly highlighted the helpfulness of panning and zooming within the graph space. P16 instead claimed that navigation is helpful to them regardless of project size as they organize their nodes into distinct and distant groups which require navigating between as they work with the structure. Although not explicitly discussing navigation, both

PRO3 and P18 similarly stated how they frequently zoom in and out to alter the visual granularity of the structure, which when laid out appropriately, can be understood at varying degrees of detail. P6, P9, and P17, on the other hand, claimed that they typically focus on very localized areas of the graph without the need to move away, therefore relying less frequently upon navigation tools. However, P19, who also reports focusing on localized areas of the graph, instead explained that they often navigate around even in this situation to remind them of the context within which the localized editing is taking place. This varied evidence demonstrates that navigation is of importance to authors despite few instances occurring in the interviews, but also highlights the need for future work beyond the scope of this study to understand the different individual use-cases of navigation and how they impact the authoring process.

In the corresponding survey question, ten participants chose *Extremely Important*, six chose *Very Important*, and three chose *Moderately Important*, with a resulting mean of μ=4.37 (unipolar 1–5). This result, combined with the limited yet varied evidence from the interviews, suggests that traversal methods are indeed an important aspect to aid understanding of structure, although the exact details of how remain obscured. Given the high mean, this signals that there is future work beyond the scope of this study to be done to understand the nuances of exactly what forms of traversal have impacts and how they are used.

6.6.2 Fast Track Testing

Letting users jump to any state of the story enables more rapid and focused testing sessions.

Point 1 — Supported

Being able to begin testing from any node means that unimportant content for a particular testing session can be skipped over, reducing the need to test from the beginning each time, which means that testing sessions can be rapid and more focused on a particular point as prior content is not traversed.

As shown in the *Where authors begin testing from can affect the scale and purpose of testing* Theme, the interview data strongly suggested that testing from arbitrary locations within the story is frequently done by a majority of authors typically to test a short section of the narrative without having to traverse from the beginning of the story.

In the corresponding survey question, 14 participants chose *Extremely Important*, two chose *Very Important*, PRO5 and P12 chose *Moderately Important*, and PRO4 chose *Slightly Important*. P12 stated that they usually test from the beginning as "it would give me a refresher on everything else that I'd done up to that point", but they were working with shorter stories. They later clarified that they think it is "fairly important to have that function" especially when "you are creating an absolutely huge project". PRO4 explained that in their personal use-case, which is the editing of dialogue structures for games,

they don't often have the need to test from arbitrary points and prefer to test from the beginning due to the manageable size of the structures, but later clarified that for bigger stories, the ability to do so is "super important".

Between the evidence in the relevant Theme and the corresponding survey question having a high mean of μ=4.53 (unipolar 1–5) even without adjusting for clarifications, there is a strong case for the criticality of being able to test from arbitrary points of the story, and that being able to do so results in quicker and more focused testing as prior content need not be traversed.

Points 2, 3 — Merged; Supported

Allowing users to control the story state during testing allows the author to explore specific state configurations and their impact upon story flow.

In the interviews, the *Desire to tweak variables while testing* Code, made up of ten participants including two professionals, shows that a range of participants would like the ability to tweak variables while testing. PRO3, for example, described that being able to view variables while testing "gives you visibility into what's going on", and that being able to modify them is "really the only way you can test [their impact]", noting that as they traverse a story it is easy to mentally lose track of the states as they change, meaning the values and their impacts are often unknown. They followed by explaining that their existing authoring tool of choice, *Twine*, allows for tracking variables "at a very rudimentary level" but that the ability to manipulate them is missing which they "would personally find very helpful". PRO5 provided a practical use of this feature in that once they have, at a given variable branch, "verified that the behavior is what you expect", that they then revert the state to what it was before and then "change [variables] so that you will follow the other path and make sure that you do in that case follow the other path", particularly in case they forget to check later, which would result in "shipping with all sorts of things that are not doing what you meant [them] to do". They also expressed interest in the ability to store and restore saved states, explaining that they have done this when authoring their own narrative works in other authoring tools. They later clarified that although the states you may be testing may not equate to the potential states the authors can achieve at the equivalent point through reading, it's a better solution than "having to walk through the whole hypertext, especially if it's big".

While only three participants are involved in the *Moves variables if they appear prior to desired testing location* Code, it does demonstrate a practical problem that some authors encounter that point 2, as implemented in the interactive authoring workflows, can resolve. In this case, participants began testing and met dynamic conditions that relied upon a variable that was initialized prior to the starting point, meaning it was unset or otherwise set with an unexpected value. Their solution was to physically move the variable's initialization to be situated just before the dynamic condition being tested, which introduces the risk of human error both in breaking other dependent logic and

forgetting that the alteration was made, a problem that can be resolved by simply allowing variables to be modified. Point 2 allows for states to be modified as the story is being tested, which mitigates unset variables being a problem, as they can be manipulated to any fitting value even if they have not yet been initialized. P9 similarly described that they had conditional dialogue throughout their story reliant upon a choice made at the start, and expressed interest in being able to quickly load and switch between predefined save states to test the impact of the various configurations the player could have created with their initial choices.

In the corresponding survey question, nine chose *Extremely Effective*, six chose *Very Effective*, PRO4, P7, and P12 chose *Moderately Effective*, and P6 chose *Slightly Effective*. In discussion with P6, it was concluded that they had misread the question to be about the ability to add variables into a story, and that they didn't find that practical as they think the inclusion of variable-based interactivity is secondary to a crafted linear narrative. In the case of P12, they explained that most of their experience is with an authoring tool that does not support variables, and that they were uncomfortable in confidently answering based on their limited experience with the interactive authoring workflows alone, although this was an isolated case. PRO4, on the other hand, expressed that in their use-case of authoring tools, which is for in-game dialogue, that they instead use save games to store and retrieve configurations of the entire game state, noting that while the methods are different, they achieve the same thing, as they "save and load all the time while testing the game for exactly that reason, to go through different pathways". They similarly described the use of an interactive in-game debugging tool that allows the author to, by answering a set of predetermined questions, build up and apply a custom game state so that they can test that specific combination, such as "test[ing] act 2 having failed act 1 . . . test[ing] act 2 to having succeeded act 1 . . . test[ing] act 2 with [item X]". This is in line with much of PRO4's explicit testing done within a game engine as covered in the *Built-in testing isn't necessarily representative of the intended story experience* Theme.

Points 2 and 3 were initially separated as although they were both related to the concept of tweaking variables while testing, they presented different scenarios under which tweaking took place, the former being altering individual variables and the latter allowing for predefined configurations to be saved and loaded. While there is favorable evidence for the tweaking of individual variables while testing, there are fewer data points for the saving and loading of states, with the most prominent example being from PRO4 in relation to loading game states and using in-game interactive debuggers. This suggests that while the ability to tweak variables while testing is sought after, the saving and loading of configurations is more niche. It can also be argued that saving and loading states of variables are a specialization of tweaking variables during testing. Therefore, with the evidence chiefly supporting the arbitrary tweaking of variables during testing, the high corresponding survey question mean of μ=4.21 (unipolar 1–5), and the niche

and related nature of managing states, the points have been merged chiefly focusing upon the former.

Point 4 — Supported

Supporting recording and playback of traversals means that users can repeat focused testing sessions quickly with a reduction of human error in comparison to manually repeating the same traversals.

In the interviews, several participants discussed the Recordings feature present in the interactive authoring workflows as indicated by the eight related Codes that include Recordings, but this was only in response to survey answer clarifications rather than raising the feature on their own accord, which initially suggests that it may be of less importance than neighboring features to aid testing. The corresponding survey question, however, suggests that the concept of the feature is helpful to participants if present. Eight participants chose *Extremely Effective*, six chose *Very Effective*, three chose *Moderately Effective*, and two chose *Slightly Effective*, giving an overall mean of μ=4.05 (unipolar 1–5). PRO3 was particularly fond of the feature, providing a use case for recovering previous readings for the sake of digital preservation or simply to recall how that reading had evolved over time, describing it as "really a quite powerful" feature. They also noted how the Recordings feature being able to identify if changes made since an existing recording was captured would break the recording was a "really powerful, powerful tool" and that they "can't think of another system that has something like that". PRO4 also complemented the feature as being important, comparing it to their studio's increasing use of software like *OBS Studio* to record play sessions for both debugging and to recall particular choice sequences, much alike the Recordings feature. PRO5, on the other hand, thought that the feature was only *Slightly Effective* as they considered how it doesn't "[give] you a whole lot of insight into what the manifold of trajectories through the hypertext [feel] like". However, they then explained how they would find it helpful in debugging alterations to a narrative that may break certain pathways, as PRO3 had, and said that they felt their initial survey response was "uncharitable" upon reflection. P7, who also chose *Slightly Effective*, explained how they felt that the Recordings feature was comparable to playing through manually each time, but after discussion in the interview concluded that the feature provides benefits beyond what they had initially assumed.

The corresponding survey question data and the clarifications within the interviews suggest that participants appreciate the presence of this feature but are sometimes unable to grasp its exact purpose, which could be due to it being relatively unique and uncommon within existing authoring tools. For the purpose of this study, the original description of this point holds true given the presented evidence, although the fact that this feature is received well yet is sometimes challenging to grasp suggests that it would benefit from future research beyond the scope of this study to further refine its testing aid.

6.6.3 Structure

Interfaces that use a visual metaphor to represent story structure and connectedness enable idiosyncratic organization and management of an author's own story structure.

Point 1 — Supported

Spatial hypertext graph design helps with personal organization due to its intuitiveness and lack of restrictions in regard to positioning of content.

One of the key takeaways from the *Spatial hypertext features that enhance idiosyncratic organization are valued by authors* Theme was that authors frequently use the physical space of graphs and the spatial relationships between nodes on the graphs to organize their nodes into various idiosyncratic patterns. This provides strong direct evidence in support of this point.

In the corresponding survey question, six participants chose *Extremely Effective*, ten chose *Very Effective*, and three chose *Moderately Effective*. Two of the three participants that made up *Moderately Effective*, despite their choice in the survey, described in the interviews actively using graphs and personalized node layouts to organize their content into groups. Despite plentiful discussion around the concept of organization of content, no participants otherwise explicitly clarified their choice within the survey.

The strong evidence in the relevant Theme and the corresponding survey question having a mean of μ=4.16 (unipolar 1–5) strongly suggests that graph-like interfaces as a part of the spatial hypertext paradigm are helpful for personal organization of content.

Point 2 — Supported

Supporting arbitrary visual grouping tools that do not impact the structure aids personal organization.

As discussed during the *Spatial hypertext features that enhance idiosyncratic organization are valued by authors* Theme, nine participants mentioned the use of either Frames or similar features for explicit grouping of content within a spatial hypertext graph, and one professional whose authoring tool of choice does not include the feature described how the feature is requested by multiple authors on their team. This provides strong direct evidence in support of this point.

In the corresponding survey question, nine participants chose *Extremely Effective*, nine chose *Very Effective*, and PRO3 chose *Moderately Effective*. PRO3 clarified their choice by stating that while they "understand theoretically why that's useful" that it's simply "not critical for me . . . purely [based on] my particular background" and "the way I particularly write". Within the same Theme mentioned above, PRO3 described how they consider the visual layout of the nodes in a spatial hypertext as an intrinsic element of the narrative itself. Therefore, their choice for this survey question reflects their personal use

case as a hypertext fiction author who considers visual structure a part of the narrative rather than as a means to personally organize the literal content of the narrative. The direct evidence in the relevant Theme and the corresponding survey question having a mean of μ=4.42 (unipolar 1–5) strongly suggests that dedicated grouping tools like Frames help authors to organize their content into idiosyncratic structures despite having no impact upon the story itself, with the exception of authors like PRO3 that consider the visual layout a component of the narrative experience itself.

Point 3 — Partially Supported

Support for traversal methods directly complements the ability to personally organize spatial hypertext graph contents.

In the interviews, ten participants touched upon navigation in some capacity, always in response to a prompt from the researcher rather than on their own accord, and when doing so, discussion was short-lived. This suggests that navigation is either not that important, or conversely it is integral to spatial hypertext and is simply expected as a feature without much thought given to it. Of all ten participants, none explicitly discussed the benefits of navigation upon the organization of their story contents. However, we can infer from the *Spatial hypertext features that enhance idiosyncratic organization are valued by authors* Theme that authors organizing and laying out nodes to their liking inherently requires a certain level of navigation, especially when dealing with larger structures where the content may not all fit into view. For example, seven authors in the *Organizes nodes in an arborescent structure* Code described laying out their nodes in a tree-like structure either vertically or horizontally, which can quickly overflow the available screen space, meaning that authors must rely upon graph navigation to not only observe their layouts but to append or modify them. Similarly, nine participants in the *Organizes nodes spatially into pseudo-groups* Code described grouping their nodes together into distinct chunks, which again will eventually overflow the available screen space, particularly if the groups are distant, which requires navigation to effectively work with. Although it is difficult to ascertain from the data gathered in this study, it is possible that without effective navigation, personal organization using methods like the above could be negatively impacted.

In the corresponding survey question, nine participants chose *Extremely Important*, seven chose *Very Important*, P6 and P9 chose *Moderately Important*, and P15 chose *Slightly Important*. P6 and P9 did not provide explicit reasoning as to their choices, but in their respective interviews, both expressed that they typically work on small-scale stories and that consequently navigation was considered of lesser importance to them in that context. While this is not definite, it could mean that for some users, working on small scales diminishes the importance of navigation. P15 also provided no explicit reasoning and only has experience with an authoring tool that does not include a spatial hypertext graph, but nothing certain was able to be inferred from their data. The high mean of

μ=4.26 (unipolar 1–5) alongside the mostly inferred evidence suggests that navigation does play an important role in regard to personal organization of story content, but the nuances of exactly how is a subject for future work beyond the scope of this study.

Point 4 — Partially Supported

Being able to identify types and unique instances of nodes helps with personal organization as it lowers cognitive friction and helps users focus on organization rather than identification.

Although in the *Spatial hypertext features that enhance idiosyncratic organization are valued by authors* Theme there is significant evidence for idiosyncratic organization, there is no direct evidence within the interviews that the ability to identify nodes beforehand assisted in this process. However, we can infer that creating an organized structure, whether spatially or through visual attributes, does inherently require a certain degree of identification in the first place, but it remains unclear to what extent this matters.

In the corresponding survey question, nine participants chose *Extremely Important*, nine chose *Very Important*, and one chose *Moderately Important*. No participants clarified their reasoning for this choice in the interviews. Given the high mean of μ=4.42 (unipolar 1–5) alongside the supporting evidence being only inferred rather than direct, this suggests that identification of type and instance of nodes is of importance when organizing into structures, but the exact means of how this helps remains unclear and is subject for future work beyond the scope of this study.

6.6.4 Experimentation

Interfaces that use a visual metaphor to represent story structure and connectedness enable easier experimentation of structure and connectedness.

(In this context, experimenting refers to authors making changes by exploring different structures and configurations. Examples of this include changing the number of outputs a node has, changing where an existing connection goes, and adding or removing nodes.)

Point 1 — Supported

Spatial hypertext graph design provides greater context within which experimentation can take place.

In the interviews, PRO3, PRO4, and PRO5 all stressed the importance of spatial hypertext graph solutions upon their authoring experience. PRO3, for example, described that one of the few downsides of authoring narratives in *HyperCard*, a generic hypertext authoring tool without a spatial graph, was that "you could never get out of the card and step out to a view that gave you a structural representation", adding that it's "also the problem with writing stuff in HTML". They went on to express that the spatial nature of

Twine, their recent authoring tool of choice, is "a really powerful tool because it gives you the ability to put stuff there, identify a link's place, source, and target", which when contrasted to their thoughts on *HyperCard* and HTML demonstrates the importance of enhanced context brought about by spatial hypertext graphs. PRO4 similarly described a transition between two proprietary authoring tools at a game studio that they use as "night and day". Their previous authoring tool, a non-node-based editor that made heavy use of multifaceted controls much alike *Quest*, was described as "extremely difficult to make the kind of conversations we wanted to make". On the other hand, their newer node-based authoring tool was favored largely due to the inclusion of a spatial hypertext graph instead of only multifaceted controls. They described that creation and deletion of content is a lot easier, and that the inclusion of a spatial hypertext graph makes it possible to "see how it's going to affect the flow of your conversation" when doing so, and that they find this is "crucial to iterating so you don't end up with bad material just because it's too hard to change it". They concluded by describing the move to a node-based authoring tool as "[unlocking] a lot of freedom" and noting that they think that "it changed the way that we write a lot and I would never want to go back". PRO5, themselves the creator of a prominent spatial hypertext authoring tool, expressed that they "really need spatial hypertext" as an aspect of an authoring tool. They described that when writing, "usually the story is evolving as I'm evolving links [between nodes]", and that they often use customizable aspects of spatial hypertext to provide additional context when revisiting parts of the story to iterate upon the content. For example, in their latest novel, there is a missing student represented as a node with an orange border applied, which was done to serve as a contextual reminder about the purpose of the node when returning to this particular character to iterate on their place within the story. In the same novel, they described representing two inseparable students as two nodes placed in close spatial proximity, which is similarly used as a reminder as "I'll have foreshadowing now about their inseparability" when iterating upon them.

The premise of the *Spatial hypertext features that enhance idiosyncratic organization are valued by authors* Theme is that aspects of spatial hypertext graphs allow for idiosyncratic organization of contents within the graph. This chiefly comes in the form of using node attributes or spatially laying out nodes to create pseudo-groups. From this, although indirectly, we can infer that users creating their own rules and structures of their own works inherently increases context when editing in a way that makes sense to them, further demonstrating the benefits of spatial hypertext graphs on the editing process. As an example, PRO4 described using node attributes such as titles, descriptions, and colors, explaining that when they are not exposed or insufficiently represent the node content, "the more mental bandwidth I'm putting in to try to understand and remember what my not exposed information is", which demonstrates the context that spatial hypertext features provide when editing and revisiting content to make creative alterations.

Additionally, the six participants that made up the *Desire to see a graph while editing* Code, all of which used an authoring tool without a graph that allowed editing, expressed the desire to have a graph present when editing, mostly due to the enhanced context (or lack thereof without a graph). For example, P19 described that when using a text-based authoring tool, "if you have three of four choices and then you keep writing underneath for something else, it sometimes gets a bit difficult to see", and consequently that they would prefer a visual node-based representation, and that they drafted their narrative in a flowchart program prior to implementing it in the text-based authoring tool due to this lack of context. P13 similarly described the desire for a graph because "if I'm writing in bits all over the place within the story . . . I can see what I've [already] got and where it is".

In the corresponding survey question, 13 participants chose *Extremely Important*, five chose *Very Important*, and PRO5 chose *Moderately Important*. PRO5 did not clarify why but did express favor toward spatial hypertext graphs and their contextual benefits as described above. This survey question targeted both the importance of clarity within a spatial hypertext graph and the ease of identifying and understanding connections between content when experimenting. The evidence presented above along with the high mean of μ=4.63 (unipolar 1–5) strongly suggests that spatial hypertext graphs do provide beneficial context within which iterative editing and exploration can take place.

Point 2 — Supported

Being able to identify types and unique instances of nodes helps with experimentation as it prevents focus being spent on identification which could slow or hinder the experimentation process.

A large part of the *Spatial hypertext features that enhance idiosyncratic organization are valued by authors* Theme is that authors utilize the attributes of nodes within spatial hypertext graphs to create idiosyncratic rules for organization. These attributes are in turn used to identify the nodes in some way personal to the author, which aids the revisiting and editing of content as their focus can be on the actual intended manipulation and not the identification of the content being edited. For example, PRO4 described using node attributes such as titles, descriptions, and colors, explaining that when they are not exposed or insufficiently represent the node content, "the more mental bandwidth I'm putting in to try to understand and remember what my not exposed information is", which demonstrates how authors may rely on attributes to quickly identify nodes when editing and revisiting content to make creative alterations. They also described a particular application of using color to quickly identify content in that "if something is an NPC speaking, that's going to be a certain color, if it's a potential player response, that's going to be a different color", and that they use this coloring of nodes to "get a sense of the flow of a dialogue just by looking at the colors" at a high level, which mitigates the need to focus on identification when writing and revising content.

In the corresponding survey question, ten participants chose *Extremely Important*, six chose *Very Important*, and three chose *Moderately Important*. No participants clarified their reasoning for this choice. Given the high mean of μ=4.37 (unipolar 1–5) alongside the supporting evidence being practical yet limited in scope, this suggests that identification of nodes when experimenting is important to authors. While beyond the scope of this study, it remains for future investigation to identify the exact details of how identification aids can be enhanced to better support authors in this regard.

Point 3 — Partially Supported

Support for traversal methods directly complements the ease of experimentation.

In the interviews, ten participants touched upon navigation in some capacity, always in response to a prompt from the researcher rather than on their own accord, and when doing so, discussion was short-lived. This suggests that navigation is either not that important, or conversely it is integral to spatial hypertext and is simple expected as a feature without much thought given to it. Of the ten participants, three expressed that they found navigation more practical when dealing with larger stories, as shown in the *Thinks navigation is more important for large projects* Code. Although few in number, this does suggest that some authors find navigation at the macro scale more helpful than at the micro scale where fine-grained editing and experimentation takes place. On the other hand, P16 instead claimed that navigation is helpful to them regardless of project size, as they organize their nodes into distinct and distant groups which require navigating between, and within each group they make liberal use of whitespace, therefore requiring navigation even when editing and experimenting with localized areas. This is not always the case though, as P6, P9, and P17 each described how they don't find navigation as important as they tend to work in localized areas of the graph without the need to move around much, although it is possible that they did not consider navigation between areas in their response. Another related behavior was described by P19, who reported focusing on localized areas of the graph, but periodically navigates around to other regions to remind them of the context within which they are making changes. PRO2 instead expressed the desire to see the complete structure, and when becoming too large to legibly fit on a single screen, allow for "zooming in close enough so that you can play around with individual nodes" as well as panning around once within a local region.

In the corresponding survey question, 12 participants chose *Extremely Important*, three chose *Very Important*, and four chose *Moderately Important*, with a resulting mean of μ=4.42 (unipolar 1–5). The high mean combined with the limited yet varied evidence from the interviews suggests that inclusion of traversal methods do complement editing and experimenting for authors, but the exact degree to which they do and the way in which they are utilized appears to differ between users. Due to the limited amount of data gathered, these differing behaviors cannot be grouped with accuracy, but this granularity of understanding lies beyond the scope of this study and is for future work to investigate.

Point 4 — Partially Supported

Being able to identify the connected state of links between content helps with experimentation as it prevents focus being diverted to understanding the connections which could slow or otherwise hinder the experimenting process.

As the *Spatial hypertext graphs and additional enhancements assist authors in making sense of story structures* Theme demonstrates, authors use graphs in numerous ways to visually make sense of their story structure, which inherently requires understandable connectivity between content. The *Thinks that animated lines between nodes help to understand connectivity* Code, made up of five participants including two professionals, similarly shows how some authors value enhanced lines to aid with better understanding connectivity between elements. In the *Being able to visualize links between content is important* Code, PRO3 described linking between content, "the ability to go from span to span", as "fundamental" and that it is "why I choose to write something as an interactive, as opposed to just a linear text", following up that second to creating content, being able to "create a link to another place . . . in a way that I can visualize" is critical. They provided an example in that authoring with either *HyperCard* or HTML does not afford visual representations of links in a way that can be easily understood, and conversely that *Twine* enables this and "is a really powerful tool because it gives you the ability to put stuff there, identify a link's place, source, and target". The clear importance of visualizing connections and the variety of ways they can be used to understand the story strongly suggests that the connections themselves being clear and understandable is important to effectively support creative experimentation. No other participants explicitly discussed the benefits of understanding connections upon the editing and experimenting process, instead chiefly focusing upon aiding understanding of the story structure. That said, we can infer with reason that authors do benefit from understanding connections when editing and experimenting as outlined above and in related Themes.

In the corresponding survey question, 13 participants chose *Extremely Important*, five chose *Very Important*, and one chose *Moderately Important*. This survey question targeted both the importance of clarity within a spatial hypertext graph and the helpfulness of being able to easily identify and understand connections between content upon experimenting. The high survey mean of μ=4.63 (unipolar 1–5) combined with the limited evidence from the interviews suggests a relationship between the ease of identifying and understanding connectivity of content and its corresponding impact upon the editing and experimenting process. However, due to the evidence from the interviews being limited in scope and quantity, the exact nature of this relationship is unclear and warrants future investigation that is beyond the scope of this study.

Point 5 — Supported

Being able to easily create and manage connections between nodes with minimal cognitive friction ensures that the user can focus on the actual restructuring and reorganizing of connectivity between nodes rather than focusing on the details of managing connections.

The *Ease of connecting content is important* Code, which is made up of 11 participants including three professionals, demonstrates the importance of being able to easily create and manage connections, a process described by PRO5 as "fluent linking". PRO2 stated that when editing a story, "the ability to create links as easily as possible … from anywhere to anywhere very quickly" is a "key function" of the editing experience, and highlighted the fundamental relationship between interactive fiction and linking of content. Similarly, PRO3 described linking between content, "the ability to go from span to span", as "fundamental" and that it is "why I choose to write something as an interactive, as opposed to just a linear text", following up that second to creating content, being able to "create a link to another place … in a way that I can visualize" is critical. P17 described how "the ease of being able to write a flowing story" in terms of being able to "make separate branches and maybe combine them … without being limited in my options" is important to them, noting that, in this context, "if I'm constantly having to do ten steps to do one thing in a tool, I'd be less likely to continue using it". P19 likewise reflected upon the ease of manipulating connections, describing that when editing and experimenting, "it's so important to be able to quickly alter those links … because it kind of defeats the point of experimenting if you can't do it quickly and easily". P18 also praised the "quickness of access between making connections", providing an example where "if something didn't really work for you … you could just cut the connection … there wasn't anything you had to go through and read and [consider if] the player can access … it was just [cutting the connection and now the player] cannot go here anymore because that connection doesn't exist", further highlighting the benefits of efficient connection management when experimenting with content connectivity.

In the corresponding survey question, 11 participants chose *Extremely Important*, seven chose *Very Important*, and one chose *Moderately Important* although they did not clarify why. The high mean of μ=4.53 (unipolar 1–5) and the evidence outlined above suggests that the ease of creating and managing connections has a direct impact upon an author's ability to experiment with their story without hindrance. However, the nuances of how creation and management of connections can be made less intrusive and more intuitive remains a topic for future usability research within this field beyond the scope of this study.

6.6.5 Branching

Interfaces that use a visual metaphor to represent story structure and connectedness enable easier creation and management of branches.

Point 1 — Revised; Supported

Authors tend to organize nodes to visually mimic story flow which makes it easier to work with branches due to increased local context.

A significant component of the *Spatial hypertext features that enhance idiosyncratic organization are valued by authors* Theme demonstrated that authors organize their nodes into personalized structures and that these structures often visually mimic the underlying story flow. For example, seven authors in the *Organizes nodes in an arborescent structure* Code described their personal layout as tree-like, mimicking the flow of the story visually, typically as a sequential linking of events. In the case of P14, they explicitly described organizing their nodes to visually mimic hypertext patterns such as Split-Join and Mirror World. We already know from the *Spatial hypertext graphs and additional enhancements assist authors in making sense of story structures* Theme that authors will use aspects of spatial hypertext graphs to visually understand their story structures. P7, P9, P14, and P20 each specifically described visually mimicking the story flow as beneficial to understanding branches in context of their surroundings, with P14 adding that "when you revisit [your story] after not looking at it for weeks…it's very, very useful". However, in relation to creating branches, PRO4 explained that "exposing as much context as possible is important so that as you're reading it, you understand not only the path that you can go on, but the paths that you're not taking", which suggests that node layouts mimicking story flow is but one aspect of context that aids in this process, and may not be sufficient for all authors in isolation. The authoring tool that PRO4 uses in a professional setting colors nodes based on their type and also visually exposes a significant chunk of inner dialogue text that each node represents, further enhancing contextual understanding of branches beyond just the spatial layout of nodes.

In the corresponding survey question, seven participants chose *Extremely Useful*, nine chose *Very Useful*, and three chose *Moderately Useful*, resulting in a mean of μ=4.21 (unipolar 1–5). The evidence for this point chiefly demonstrates the benefits of retrospectively understanding the context of a branch through visual mimicking of story structure, with limited evidence for the benefits of contextual enhancement for branch creation. Additionally, the evidence suggests that visually mimicking story flow is but one aspect within a greater set of contextual hints, within which its prominence in both isolation and in contrast to the other aspects remains unknown. However, it is clear that authors do mimic story flow when organizing their content and that it does aid working with branching structures.

Point 2 — Partially Supported

Reducing the complexity of creating and manipulating connections enables easier creation and experimentation with branches.

The fifth point of the *Principle of Experimentation* suggested that the ease of creating and managing connections has an impact upon an author's ability to iterate and revise content when experimenting. Despite this, there was little explicit discussion of its impact upon the ease of creation and experimentation with branch structures. P18 briefly touched upon the subject, stating that in *Twine*, "the quickness of access [for] making connections" was something that they were fond of when working with branching structures, noting their preference over *Inform*, a text-based authoring tool, as "if something didn't really work for you . . . you could just cut the connection . . . there wasn't anything you had to go through and read and [consider if] the player can access . . . it was just [cutting the connection and now the player] cannot go here anymore because that connection doesn't exist". As also described in the fifth point of the *Principle of Experimentation*, the *Ease of connecting content is important* Code demonstrates the importance of being able to create connections between content easily, but only P17 discussed this in the context of branch creation. This absence of discussion of the ease of connecting content in the context of branches does not indicate that it is untrue, but rather that participants may have not differentiated between creating connections between two linearly related nodes and those that result or take part in divergence of the story structure.

In the corresponding survey question, 12 participants chose *Extremely Important*, six chose *Very Important*, and one chose *Slightly Important*, although this participant did not have experience with graph-based authoring tools and provided no further reason for their choice. Given the high mean of μ=4.53 (unipolar 1–5) and the limited supporting evidence, this indicates that the ease of creating and managing connections is of some importance to working with branches, but the nature of exactly how still remains unclear. Therefore, while the data in this study suggests such a relationship is present, future research in the field could aim to better understand the nuances between the implementation of connection management and branch creation in spatial hypertext graphs.

6.6.6 Contented Authoring

Users that become content with a simple method of authoring are unlikely to explore more advanced features, even if it would enhance their work.

Note that due to the lack of conclusive supporting evidence for the below contained points, this principle will be removed from the final collection.

Point 1 — Unsupported

Including nodes with generalized functionality can reduce the chance of authors becoming locked into repeatedly reusing the same kinds of nodes.

In the corresponding survey question, nine participants chose *Strongly Agree* and ten chose *Agree*, resulting in a positive sum of Σ=28 (bipolar ±2), although no participants

clarified their choice. This choice was made exclusively in context of the interactive authoring workflows, suggesting that in an authoring tool that provides a flexible set of nodes as opposed to rigidly constrained fixed-function node types, authors feel comfortable building structures in a variety of ways, although the degree to which this factor influenced their choice remains unclear. The interviews did not provide any conclusive evidence to confirm this. This point does not necessarily translate equally to authoring tools with predefined constraints where authors are expected to rely upon a limited set of nodes. For example, PRO4 uses an authoring tool in a professional capacity for the creation of dialogue for a specific game engine, and consequently the node types available for authors to work with are limited based on what the game engine's dialogue system expects and understands. In the authoring tool, these nodes are colored based on their predetermined type, which as PRO4 explained can help to "get a sense of the flow of a dialogue just by looking at the colors of it". Generalizing node types would inherently remove this functionality and complicate interaction with the game engine. Therefore, we can conclude that the success of this point is also dependent upon the constraints by which the authoring tool is designed and that consequently generalizing nodes could aid or hinder the authoring experience based on the prerequisites that the authoring tool has.

Despite the survey resulting in a high sum, this point remains inconclusive due to a lack of supporting evidence gathered from this study. However, this does not invalidate the point, and future work outside of this study could aim to identify the degree to which generalized nodes alter the variance with which authors can build different structures, and to further investigate circumstances at which this point becomes detrimental rather than assistive such as in the case described above.

Point 2 — Partially Supported

Providing multiple ways for users to complete tasks can help to break a repetitive workflow.

The corresponding survey question results for this point varied greatly, resulting in a sum of Σ=3 (bipolar ±2). Two participants chose *Strongly Agree*, with only P13 clarifying that "having different ways of being able to do things means you're much more likely to stay focused on it and progress [a lot] quicker", and that it helped with their ADHD as "changing it up helps keep you going", although this is an isolated case and may not necessarily represent the wider population. Five participants chose *Agree*, although none clarified their reason. Seven participants chose *Neutral*, with only P8 clarifying that they find this point dependent upon the scale of authored stories, describing how at a larger scale this effect may become more prominent. Four participants chose *Disagree*, with two clarifying their choice. P19 explained that "once I am comfortable and happy with a way of doing something, I'll keep using it unless it's made clear to me that it's somehow vastly inferior to an alternative", and P20 described how they typically "choose one method as my preferred method and primarily stick with that", adding that they may periodically try other solutions but chiefly rely on their first intuitive method. PRO4 chose *Strongly*

Disagree and explained that in their opinion "the simpler and clearer a tool is, the less psychological investment the creator has in 'managing the tool' and the more they have in telling their story".

Although the sum of the survey is net positive, there is great variety not only in the results, but also in the clarifications for why variation in methods does or does not help authors to avoid repetitive workflows. This suggests that the extent to which including a variety of methods to achieve the same task impacts being caught in repetitive workflows is an individual trait, although the degree to which this is true remains inconclusive from the data gathered in this study. Future work beyond the scope of this study could aim to identify the relationship between individual authors and the use of varying methods of achieving the same task on repetitive workflows.

6.7 Summary of Themes & Refined Principles

This section collates all of the Themes from §6.5 and the refined set of principles and their points from §6.6 into a listing that summarizes their contents, clearly demonstrating the contribution of the study in a more compact and referenceable format.

6.7.1 Themes

Support for external testing is an underexplored factor

Some people send segments of incomplete story to others for testing, but this feature is rarely acknowledged explicitly, which means people must rely on built-in exports which typically package the game as if it were final, which can be a hindrance. By acknowledging that exports are sometimes used for external testing, we can enhance this with better support for debugging, which is currently underexplored.

Built-in testing isn't necessarily representative of the intended story experience

Sometimes authoring tools serve as an intermediary where the story will be experienced in an external framework like a game engine. The built-in testing harness provided by an authoring tool may not reflect this, which results in testing, especially for creative evaluation, being done out of context, which can be detrimental to the author's evaluation of the quality of the story that they are crafting.

Where authors begin testing from can affect the scale and purpose of testing

The position from which authors start testing (the beginning or at an arbitrary point) reflects the typical scale and purpose of the testing. Regardless of the difference, it is common for authors to use both methods, which means we should not limit where testing can occur from and should strive to aid difficulties that come with starting at arbitrary points.

Authors will find ways of testing their stories that can be unintended or unexpected by developers

Authors will often find ways beyond the built-in testing features supported by developers to test their stories. By understanding and embracing the additional ways that authors seek to validate and appraise their stories, we can further enhance the unexpected or unconventional features used by authors for testing to provide them a greater variety of options for testing their stories.

Recognize the need for distinct multi-stage planning and building workflows

For some authors, there is a clear distinction between the planning and building of a narrative, and that planning is often done external to authoring tools due to a lack of support for such ways of working, whether intentional or not. Recognizing this separation can help us to build more informed tools that better support authors by integrating in part or in whole solutions to the planning as well as building aspects of the creation of interactive narratives.

Spatial hypertext features that enhance idiosyncratic organization are valued by authors

When working with spatial hypertext systems, features such as visual attributes and the freedom to position nodes enable idiosyncratic rules to emerge. This kind of functionality is valued and widely used by authors. Recognizing the ways that authors create their own rules will allow us to better our own tool designs, in particular noting that authoring tools are not only a means to create a story, but are in fact a creative space within which authors explore and craft their stories as they are built. Features that may appear unrelated to narrative development, like those that allow for the emergence of idiosyncratic rules and do not impact the story, may actually be more critical than they seem to the authoring experience.

Spatial hypertext graphs and additional enhancements assist authors in making sense of story structures

Spatial hypertext graphs assist users in making sense of their story structures, but despite this can sometimes become challenging to work with, especially as stories grow. Additional enhancements to spatial hypertext graphs can bring further clarity, and understanding how authors use them can aid us in refining the enhancements and introducing new ones.

6.7.2 Principles

Metaphor Testing

Interfaces that use a visual metaphor to represent story structure and connectedness will result in less testing of non-complex stories.

Point 1

Spatial hypertext graph design enables authors to visually make sense of the story structure.

Point 2

Exposing attributes of nodes can aid in making sense of story structure through enhanced context, and when customizable can result in idiosyncratic rules which help to personalize how the author understands their own story structure.

Point 3

Clarity of connectivity between nodes helps with visually following and making sense of the story structure and can be further enhanced by augmenting connections between lines.

Point 4

Highlighting and animating connections between nodes helps to better visualize, follow, and make sense of the structure of a story, particularly in graphs that have a low legibility such as disorderly graphs or those at a physically reduced scale due to zooming.

Point 5

Support for traversal methods is essential to being able to make sense of the story structure, especially with large projects where visual space is limited and elements are frequently off the screen.

Fast Track Testing

Letting users jump to any state of the story enables more rapid and focused testing sessions.

Point 1

Being able to begin testing from any node means that unimportant content for a particular testing session can be skipped over, reducing the need to test from the beginning each time, which means that testing sessions can be rapid and more focused on a particular point as prior content is not traversed.

Point 2

Allowing users to control the story state during testing allows the author to explore specific state configurations and their impact upon story flow.

Point 3

Supporting recording and playback of traversals means that users can repeat focused testing sessions quickly with a reduction of human error in comparison to manually repeating the same traversals.

Structure

Interfaces that use a visual metaphor to represent story structure and connectedness enable idiosyncratic organization and management of an author's own story structure.

Point 1

Spatial hypertext graph design helps with personal organization due to its intuitiveness and lack of restrictions in regard to positioning of content.

Point 2

Supporting arbitrary visual grouping tools that do not impact the structure aids personal organization.

Point 3

Support for traversal methods directly complements the ability to personally organize spatial hypertext graph contents.

Point 4

Being able to identify types and unique instances of nodes helps with personal organization as it lowers cognitive friction and helps users focus on organization rather than identification.

Experimentation

Interfaces that use a visual metaphor to represent story structure and connectedness enable easier experimentation of structure and connectedness.

(In this context, experimenting refers to authors making changes by exploring different structures and configurations. Examples of this include changing the number of outputs a node has, changing where an existing connection goes, and adding or removing nodes.)

Point 1

Spatial hypertext graph design provides greater context within which experimentation can take place.

Point 2

Being able to identify types and unique instances of nodes helps with experimentation as it prevents focus being spent on identification which could slow or hinder the experimentation process.

Point 3

Support for traversal methods directly complements the ease of experimentation.

Point 4

Being able to identify the connected state of links between content helps with experimentation

as it prevents focus being diverted to understanding the connections which could slow or otherwise hinder the experimenting process.

Point 5

Being able to easily create and manage connections between nodes with minimal cognitive friction ensures that the user can focus on the actual restructuring and reorganizing of connectivity between nodes rather than focusing on the details of managing connections.

Branching

Interfaces that use a visual metaphor to represent story structure and connectedness enable easier creation and management of branches.

Point 1

Authors tend to organize nodes to visually mimic story flow which makes it easier to work with branches due to increased local context.

Point 2

Reducing the complexity of creating and manipulating connections enables easier creation and experimentation with branches.

6.8 Conclusion

The study detailed in this chapter set out to validate, further explore, and expand upon the design principles outlined in §4 in contexts beyond that within which they were initially described. The refinement process undertaken allowed for the investigation of the robustness of the principles in new contexts, the modification of the principles based on new data gathered from the study, and worked to mitigate the chance that the principles were particular to their specific authoring environments, consequently improving the generality of their definitions.

The study, conducted with 15 students and five practicing professionals from a variety of backgrounds, was oriented around surveying and interviewing authors in relation to the principles after guiding them through interactive authoring workflows which served both as a reference design through exposure and as a conversational stimulant during interviews. Informing the process was a principle refinement and validation framework that outlined, as described in §6.1, how the original principles were factored into the study and eventually refined. As part of this process, the resulting authoring tool design artifact from §5, which considered the initial principles when being designed and refined, was used to generate a mapping of user experience interactions to the principles. This new mapping was used alongside the authoring tool design artifact to create two interactive authoring workflow guides to walk participants through typical game authoring tasks while exposing them to the principles in a new context. Interleaved with the interactive

authoring workflow guides was a survey that queried participants about the concepts that the principles covered. The guides served as a reference for the survey questions, although it was expected that participants would also draw upon their own experiences. Participants then engaged in a semi-structured interviewed based on the principles, using their survey responses and the interactive authoring workflow guides as stimulants for discussion but allowing for participants to draw upon their own experiences. Finally, the data gathered from the study were used to validate and refine the principles.

The resulting contributions of this study include seven Themes based on the qualitative interview data that highlight evidenced behavioral authoring trends across the participants and a refined list of principles with evidenced constituent points that demonstrate practical applications of each principle's concept. Both are presented in a compact and referenceable format in §6.7. The refined principles differ from the initial principles in that they have been expanded upon to contain several points each that demonstrate evidenced and practical applications of the parent principle's concept, derived from the user experience interaction mapping in §6.2.1. Moreover, they have also been validated and refined based on the evaluation of a new authoring tool design that implemented these principles rather than purely based on observations of existing authoring tools. The evaluation and refinement process led to the *Principle of Contented Authoring* being removed due to a lack of supporting evidence, resulting in a final contribution of five evidenced and demonstrated principles (*Metaphor Testing*, *Fast Track Testing*, *Structure*, *Experimentation*, *Branching*).

These seven Themes and the refined and expanded listing of principles directly answer the third research question of this thesis, which seeks to identify user experience principles for the design of video game narrative authoring tools through empirical and experimental methodologies. With these data, we have explored how different user experience design considerations alter the authoring tool experience, and how these considerations can be utilized when designing new authoring tools.

CHAPTER 7

Conclusions & Future Work

This thesis explored three major aspects of authoring technologies for interactive narrative, chiefly focusing on the subset of video game narrative. The first relates to understanding existing approaches to modeling of narrative and how well those approaches capture video game narrative, ultimately resulting in the development of a new narrative model built to better support video game narrative. This was targeted in §3 which explored existing approaches to modeling video game narrative and applied several models to two narrative games, both serving as inspiration upon which my descriptive concept of *Discoverable Narrative* and my *Novella Narrative Model* were built. The second focuses upon the user interface and user experience design paradigms found within interactive narrative authoring tools that rely on underlying narrative models and the impact that these design choices have upon how authors work. This was explored in the first major study of this thesis detailed in §4, which analyzed a representative sample of authoring tools in an observational user study to identify the impact of user experience and user interface design paradigms upon participants' workflows within the authoring tools, resulting in a preliminary set of authoring tool design principles. The third builds upon the second by seeking to verify and refine these preliminary authoring tool design principles. This was investigated in the second major study of this thesis prepared for in §5 and detailed in §6, which built upon these preliminary principles from the previous major study by applying them in a new experimental context, resulting in a validated, expanded, and refined list of applied authoring tool design principles which can aid informed user experience design of newly created authoring tools. These three queries were outlined in the first chapter as research questions alongside the thesis motivations and scope.

7.1 Contributions

This thesis has presented a range of research contributions surrounding the topic of narrative modeling and user experience design of authoring tools focusing on the subset of video game narrative authoring. The contributions together tackle the three initial research questions outlined at the beginning of the thesis in §1.1. Each of the contributions

made are detailed below in the order that they appear in the thesis body, followed by conclusions regarding the original research questions.

1. Analysis of Delivery Methods & Interface Paradigms

In §2.4.2, a substantial number of IDN authoring tools were sourced and categorized to identify commonalities between their methods of delivery (*standalone*, *web*, *integrated*) and both their primary and utilized high-level interface design paradigms (*form*, *graph*, *text*). This analysis resulted in an understanding of the distribution of authoring tools in regard to both their methods of delivery as well as the kinds of high-level user interface paradigms that they primarily support and otherwise demonstrate, both accompanied by a discussion of the advantages and disadvantages of each approach. This is important as it provides a summary for future researchers of the types of authoring tools that exist, the variance of how they are presented and the associated advantages and disadvantages of each approach, and similarly, the variance of the high-level user interface paradigms that are used, again along with the advantages and disadvantages of each method.

2. An Architecture for Game Narrative Authorship

In §3.1, a complete architecture of video game narrative authoring consisting of several internal components — a narrative model, an authoring tool, an interchange format, and an API — and two external components — an engine and a product — was proposed and later utilized in the thesis. This architecture, while imposing a set of limited constraints, remained generalized and neutral to how the resulting narrative was realized and did not tie itself to a specific engine or genre, therefore applying to a wide variety of scenarios. The benefit of this architecture to future researchers, albeit modest, is that it provides a framework within which they can develop components in the context of a greater system rather than in isolation, which as discussed can result in components being better integrated and aware of their surrounding processes. This was demonstrated in my own use of the architecture, such as how the *Novella Narrative Model*, while still serving as an individual component, was influenced by and able to communicate with other layers of the architecture, and consequently was designed with this communication in mind.

3. Discoverable Narrative

In §3.3, my descriptive concept of *Discoverable Narrative* was developed and presented in response to an identified gap, where contemporary modeling techniques struggled to identify or represent aspects of video game narrative that were discoverable, observable, or experienced by players. The concept consisted of four dimensions — *Tangibility*, *Functionality*, *Clarity*, and *Delivery* — each working together to describe the notion of a related narrative element. This was accompanied by several practical applications demonstrating how its enumerations function. *Discoverable Narrative* also provides an

initial vocabulary for the analysis, categorization, and comparison of such narrative aspects beyond that which was possible prior to its inception.

4. Novella Narrative Model

In §3.4, my own *Novella Narrative Model* was presented in detail followed by worked examples to further explain how the model functions and to demonstrate its flexibility. This model was developed around the aforementioned architecture and was consequently contextualized with respect to its neighboring components in regard to the authoring process, integration into a runtime, and realization of the final narrative. The model contributed several key innovations including native support for game narrative content, extensibility and deference of actions into a runtime implementation, built-in bidirectional communication with a runtime through its own scripting system, inclusion of aspects of *Discoverable Narrative*, and neutrality from both particular runtimes and genres of game. This contribution is important as there is a demonstrable lack of narrative models specific to video game narrative that are built for practical implementation into existing runtimes, particularly whilst also remaining neutral as described above.

5. A Pipeline for Authoring Tool Design

In §5, a custom multi-stage pipeline for the design and development of IDN authoring tools was presented, informed by professional industry-standard techniques. This was also followed by a detailed application and reporting of each phase of the pipeline in an effort to produce a refined authoring tool prototype design suitable for use later in the thesis, but also to demonstrate the functionality and success of the pipeline in practice. The benefit of this contribution is that future researchers can use the pipeline or derivatives of it to inform their authoring tool design and development process which enables a more refined user experience but also a reduction in potential usability problems. This is in comparison to developing an authoring tool blind, where actual users are not considered in the process, which can result in a poorer user experience for the authors.

6. Principles of Authoring Tool Design & Related Themes

In §4, a study was presented that identified a preliminary set of design principles that highlighted relationships between user experience aspects of IDN authoring tools and their potential impact upon the workflow of authors. This was achieved with a novel and bespoke evaluation methodology for evaluating the user experience of IDN authoring tools in a representative environment, done so in a challenging space that has resulted in a gap in the research literature. In §6, this preliminary listing was further explored, expanded upon, validated, and refined in a second study, which again developed and executed a novel and bespoke evaluation methodology as appropriate. The combined result of these two studies was twofold. Firstly, a listing of seven Themes was generated

based on qualitative data analysis that highlighted evidenced behavioral authoring trends of user experience design factors between involved participants. Secondly, the preliminary principles were expanded upon, validated, and otherwise refined to include evidenced constituent points that demonstrate practical applications of each principle's concept, derived from experience with interactive authoring workflows and general experience of participants in authoring tools of their choice. The final refined principles include *Metaphor Testing*, *Fast Track Testing*, *Structure*, *Experimentation*, and *Branching*. Both the Themes and resulting principles and their constituent points, as summarized in §6.7, serve as guidance for the development of future authoring tools and as a vocabulary for further discussion of user experience issues facing authorship technologies in this research space.

Research Questions

The above contributions have been created during this work to investigate and provide evidence to answer the three research questions proposed at the start of this thesis:

1. To what extent do existing narrative models capture the complexities of video game narrative and how might a new model better support this?

2. How do existing paradigms within authoring tool interface designs impact the user experience and workflow of authoring video game narrative?

3. What are the user experience principles for the design of video game narrative authoring tools based on empirical and experimental methodologies?

The first research question focuses on understanding how and to what extent existing approaches to narrative modeling are able to capture the nuances and complexities of video game narrative, what they are unable to capture, and how this information can be used to build a new model that better represents video game narrative. Evidence for this question comes from §3 which presented an exploration into existing approaches to modeling video game narrative and an application of a variety of existing models to two contemporary video games with differing narrative complexity in order to evaluate and understand their applicability to the video game narrative space. In response to the findings inadequately representing optional narrative content, my descriptive concept of *Discoverable Narrative* was introduced along with several worked examples, providing a vocabulary for elements of video game narrative that are discovered, observed, or experienced. Taking into account the advantages and limitations of the analyzed models, my own complete narrative model, *Novella*, was introduced alongside a generalized authoring architecture to help better contextualize the model with respect to its neighboring components in regard to the authoring process, integration into a runtime, and realization of the final narrative. This model, built with integration into runtimes in mind, introduced advantages such as mature support for game content, plentiful extensibility

and deference of actions to a runtime implementation (e.g., running code within a game itself), bidirectional communication with a runtime (e.g., both the model and a game can communicate with each other), and neutrality from any particular runtime or genre.

The second research question investigates how the use of interface design paradigms found within existing authoring tools impacts the authoring experience and how authors resultingly choose to work. Evidence for this question comes from the study presented in §4 which set out to identify the impact of different user experience and interface paradigms on the way that authors work using a novel and bespoke evaluation methodology for IDN authoring tools in a game narrative context. This study identified three representative authoring tools that each present a different prominent interface design paradigm and included them in an observational protocol where participants completed tasks representative of game writing followed by interviews to further understand the authoring experience when using one of these paradigms. The outputs from this study directly answer this research question in the form of an analysis of observed workflows between the three authoring tools and the formation of six preliminary design principles that both highlight relationships between the design of user interfaces in IDN authoring tools and their potential impact upon the workflow of authors.

The third research question seeks how we can create user experience principle guidelines for the design of video game narrative authoring tools from empirical evidence gathered in experimental methodologies which could be used as indicators by future developers to better understand the impact of their design decisions upon the user's experience with the tool that they are creating. Evidence for this question comes from §6 which built and executed a principle refinement and validation framework that took the preliminary principles from §4 and further explored, expanded upon, validated, and refined them in new contexts to create a strengthened listing. This was achieved by taking the refined authoring tool prototype design from §5, which was based upon the preliminary principles, mapping aspects of its user experience design to the principles, building interactive authoring workflow artifacts that guide participants through typical authoring experiences while exposing them to the principles, and then identifying participant behavior and thoughts on the concepts of each principle through surveys and interviews. The outputs of this study, directly targeting this research question, include a set of seven Themes based on qualitative data analysis that highlight evidenced behavioral authoring trends of user experience design factors, and a refined list of five design principles (*Metaphor Testing*, *Fast Track Testing*, *Structure*, *Experimentation*, *Branching*) expanded to include evidenced constituent points that demonstrate practical applications of each principle's concept. These Themes and refined principles and their constituent points contribute generalized user experience design guidance for the informed development of future authoring tools, but also provide a vocabulary to better describe the user experience issues facing IDN authoring tools.

7.2 Publications

Throughout the PhD that this thesis is a part of, several publications have been created and presented in a journal and a variety of conferences and workshops in the topics of narrative modeling and user experience design of authoring tools within the video game narrative subdomain. Future publications expanding on this are also in development. The listing that follows summarizes each of the publications in chronological order, noting their relation to the thesis.

1. *Novella: A Proposition for Game-Based Storytelling (2018) [89]*
   This paper was presented at the *Narrative and Hypertext Workshop* alongside the *Hypertext 2018* conference in Baltimore, USA. The paper presented a shortened application of existing narrative models to two contemporary video games with differing narrative complexity, as found in a more mature form in §3.2. The main contribution of the paper was an earlier revision of the *Novella Narrative Model* presented in §3.4, which also lightly touched upon the formation of *Discoverable Narrative* as proposed and explained in a more refined form in §3.3.

2. *Novella: An Authoring Tool for Interactive Storytelling in Games (2018) [83]*
   This paper was published and presented as part of a Doctoral Consortium program at *ICIDS 2018* in Dublin, Ireland. A more mature form of *Discoverable Narrative* with only minor differences to that in §3.3 was detailed, alongside initial exploration into usability and user experience of authoring tools, as well as an initial approach to the building of a user-friendly authoring tool prototype.

3. *Contemporary Issues in Interactive Storytelling Authoring Systems (2018) [85]*
   This full paper was published and presented at *ICIDS 2018* in Dublin, Ireland, and was also nominated for a Best Paper award. The paper firstly presents an analysis of contemporary authoring tools, focusing on delivery methods, user interface paradigms, and the troubles of availability, and is in fact the same analysis as used in the preparation for the first major study of this thesis as outlined in §4. However, as mentioned elsewhere, §2.4.2 built upon this collection at a later date, but conclusions remained the same albeit with greater evidence. Secondly, the paper performed analysis of two popular authoring tools with different user interface paradigms and analyzed them with select heuristics and laws. This was concluded by taking the prominent troubles found in the previous analysis and demonstrating how they could be better approached in a new authoring tool prototype design.

4. *Define "Authoring Tool": A Survey of Interactive Narrative Authoring Tools (2018) [86]*
   This paper was presented at the *Authoring for Interactive Storytelling Workshop* alongside the *ICIDS 2018* conference in Dublin, Ireland. One of the workshop themes was "*What is a tool, anyway, in the context of authoring for interactive storytelling?*". This

paper contributed to this topic by performing a clustering of contemporary authoring tools based on a wide range of features and then analyzing the resulting clusters to determine 'genres' of tool and to understand the spread of different approaches for supporting authoring of interactive storytelling. This is the same clustering analysis as used in the preparation for the first major study of this thesis as outlined in §4.

5. *Novella 2.0: A Hypertextual Architecture for Interactive Narrative in Games (2019) [87]*
   This full paper was published and presented at *Hypertext 2019* in Hof, Germany. It began by describing the various approaches that others have taken to modeling video game narrative, as also described in this thesis in §3.2. This was followed by the main focus of the paper, the *Novella Narrative Model*, which takes shape almost equal to that presented in this thesis in §3.4. Moreover, an accompanying system architecture for the practical implementation and realization of the model was described, which can be found in a more generalized form in this thesis in §3.1.

6. *A Novel Design Pipeline for Authoring Tools (2020) [84]*
   This paper was published and presented at *ICIDS 2020* in Bournemouth, UK, although the conference was conducted virtually due to an ongoing pandemic. The contribution that the paper made was the same pipeline for authoring tool design as described and followed in §5, although in a shorter format. The accompanying conference presentation stimulated discussion around the topic of informed authoring tool design procedures, with those participating noting the importance of this work and its motivation for the research community to further investigate the problems surrounding usability and user experience of authoring tool design.

7. *Use of Tools: UX Principles for Interactive Narrative Authoring Tools (2021) [88]*
   This article was published in the prestigious ACM *Journal on Computing and Cultural Heritage* (JOCCH). The article was written based on the interface paradigms study detailed in this thesis in §4 including all preparation (selecting representative tools from clusters and building a representative task-based narrative based on contemporary video game narrative storytelling techniques), the organization and execution of the study itself, the analysis process, and the resulting conclusions. Being a part of JOCCH, the study's contributed principles were also demonstrably linked to other authoring tools found in the cultural heritage sector in addition to IDN authoring tools as discussed in this thesis.

7.3 Future Work

While this thesis has made a variety of contributions, it has also raised several potential points of interest for future research in this space that are beyond the scope of this work.

As alluded to in the conclusion of §4, it is possible that the genre and style of the template story used in the study may have influenced the authoring experience, resulting

in principles that were biased towards those particular genres and styles, although the methodology did take several steps to mitigate this. Future work could investigate this further by intentionally varying the genre and style of stories in the same authoring tool to better understand the impact that these variations have upon the authoring experience.

In the principle refinement and validation study in §6, interactive authoring workflows created in *Adobe XD* were used in place of an implemented desktop application, which was suitable for the purposes of the study as justified in §6.2.2. To gain a fuller understanding of the impact that these principles have on an author's experience, future work could fully implement the authoring tool design as a desktop application instead of using high-fidelity interactive prototypes and run a longitudinal study that involves an expert authoring a mature game. However, as established in §4, longitudinal studies are a problematic area for IDN authoring tool research that must be overcome.

Additionally, in the principle refinement and validation process, the data gathered for several of the points was sufficient for validation to take place in the context of this study but indicated areas of potential interest for future work. For example, prior to refinement, the eighth point of *Metaphor Testing*, the third point of *Structure*, and the third point of *Experimentation* all indicate that traversal methods aid varying aspects of the authoring process, but the exact nuances of the impacts of traversal and the various ways that they are utilized were not ascertained from the gathered data. While the study confirmed the importance of these aspects, a future study that is focused on traversal methods could be dedicated to understanding the reasons that traversal methods are critical and the differences between those available in contemporary authoring tools. It is important to note that this does not mean the study is incomplete, but rather that what has been done is valuable and sufficient for the claims particular to this study, and that the gathered data raises interesting nuances that could be further explored by future projects that are beyond the scope of this thesis.

The *Recognize the need for distinct multi-stage planning and building workflows* Theme as part of the analysis of the principle and validation study in §6 demonstrated how for some authors there is a clear distinction between the planning and building of a narrative, and that existing authoring tools often lack support for their styles of planning. Recognizing this separation could help us to build more informed tools that better support authors by integrating solutions to both planning and building aspects of the creation of interactive narratives. Future work could investigate this in more depth by creating and testing variations of a bespoke authoring tool that, for example, separates planning and building into separate applications, joins planning and building into the same application but keeps them distinct, joins planning and building into the same application but tightly integrates them, and other similar amalgams.

Discoverable Narrative, as detailed in §3.3, provides a set of enumerations that have successfully been able to describe aspects of discoverable, observable, or experienced narrative within video games. However, it does so at a high level of granularity due to its

descriptive nature, which is suitable for the purposes of this thesis. Future work could build upon this concept by subdividing the enumerated dimensions, or even appending new ones, to allow for finer granularity of the narrative events that are being captured. This could also further enhance its ability to categorize and compare appropriate narrative events. Similarly, while the *Novella Narrative Model* presented in §3.4 is mature and is able to capture and implement fine levels of detail for video game narrative, it does have only limited support for *Discoverable Narrative* in the form of *Discoverable* nodes, use of *Attributes*, and general structural support for such events. As was found when integrating it into the model, supporting each variation of *Discoverable Narrative* comes with its own potential increase in modeling and structural complexity. It is possible that a refinement to the *Discoverable Narrative* model to further increase the granularity, as mentioned above, would make it more suitable for low-level narrative models.

7.4 Concluding Remarks

In this thesis, I have tackled the structural modeling of video game narrative and the impact of user experience design choices of IDN authoring tools on the author's experience.

Whether or not video games can be considered a narrative medium has been the source of debate in times past, and as explained in §2.3, it is now widely accepted that video games are not a direct subset of narrative, but instead that they contain elements that have qualities suitable for analysis from both the perspective of game studies and more traditional narrative theory. Consequently, one would expect to find plentiful narrative models that specifically target video game narrative and the complexities that it introduces, but these models are comparably few in number, perhaps due to how young and challenging the field is. Those that do exist typically have restrictions in genre (i.e., make assumptions and choices in their design which inherently lock them into a subset of genres) or are sometimes more theoretical than practical. By contributing the *Novella Narrative Model* and the concept of *Discoverable Narrative*, steps are taken to provide first-class support for modeling video game narrative in a way that is independent of genre and is suitable for direct implementation. It is my hope that these contributions will inspire other video game narrative researchers to develop models with a similar focus, further enriching modeling space and the vocabulary that I have introduced here.

Early interactive narratives were implemented manually in low-level programming languages, and over the several decades that followed, authoring tools were gradually introduced as discussed in §2.4.1, evolving into the fully-featured graphical applications that we are familiar with today. Despite the rich history, popularity, and recognized importance of quality user experience within these authoring tools, there are few works that focus on user experience design within authoring tools, the emphasis instead being on the experiences created, which has led to a UX focus in this research space on the audience rather than the authors. In the broader field of user experience design, there is

a vast array of established principles, heuristics, and evaluation methods that apply to a range of fields, yet these appear to be overlooked in the IDN authoring tool space despite its utmost importance in making the authoring experience more accessible, particularly for non-technical users. To remedy this, I introduced a generalized multi-stage pipeline for the design and development of IDN authoring tools informed by professional industry-standard techniques. The hope behind this contribution is that future researchers and developers will be able to use the pipeline or derivatives of it to inform their authoring tool design and development process to reduce usability problems and improve the overall user experience, and of course to raise general awareness of the importance of following proper user experience design practices when creating authoring tools. Similarly, my seven Themes and five principles along with their constituent points contribute generalized and actionable domain-specific user experience guidance for the informed design of IDN authoring tools in an area that is often overlooked and in need of wider attention. Moreover, the guidance provides an additional vocabulary to better describe the user experience issues facing authoring tools. This represents a significant step in a sparse research space and I hope that it motivates other researchers to continue along this line of work by discovering and evidencing other principles, either generic or relating to particular genres of story and tool, and that it raises awareness of not only the importance of user experience in authoring tool design and the impact that seemingly innocent design decisions can have on the overall user experience of authoring tools, but also that it motivates further discussion and research in this area going forward.

References

[1] Antti Aarne, *The Types of the Folktale: A Classification and Bibliography*, 2nd edition, translated by Stith Thompson. Suomalainen Tiedeakatemia, Academia Scientiarum Fennica, 1961, ISBN: 951-41-0132-4 (cited on page 10).

[2] Antti Aarne, *Verzeichnis der Märchentypen*, 3. Folklore Fellows' Communications, 1910 (cited on page 10).

[3] Espen Aarseth, "A Narrative Theory of Games," in *Proceedings of the International Conference on the Foundations of Digital Games*, FDG '12, New York, NY, USA: ACM, 2012, pages 129–133, ISBN: 978-1-4503-1333-9. DOI: 10.1145/2282338.2282365 (cited on pages 8, 30, 81, 87, 92).

[4] Espen Aarseth, *Cybertext: Perspectives on Ergodic Literature*. Baltimore, Maryland, USA: Johns Hopkins University Press, 1997, ISBN: 0-8018-5579-9 (cited on pages 18, 19, 21).

[5] Espen Aarseth, "Genre Trouble: Narrativism and the Art of Simulation," in *First Person: New Media As Story, Performance, and Game*, Noah Wardrip-Fruin and Pat Harrigan, Eds., Cambridge, MA: MIT Press, 2004, pages 45–55, ISBN: 978-0-262-23232-6 (cited on page 30).

[6] Ernest Adams, *Fundamentals of Game Design*, 3rd edition. Thousand Oaks, CA, USA: New Riders Publishing, 2014, ISBN: 978-0-321-92967-9 (cited on pages 37, 53, 88, 130).

[7] Lauralee Alben, "Quality of Experience: Defining the Criteria for Effective Interaction Design," *Interactions*, volume 3, number 3, pages 11–15, 1996, ISSN: 1072-5520. DOI: 10.1145/235008.235010 (cited on page 63).

[8] Roberto Andreoli, Angela Corolla, Armando Faggiano, Delfina Malandrino, Donato Pirozzi, Mirta Ranaldi, Gianluca Santangelo, and Vittorio Scarano, "A Framework to Design, Develop, and Evaluate Immersive and Collaborative Serious Games in Cultural Heritage," *Journal on Computing and Cultural Heritage*, volume 11, number 1, 4:1–4:22, 2017, ISSN: 1556-4673. DOI: 10.1145/3064644 (cited on page 123).

[9] Ahlem Assila, Káthia Marçal de Oliveira, and Houcine Ezzedine, "Standardized Usability Questionnaires: Features and Quality Focus," *electronic Journal of Computer Science and Information Technology*, volume 6, number 1, 2016, ISSN: 1985-7721 (cited on page 73).

[10] Frederik Aust, Birk Diedenhofen, Sebastian Ullrich, and Jochen Musch, "Seriousness Checks Are Useful to Improve Data Validity in Online Research," *Behavior Research Methods*, volume 45, number 2, pages 527–535, 2013, ISSN: 1554-3528. DOI: 10.3758/s13428-012-0265-2 (cited on pages 74, 205).

[11] Mieke Bal, *Narratology: Introduction to the Theory of Narrative*. Toronto: University of Toronto Press, 2007, ISBN: 978-0-8020-7806-3 (cited on page 9).

[12] Olivier Balet, "INSCAPE: an authoring platform for interactive storytelling," in *Proceedings of the 4th international conference on Virtual storytelling: using virtual reality technologies for storytelling*, Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, Dec. 5, 2007, pages 176–177, ISBN: 978-3-540-77037-4. DOI: 10.1007/978-3-540-77039-8_15 (cited on pages xvii, 27, 54).

[13] Aaron Bangor, Philip T. Kortum, and James T. Miller, "An Empirical Evaluation of the System Usability Scale," *International Journal of Human-Computer Interaction*, volume 24, number 6, pages 574–594, 2008, ISSN: 1044-7318. DOI: 10.1080/10447310802205776 (cited on page 73).

[14] Javier A. Bargas-Avila and Kasper Hornbæk, "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, New York, NY, USA: ACM, 2011, pages 2689–2698, ISBN: 978-1-4503-0228-9. DOI: 10.1145/1978942.1979336 (cited on page 63).

[15] Roland Barthes, *S/Z*, translated by Richard Miller. Hill and Wang, 1970, ISBN: 978-0-374-52167-7 (cited on pages 11, 12).

[16] Roland Barthes, "To Write: An Intransitive Verb?" In *The Structuralist Controversy: The Languages of Criticism and the Sciences of Man*, Richard Macksey and Eugenio Donato, Eds., 4th edition, Baltimore, Maryland, USA: Johns Hopkins University Press, 1979, pages 134–145, ISBN: 0-8018-1047-7 (cited on page 11).

[17] Roland Barthes and Lionel Duisit, "An Introduction to the Structural Analysis of Narrative," *New Literary History*, volume 6, number 2, pages 237–272, 1975, ISSN: 00286087, 1080661X. DOI: 10.2307/468419 (cited on page 88).

[18] Mark Bernstein, "Can We Talk About Spatial Hypertext?" In *Proceedings of the 22nd ACM Conference on Hypertext and Hypermedia*, HT '11, New York, NY, USA: ACM, 2011, pages 103–112, ISBN: 978-1-4503-0256-2. DOI: 10.1145/1995966.1995983 (cited on pages 57, 67, 143, 147–149, 188, 191, 192, 225).

[19] Mark Bernstein, "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*, HT '98, Pittsburgh, Pennsylvania, USA: ACM, 1998, pages 21–29, ISBN: 978-0-89791-972-2. DOI: 10.1145/276627.276630 (cited on page 96).

[20] Mark Bernstein, "Storyspace 1," in *Proceedings of the Thirteenth ACM Conference on Hypertext and Hypermedia*, HT '02, New York, NY, USA: ACM, 2002, pages 172–181, ISBN: 978-1-58113-477-3. DOI: 10.1145/513338.513383 (cited on pages xvii, 130).

[21] Mark Bernstein, "Storyspace 3," in *Proceedings of the 27th ACM Conference on Hypertext and Social Media*, HT '16, New York, NY, USA: ACM, 2016, pages 201–206, ISBN: 978-1-4503-4247-6. DOI: 10.1145/2914586.2914624 (cited on pages xvii, 54, 130).

[22] Mark Bernstein, "The Navigation Problem Reconsidered," in *Hypertext/hypermedia handbook*, USA: McGraw-Hill, Inc., 1991, pages 285–297, ISBN: 978-0-07-016622-6 (cited on page 18).

[23] Mark Bernstein, David E. Millard, and Mark J. Weal, "On Writing Sculptural Hypertext," in *Proceedings of the Thirteenth ACM Conference on Hypertext and Hypermedia*, HT '02, New York, NY, USA: ACM, 2002, pages 65–66, ISBN: 978-1-58113-477-3. DOI: 10.1145/513338.513355 (cited on page 111).

[24] Sebastian Hurup Bevensee and Henrik Schoenau-Fog, "Conceptualizing Productive Interactivity in Emergent Narratives," in *Interactive Storytelling*, Lecture Notes in Computer Science, Cham: Springer, 2013, pages 61–64, ISBN: 978-3-319-02756-2. DOI: 10.1007/978-3-319-02756-2_7 (cited on page 31).

[25] Hugh Beyer and Karen Holtzblatt, *Contextual Design: Defining Customer-Centered Systems*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, ISBN: 978-0-08-050304-2 (cited on page 75).

[26] Jim Bizzochi, "Games and Narrative: An Analytical Framework," *Loading — The Journal of the Canadian Games Studies Association*, volume 1, number 1, pages 5–10, 2007, ISSN: 1923-2691 (cited on pages 30, 87).

[27] Staffan Björk and Jussi Holopainen, "Describing Games: An Interaction-Centric Structural Framework," in *Proceedings of the 2003 DiGRA International Conference: Level Up*, volume 2, 2003, pages 4–6 (cited on page 87).

[28] Wayne C. Booth, *The Rhetoric of Fiction*, 2nd edition. Chicago: University of Chicago Press, 1983, ISBN: 978-0-226-06558-8 (cited on pages 130, 287).

[29] Barbaros Bostan and Orcun Turan, "Deconstructing Game Stories with Propp's Morphology," in *Proceedings of Eurasia Graphics 2017: International Conference on Computer Graphics, Animation, and Gaming Technologies*, Istanbul, Turkey, 2017 (cited on pages 87–91, 281).

[30] Jeffrey E. Brand and Scott J. Knight, "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, 2005 (cited on pages 30, 87).

[31] Steve Bromley, "Interviewing Players," in *Games User Research*, Oxford University Press, 2018, ISBN: 978-0-19-879484-4 (cited on page 172).

[32] John Brooke, "SUS: A 'Quick and Dirty' Usability Scale," in *Usability Evaluation In Industry*, P. Jordan, B. Thomas, I McClelland, and B Weerdmeester, Eds., London: Taylor and Francis, 1996, pages 189–194 (cited on page 73).

[33] Peter Brooks, *Reading for the Plot: Design and Intention in Narrative*. Harvard University Press, 1992, ISBN: 978-0-674-74892-7 (cited on pages 9, 11).

[34] Andrew Brusentsev, Michael Hitchens, and Deborah Richards, "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, New York, NY, USA: ACM, 2012, 2:1–2:10, ISBN: 978-1-4503-1410-7. DOI: 10.1145/2336727.2336729 (cited on pages 85, 88, 91).

[35] Vannevar Bush, "As We May Think," *Interactions*, volume 3, number 2, pages 35–46, 1996, ISSN: 1072-5520. DOI: 10.1145/227181.227186 (cited on page 13).

[36] Elke Cabooter, Bert Weijters, Maggie Geuens, and Iris Vermeir, "Scale Format Effects on Response Option Interpretation and Use," *Journal of Business Research*, volume 69, number 7, pages 2574–2584, 2016, ISSN: 0148-2963. DOI: 10.1016/j.jbusres.2015.10.138 (cited on pages 74, 202).

[37] Joseph Campbell, *The Hero with a Thousand Faces*. New World Library, 2008, ISBN: 978-1-57731-593-3 (cited on pages 27, 81, 82).

[38] Steve Carmody, Theodor Nelson, David Rice, and Andy van Dam. (1968). "Hypertext Editing System Manuals," Brown Digital Repository, [Online]. Available: https://repository.library.brown.edu/studio/item/bdr:960882 (cited on pages xvii, 15).

[39] Fred Charles, Marc Cavazza, Cameron Smith, Gersende Georg, and Julie Porteous, "Instantiating Interactive Narratives from Patient Education Documents," in *Artificial Intelligence in Medicine*, Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2013, pages 273–283, ISBN: 978-3-642-38326-7. DOI: 10.1007/978-3-642-38326-7_39 (cited on page 28).

[40] Fred Charles, David Pizzi, Marc Cavazza, Thurid Vogt, and Elisabeth André, "EmoEmma: Emotional Speech Input for Interactive Storytelling," in *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '09, volume 2, Budapest, Hungary: International Foundation for Autonomous Agents and Multiagent Systems, 2009, pages 1381–1382, ISBN: 978-0-9817381-7-8 (cited on pages xvii, 54, 55, 126, 130).

[41] Simon Chauvin, Guillaume Levieux, Jean-Yves Donnart, and Stéphane Natkin, "An Out-of-Character Approach to Emergent Game Narratives," in *Proceedings of the 9th International Conference on the Foundations of Digital Games*, FDG '14, Society for the Advancement of the Science of Digital Games, 2014 (cited on page 31).

[42] Yun-Gyung Cheong and R. Michael Young, "A Framework for Summarizing Game Experiences As Narratives," in *Proceedings of the Second AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, AIIDE '06, Marina del Rey, California: AAAI Press, 2006, pages 106–108 (cited on page 87).

[43] *Choose Your Own Adventure (novel series)*. Bantam Books, 1979 (cited on pages 3, 33).

[44] Jeff Conklin, "Hypertext: An Introduction and Survey," *Computer*, volume 20, number 9, pages 17–41, 1987, ISSN: 1558-0814. DOI: 10.1109/MC.1987.1663693 (cited on pages 16, 57).

[45] Alan Cooper, *The Inmates Are Running the Asylum: Why High Tech Products Drive Us Crazy and How to Restore the Sanity*, 2nd edition. Pearson Higher Education, 2004, ISBN: 978-0-672-32614-1 (cited on pages 65, 124).

[46] Greg Costikyan, "Where Stories End and Games Begin," *Game Developer*, volume 7, number 9, pages 44–53, 2000 (cited on page 30).

[47] Andries van Dam, "Hypertext '87: Keynote Address," *Communications of the ACM*, volume 31, number 7, pages 887–895, 1988, ISSN: 0001-0782. DOI: 10.1145/48511.48519 (cited on page 17).

[48] Stephen Davies, "Still Building the Memex," *Communications of the ACM*, volume 54, number 2, pages 80–88, 2011, ISSN: 0001-0782. DOI: 10.1145/1897816.1897840 (cited on page 13).

[49] Hugh C. Davis, Wendy Hall, Ian Heath, Gary J. Hill, and Robert J. Wilkins, "MICROCOSM: An Open Hypermedia Environment for Information Integration," Monograph, 1992. [Online]. Available: https://eprints.soton.ac.uk/250713 (cited on pages xvii, 16).

[50] Douglas R. Dechow and Daniele C. Struppa, Eds., *Intertwingled: The Work and Influence of Ted Nelson*, History of Computing. Springer, 2015, ISBN: 978-3-319-16924-8. DOI: 10.1007/978-3-319-16925-5 (cited on page 14).

[51] Guylain Delmas, Ronan Champagnat, and Michel Augeraud, "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, Berlin, Heidelberg: Springer, 2007, pages 187–195, ISBN: 978-3-540-77037-4. [Online]. Available: http://dl.acm.org/citation.cfm?id=1777851.1777873 (cited on pages 82, 84).

[52] Steven J. DeRose and Andries van Dam, "Document Structure and Markup in the FRESS Hypertext System," *Markup Languages*, volume 1, number 1, pages 7–32, 1999, ISSN: 1099-6621. DOI: 10.1162/109966299751940814 (cited on pages xvii, 17).

[53] Heather Desurvire, Martin Caplan, and Jozsef Toth, "Using Heuristics to Evaluate the Playability of Games," in *CHI '04 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '04, New York, NY, USA: ACM, 2004, pages 1509–1512, ISBN: 978-1-58113-703-3. DOI: 10.1145/985921.986102 (cited on pages 72, 152).

[54] Heather Desurvire and Max Kreminski, "Are Game Design and User Research Guidelines Specific to Virtual Reality Effective in Creating a More Optimal Player Experience? Yes, VR PLAY," in *Design, User Experience, and Usability: Theory and Practice*, Lecture Notes in Computer Science, Cham: Springer, 2018, pages 40–59, ISBN: 978-3-319-91797-9. DOI: 10.1007/978-3-319-91797-9_4 (cited on pages 72, 152).

[55] Heather Desurvire and Magy Seif El-Nasr, "Methods for Game User Research: Studying Player Behavior to Enhance Game Design," *IEEE Computer Graphics and Applications*, volume 33, number 4, pages 82–87, 2013, ISSN: 1558-1756. DOI: 10.1109/MCG.2013.61 (cited on page 157).

[56] Heather Desurvire and Charlotte Wiberg, "Game Usability Heuristics (PLAY) for Evaluating and Designing Better Games: The Next Iteration," in *Online Communities and Social Computing*, A. Ant Ozok and Panayiotis Zaphiris, Eds., Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2009, pages 557–566, ISBN: 978-3-642-02774-1. DOI: 10.1007/978-3-642-02774-1_60 (cited on pages 72, 123, 152, 157).

[57] Heather Desurvire and Charlotte Wiberg, "Master of the Game: Assessing Approachability in Future Game Design," in *CHI '08 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '08, New York, NY, USA: ACM, 2008, pages 3177–3182, ISBN: 978-1-60558-012-8. DOI: 10.1145/1358628.1358827 (cited on pages 72, 123, 152).

[58] Debra Dixon, *Goal, Motivation and Conflict: The Building Blocks of Good Fiction*. Bell Bridge Books, 2013, ISBN: 978-1-61194-318-4 (cited on pages 83, 85).

[59] Walter Doherty and Arvind Thadhani, "The Economic Value of Rapid Response Time," *IBM Report*, 1982 (cited on page 68).

[60] Stéphane Donikian and Jean-Noël Portugal, "Writing Interactive Fiction Scenarii with DraMachina," in *Technologies for Interactive Digital Storytelling and Entertainment*, Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2004, pages 101–112, ISBN: 978-3-540-27797-2. DOI: 10.1007/978-3-540-27797-2_14 (cited on pages xvii, 54).

[61] Hermann Ebbinghaus, *Memory: A Contribution to Experimental Psychology*. New York: Teachers College, Columbia University, 1913 (cited on page 71).

[62] *Endless Quest (novel series)*. TSR, Inc., 1982 (cited on page 3).

[63] Douglas Engelbart, "A Conceptual Framework for the Augumentation of Man's Intellect," in *Vistas in Information Handling*, volume 1, Washington, DC, USA: Spartan Books, 1963, pages 1–29 (cited on pages xvii, 16).

[64] Victor Erlich, *Russian Formalism: History, Doctrine*, Slavistic Printings and Reprintings 4. De Gruyter Mouton, 1980, ISBN: 978-90-279-0450-8. DOI: 10.1515/9783110873375. [Online]. Available: https://www.degruyter.com/view/title/3064 (cited on page 9).

[65] Clara Fernández-Vara, "Game Spaces Speak Volumes: Indexical Storytelling," in *Proceedings of the 2011 DiGRA International Conference: Think Design Play*, DiGRA '11, Digital Games Research Association, 2011 (cited on page 36).

[66] Janet Fiderio, "A Grand Vision," *BYTE*, volume 13, number 10, pages 237–244, 1988, ISSN: 0360-5280 (cited on page 14).

[67] Syd Field, *Screenplay: The Foundations of Screenwriting*. New York, USA: Bantam Dell, 2005, ISBN: 978-0-385-33903-2 (cited on page 8).

[68] Kraig Finstad, "The Usability Metric for User Experience," *Interacting with Computers*, volume 22, number 5, pages 323–327, 2010, ISSN: 0953-5438. DOI: 10.1016/j.intcom.2010.04.004 (cited on page 73).

[69] Paul M. Fitts, "The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement," *Journal of Experimental Psychology*, volume 47, number 6, pages 381–391, 1954, ISSN: 0022-1015(Print). DOI: 10.1037/h0055392 (cited on page 68).

[70] Edward Forster, *Aspects of the Novel*. Penguin Classics, 2005, ISBN: 978-0-14-144169-6 (cited on pages 8, 88, 92).

[71] Gonzalo Frasca, "Ludologists Love Stories, Too: Notes From a Debate That Never Took Place," in *Proceedings of the 2003 DiGRA International Conference: Level Up*, 2003 (cited on page 30).

[72] Gonzalo Frasca. (1999). "Ludology meets Narratology: Similitude and Differences Between (Video)Games and Narrative," [Online]. Available: https://ludology.typepad.com/weblog/articles/ludology.htm (cited on page 30).

[73] Anna-Katharina Frison and Pamela Zotz, "The Intersection of User Experience (UX), Customer Experience (CX), and Brand Experience (BX)," in *User Experience Is Brand Experience: The Psychology Behind Successful Digital Products and Services*, Management for Professionals, Felix van de Sand, Anna-Katharina Frison, Pamela Zotz, Andreas Riener, and Katharina Holl, Eds., Cham: Springer, 2020, pages 71–93, ISBN: 978-3-030-29868-5. DOI: 10.1007/978-3-030-29868-5_5 (cited on page 63).

[74] Tracy Fullerton, *Game Design Workshop: A Playcentric Approach to Creating Innovative Games*, 4th edition. CRC Press, 2018, ISBN: 978-1-138-09877-0 (cited on page 221).

[75] Jill Gerhardt-Powals, "Cognitive Engineering Principles for Enhancing Human-Computer Performance," *International Journal of Human-Computer Interaction*, volume 8, number 2, pages 189–211, 1996, ISSN: 1044-7318. DOI: 10.1080/10447319609526147 (cited on pages 65, 72).

[76] Pablo Gervás, "Propp's Morphology of the Folk Tale as a Grammar for Generation," in *2013 Workshop on Computational Models of Narrative*, OpenAccess Series in Informatics, volume 32, Dagstuhl, Germany: Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2013, pages 106–122, ISBN: 978-3-939897-57-6. DOI: 10.4230/OASIcs.CMN.2013.106 (cited on page 27).

[77] Clive Gifford and Robert Young, *Creating Adventures on Your Commodore 64*. London: Interface Publications, 1984, ISBN: 0-907563-49-X (cited on page 45).

[78] Florian Glock, Anne Junker, Marina Kraus, Christian Lehrian, Alexander Schäfer, Steve Hoffmann, and Ulrike Spierling, ""Office Brawl": A Conversational Storytelling Game and Its Creation Process," in *Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology*, ACE '11, New York, NY, USA: ACM, 2011, 88:1–88:2, ISBN: 978-1-4503-0827-4. DOI: 10.1145/2071423.2071529 (cited on pages xvii, 54).

[79] S. Göbel, L. Salvatore, and R. Konrad, "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*, Florence, Italy: IEEE, 2008, pages 103–110. DOI: 10.1109/AXMEDIS.2008.45 (cited on pages xvii, 27, 54, 126).

[80] Stefan Göbel, Oliver Schneider, Ido Iurgel, Axel Feix, Christian Knöpfle, and Alexander Rettig, "Virtual Human: Storytelling and Computer Graphics for a Virtual Human Platform," in *Technologies for Interactive Digital Storytelling and Entertainment*, Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2004, pages 79–88, ISBN: 978-3-540-27797-2. DOI: 10.1007/978-3-540-27797-2_11 (cited on pages xviii, 54).

[81] Elizabeth Goodman, Mike Kuniavsky, and Andrea Moed, *Observing the User Experience: A Practitioner's Guide to User Research*. Elsevier, Sep. 1, 2012, ISBN: 978-0-12-384870-3 (cited on pages 74, 75, 124).

[82] Will Grant, *101 UX Principles: A Definitive Design Guide*. Packt Publishing, 2018, ISBN: 978-1-78883-736-1 (cited on page 178).

[83] Daniel Green, "Novella: An Authoring Tool for Interactive Storytelling in Games," in *Interactive Storytelling*, Lecture Notes in Computer Science, Cham: Springer, 2018, pages 556–559, ISBN: 978-3-030-04028-4. DOI: 10.1007/978-3-030-04028-4_66 (cited on pages xx, 77, 259).

[84] Daniel Green, Charlie Hargood, and Fred Charles, "A Novel Design Pipeline for Authoring Tools," in *Interactive Storytelling*, Lecture Notes in Computer Science, Cham: Springer, 2020, pages 102–110, ISBN: 978-3-030-62516-0. DOI: 10.1007/978-3-030-62516-0_9 (cited on pages xx, 155, 260).

[85] Daniel Green, Charlie Hargood, and Fred Charles, "Contemporary Issues in Interactive Storytelling Authoring Systems," in *Interactive Storytelling*, Lecture Notes in Computer Science, Cham: Springer, 2018, pages 501–513, ISBN: 978-3-030-04028-4. DOI: 10.1007/978-3-030-04028-4_59 (cited on pages xx, 124, 259).

[86] Daniel Green, Charlie Hargood, and Fred Charles, "Define "Authoring Tool": A Survey of Interactive Narrative Authoring Tools," presented at the Authoring for Interactive Storytelling Workshop, 2018 (cited on pages xx, 43, 124, 259).

[87] Daniel Green, Charlie Hargood, and Fred Charles, "Novella 2.0: A Hypertextual Architecture for Interactive Narrative in Games," in *Proceedings of the 30th ACM Conference on Hypertext and Social Media*, HT '19, Hof, Germany: ACM, 2019, pages 77–86, ISBN: 978-1-4503-6885-8. DOI: 10.1145/3342220.3343655 (cited on pages xx, 77, 260).

[88] Daniel Green, Charlie Hargood, and Fred Charles, "Use of Tools: UX Principles for Interactive Narrative Authoring Tools," *Journal on Computing and Cultural Heritage*, volume 14, number 3, 2021. DOI: 10.1145/3458769 (cited on pages xx, 124, 260).

[89] Daniel Green, Charlie Hargood, Fred Charles, and Alex Jones, "Novella: A Proposition for Game-Based Storytelling," presented at the Narrative and Hypertext Workshop, 2018 (cited on pages xx, 77, 82, 259).

[90] Saul Greenberg and Bill Buxton, "Usability Evaluation Considered Harmful (Some of the Time)," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, Florence, Italy: ACM, 2008, pages 111–120, ISBN: 978-1-60558-011-1. DOI: 10.1145/1357054.1357074 (cited on pages 152, 154).

[91] Algirdas Greimas, *Sémantique Structurale: Recherche de Méthode*. Paris: Larousse, 1966, ISBN: 978-2-03-070314-4 (cited on pages 9, 11).

[92] Algirdas Julien Greimas, *On Meaning: Selected Writings in Semiotic Theory*, New Edition, translated by P. J. Perron and F. H. Collins. Minneapolis: University of Minnesota Press, 1987, ISBN: 978-0-8166-1519-3 (cited on page 11).

[93] Andrea Guarneri, Laura A. Ripamonti, Francesco Tissoni, Marco Trubian, Dario Maggiorini, and Davide Gadia, "GHOST: A GHOst STory-writer," in *Proceedings of the 12th Biannual Conference on Italian SIGCHI Chapter*, CHItaly '17, New York, NY, USA: ACM, 2017, 24:1–24:9, ISBN: 978-1-4503-5237-6. DOI: 10.1145/3125571.3125580 (cited on pages xvii, 54).

[94] Erika Hall, *Just Enough Research*. New York, NY, USA: A Book Apart, 2013, ISBN: 978-1-937557-11-9 (cited on page 202).

[95] Stephen Halliwell, *Aristotle's Poetics*, New Edition. Bristol Classical Press, 1998, ISBN: 978-0-7156-2858-4 (cited on page 7).

[96] Matthew J. Hamm, *Wireframing Essentials*. Packt Publishing Ltd, 2014, ISBN: 978-1-84969-855-9 (cited on page 156).

[97] B. L. Hanson, "Human Factors and Behavioral Science: A Brief History of Applied Behavioral Science at Bell Laboratories," *The Bell System Technical Journal*, volume 62, number 6, pages 1571–1590, 1983, ISSN: 0005-8580. DOI: 10.1002/j.1538-7305.1983.tb03499.x (cited on page 63).

[98] Charlie Hargood, Mark J. Weal, and David E. Millard, "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, 2018, pages 128–135, ISBN: 978-1-4503-5427-1. DOI: 10.1145/3209542.3209559 (cited on pages xvii, 33, 51, 54, 55, 126, 130).

[99] Marc Hassenzahl and Noam Tractinsky, "User Experience — A Research Agenda," *Behaviour & Information Technology*, volume 25, number 2, pages 91–97, 2006, ISSN: 0144-929X. DOI: 10.1080/01449290500330331 (cited on page 63).

[100] Tobias Heussner, Toiya Kristen Finley, Jennifer Brandes Hepler, and Ann Lemay, *The Game Narrative Toolbox*. CRC Press, 2015, ISBN: 978-1-317-66163-4 (cited on pages 159, 160).

[101] W. E. Hick, "On the Rate of Gain of Information," *Quarterly Journal of Experimental Psychology*, volume 4, number 1, pages 11–26, 1952, ISSN: 0033-555X. DOI: 10.1080/17470215208416600 (cited on page 68).

[102] Ray Hyman, "Stimulus Information as a Determinant of Reaction Time," *Journal of Experimental Psychology*, volume 45, number 3, pages 188–196, 1953, ISSN: 0022-1015. DOI: 10.1037/h0056940 (cited on page 68).

[103] Henry Jenkins, "Game Design as Narrative Architecture," *First Person: New Media as Story, Performance, and Game*, volume 44, Noah Wardrip-Fruin and Pat Harrigan, Eds., 2004 (cited on pages 30, 35, 130).

[104] Alex Jones, Brad Gyori, Charlie Hargood, Fred Charles, and Daniel Green, "Shelley's Heart: Experiences in Designing a Multi-Reader Locative Narrative," presented at the Narrative and Hypertext Workshop, 2018 (cited on page xx).

[105] Jesper Juul, "A Clash between Game and Narrative," M.S. thesis, Institute of Nordic Language and Literature, University of Copenhagen, 1999 (cited on page 30).

[106] Jesper Juul, "Games Telling Stories? A Brief Note on Games and Narratives," *Game Studies*, volume 1, pages 1–12, 2001 (cited on page 30).

[107] Noriaki Kano, Nobuhiko Seraku, Fumio Takahashi, and Shin-ichi Tsuji, "Attractive Quality and Must-Be Quality," *Journal of The Japanese Society for Quality Control*, volume 14, number 2, pages 147–156, 1984. DOI: 10.20684/quality.14.2_147 (cited on page 158).

[108] Mubbasir Kapadia, Seth Frey, Alexander Shoulson, Robert W. Sumner, and Markus Gross, "CANVAS: Computer-Assisted Narrative Animation Synthesis," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '16, Zurich, Switzerland: Eurographics Association, 2016, pages 199–209, ISBN: 978-3-905674-61-3. [Online]. Available: http://dl.acm.org/citation.cfm?id=2982818.2982846 (cited on page 93).

[109] SeokKyoo Kim, SungHyun Moon, SangYong Han, and Juno Chan, "Programming the Story: Interactive Storytelling System," *Informatica*, volume 35, number 2, 2011, ISSN: 1854-3871. [Online]. Available: http://www.informatica.si/index.php/informatica/article/view/346 (cited on pages xvii, 54).

[110] Hartmut Koenitz, "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, Berlin, Heidelberg: Springer, 2011, pages 79–84, ISBN: 978-3-642-25288-4. DOI: 10.1007/978-3-642-25289-1_9 (cited on pages xvii, 54, 126).

[111] Hartmut Koenitz and Kun-Ju Chen, "Genres, Structures and Strategies in Interactive Digital Narratives — Analyzing a Body of Works Created in ASAPS," in *Interactive Storytelling*, Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2012, pages 84–95, ISBN: 978-3-642-34851-8. DOI: 10.1007/978-3-642-34851-8_8 (cited on pages 143, 147, 194).

[112] Kurt Koffka, *Principles of Gestalt Psychology*. New York: Harcourt, Brace, 1935 (cited on page 67).

[113] Wolfgang Köhler, "Gestalt Psychology," *Psychologische Forschung*, volume 31, number 1, pages XVIII–XXX, 1967, ISSN: 1430-2772. DOI: 10.1007/BF00422382 (cited on page 67).

[114] Hannu Korhonen, "Evaluating Playability of Mobile Games with the Expert Review Method," Ph.D. dissertation, Tampere University, 2016 (cited on pages 123, 152).

[115] Raph Koster, *Theory of Fun for Game Design*, 2nd edition. O'Reilly Media, 2013, ISBN: 978-1-4493-6321-5 (cited on page 39).

[116] Michael Kriegel and Ruth Aylett, "An Authoring Tool for an Emergent Narrative Storytelling System," in *AAAI Fall, Symposium on Intelligent Narrative Technologies*, 2007 (cited on pages xvii, 31, 54).

[117] Jon A. Krosnick, "Survey Research," *Annual Review of Psychology*, volume 50, number 1, pages 537–567, 1999. DOI: 10.1146/annurev.psych.50.1.537 (cited on page 201).

[118] Jon A. Krosnick and Leandre R. Fabrigar, "Designing Rating Scales for Effective Measurement in Surveys," *Survey Measurement and Process Quality*, Wiley Series in Probability and Statistics, pages 141–164, 1997, ISSN: 9781118490013. DOI: 10.1002/9781118490013.ch6 (cited on pages 74, 202).

[119] Masaaki Kurosu and Kaori Kashimura, "Apparent Usability vs. Inherent Usability: Experimental Analysis on the Determinants of the Apparent Usability," in *Conference Companion on Human Factors in Computing Systems*, CHI '95, New York, NY, USA: ACM, 1995, pages 292–293, ISBN: 978-0-89791-755-1. DOI: 10.1145/223355.223680 (cited on page 65).

[120] Ben Kybartas and Rafael Bidarra, "A Survey on Story Generation Techniques for Authoring Computational Narratives," *IEEE Transactions on Computational Intelligence and AI in Games*, volume 9, number 3, pages 239–253, 2017, ISSN: 1943-068X. DOI: 10.1109/TCIAIG.2016.2546063 (cited on pages 25, 26).

[121] George Lakoff, *Structural Complexity in Fairy Tales*. UC Berkeley, 1972 (cited on page 27).

[122] Carine Lallemand, Guillaume Gronier, and Vincent Koenig, "User Experience: A Concept Without Consensus? Exploring Practitioners' Perspectives Through an International Survey," *Computers in Human Behavior*, volume 43, pages 35–48, 2015, ISSN: 0747-5632. DOI: 10.1016/j.chb.2014.10.048 (cited on page 63).

[123] Petri Lankoski and Staffan Björk, "Formal Analysis of Gameplay," in *Game Research Methods*, Pittsburgh, USA: ETC Press, 2015, pages 23–35, ISBN: 978-1-312-88473-1 (cited on page 87).

[124] Effie Law, Virpi Roto, Arnold P.O.S. Vermeeren, Joke Kort, and Marc Hassenzahl, "Towards a Shared Definition of User Experience," in *CHI '08 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '08, New York, NY, USA: ACM, 2008, pages 2395–2398, ISBN: 978-1-60558-012-8. DOI: 10.1145/1358628.1358693 (cited on page 63).

[125] Effie Lai-Chong Law, Virpi Roto, Marc Hassenzahl, Arnold P.O.S. Vermeeren, and Joke Kort, "Understanding, Scoping and Defining User Experience: A Survey Approach," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, Boston, MA, USA: ACM, 2009, pages 719–728, ISBN: 978-1-60558-246-7. DOI: 10.1145/1518701.1518813 (cited on page 63).

[126] Sébastien Lê, Julie Josse, and François Husson, "FactoMineR: An R Package for Multivariate Analysis," *Journal of Statistical Software*, volume 25, number 1, pages 1–18, 2008 (cited on page 127).

[127] Clayton Lewis, "Using the 'Thinking-Aloud' Method in Cognitive Interface Design," IBM Research, Tech. Rep. RC9265, 1982 (cited on page 157).

[128] James R. Lewis, "Usability: Lessons Learned ... and Yet to Be Learned," *International Journal of Human-Computer Interaction*, volume 30, number 9, pages 663–684, 2014, ISSN: 1044-7318. DOI: 10.1080/10447318.2014.930311 (cited on page 73).

[129] James R. Lewis and Jeff Sauro, "Can I Leave This One Out? The Effect of Dropping an Item From the SUS," *Journal of Usability Studies*, volume 13, number 1, pages 38–46, 2017, ISSN: 1931-3357 (cited on page 73).

[130] James R. Lewis, Brian S. Utesch, and Deborah E. Maher, "Measuring Perceived Usability: The SUS, UMUX-LITE, and AltUsability," *International Journal of Human-Computer Interaction*, volume 31, number 8, pages 496–505, 2015, ISSN: 1044-7318. DOI: 10.1080/10447318.2015.1064654 (cited on page 73).

[131] James R. Lewis, Brian S. Utesch, and Deborah E. Maher, "UMUX-LITE: When There's No Time for the SUS," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, New York, NY, USA: ACM, 2013, pages 2099–2102, ISBN: 978-1-4503-1899-0. DOI: 10.1145/2470654.2481287 (cited on page 73).

[132] David Lightbown, *Designing the User Experience of Game Development Tools*. Boca Raton: Routledge, 2015, ISBN: 978-1-4822-4019-1 (cited on pages 75, 157).

[133] ZhiQiang Liu and KaMing Leung, "Script Visualization (ScriptVis): A Smart System That Makes Writing Fun," in *Proceedings of the 2003 International Conference on Machine Learning and Cybernetics*, ICMLC '03, volume 5, Xi'an, China: IEEE, 2003, pages 2990–2995. DOI: 10.1109/ICMLC.2003.1260088 (cited on pages xvii, 28).

[134] Sandy Louchart, Ruth Aylett, Michael Kriegel, Joao Dias, Rui Figueiredo, and Ana Paiva, "Authoring Emergent Narrative-Based Games," *Journal of Game Development*, volume 3, number 1, pages 19–37, 2007, ISSN: 1543-9399 (cited on page 31).

[135] Todd Lubart, "How Can Computers Be Partners in the Creative Process: Classification and Commentary on the Special Issue," *International Journal of Human-Computer Studies*, volume 63, number 4, pages 365–369, 2005, ISSN: 1071-5819. DOI: 10.1016/j.ijhcs.2005.04.002 (cited on page 25).

[136] Kirstin Lyon and Peter J. Nürnberg, "Applying Information Visualisation Techniques to Spatial Hypertext Tools," in *Metainformatics*, Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2005, pages 85–93, ISBN: 978-3-540-32105-7. DOI: 10.1007/11518358_7 (cited on pages 57, 67, 225).

[137] Stephanie Marsh, *User Research: A Practical Guide to Designing Better Products and Services*. London: Kogan Page Ltd., 2018, ISBN: 978-0-7494-8104-9 (cited on pages 74–76, 202).

[138] Catherine C. Marshall and Frank M. Shipman, "Searching for the Missing Link: Discovering Implicit Structure in Spatial Hypertext," in *Proceedings of the 5th ACM Conference on Hypertext*, HT '93, New York, NY, USA: ACM, 1993, pages 217–230, ISBN: 978-0-89791-624-0. DOI: 10.1145/168750.168826 (cited on pages 57, 67, 191, 225).

[139] Chris Martens and Owais Iqbal, "Villanelle: An Authoring Tool for Autonomous Characters in Interactive Fiction," in *Interactive Storytelling*, Lecture Notes in Computer Science, Cham: Springer, 2019, pages 290–303, ISBN: 978-3-030-33894-7. DOI: 10.1007/978-3-030-33894-7_29 (cited on pages xviii, 54, 155).

[140] Abraham Maslow, "A Theory of Human Motivation," *Psychological Review*, volume 50, number 4, pages 370–396, 1943, ISSN: 0033-295X. DOI: 10.1037/h0054346 (cited on page 64).

[141] Stacey Mason, "On Games and Links: Extending the Vocabulary of Agency and Immersion in Interactive Narratives," in *Interactive Storytelling*, Lecture Notes in Computer Science, Cham: Springer, 2013, pages 25–34, ISBN: 978-3-319-02756-2 (cited on pages 42, 130).

[142] Michael Mateas and Andrew Stern, "Procedural Authorship: A Case Study of the Interactive Drama Façade," in *Proceedings of the 6th Digital Arts and Culture*, Copenhagen, Denmark, 2005 (cited on page 28).

[143] Donald L. McCracken and Robert M. Akscyn, "Experience With the ZOG Human-Computer Interface System," *International Journal of Man-Machine Studies*, volume 21, number 4, pages 293–310, 1984, ISSN: 0020-7373. DOI: 10.1016/S0020-7373(84)80050-4 (cited on page 15).

[144] Michael Medlock, "The Rapid Iterative Test and Evaluation Method (RITE)," in *Games User Research*, Oxford University Press, 2018, ISBN: 978-0-19-879484-4 (cited on pages 155, 157, 168, 170).

[145] Michael Medlock, Dennis Wixon, Mark Terrano, Ramon Romero, and Bill Fulton, "Using the RITE Method to Improve Products: A Definition and a Case Study," *Usability Professionals Association*, volume 51, 2002 (cited on pages 155, 157, 168).

[146] James Meehan, "TALE-SPIN, an Interactive Program That Writes Stories," in *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, IJCAI '77, volume 1, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1977, pages 91–98 (cited on pages xviii, 28).

[147] Wolfgang Metzger, *Laws of Seeing*. Cambridge, MA, US: MIT Press, 2006, ISBN: 978-0-262-13467-5 (cited on pages 67, 188).

[148] Norman Meyrowitz, "Intermedia: The Architecture and Construction of an Object-Oriented Hypemedia System and Applications Framework," in *Conference Proceedings on Object-Oriented Programming Systems, Languages and Applications*, OOPSLA '86, New York, NY, USA: ACM, 1986, pages 186–201, ISBN: 978-0-89791-204-4. DOI: 10.1145/28697.28716 (cited on pages xvii, 16).

[149] David E Millard, Charlie Hargood, Yvonne Howard, and Heather Packer, "The StoryPlaces Authoring Tool: Pattern Centric Authoring," presented at the Authoring for Interactive Storytelling Workshop, 2017 (cited on pages xvii, 2, 33, 51, 54, 55, 65, 123, 155).

[150] David E. Millard and Charlie Hargood, "Tiree Tales: A Co-operative Inquiry Into the Poetics of Location-Based Narrative," in *Proceedings of the 28th ACM Conference on Hypertext and Social Media*, HT '17, New York, NY, USA: ACM, 2017, pages 15–24, ISBN: 978-1-4503-4708-2. DOI: 10.1145/3078714.3078716 (cited on page 123).

[151] George A. Miller, "The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information," *Psychological Review*, volume 63, number 2, pages 81–97, 1956, ISSN: 0033-295X. DOI: 10.1037/h0043158 (cited on page 70).

[152] Alexander G. Mirnig, Alexander Meschtscherjakov, Daniela Wurhofer, Thomas Meneweger, and Manfred Tscheligi, "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, New York, NY, USA: ACM, 2015, pages 437–450, ISBN: 978-1-4503-3146-3. DOI: 10.1145/2702613.2732511 (cited on page 63).

[153] Alex Mitchell and Kevin McGee, "The HypeDyn Hypertext Fiction Authoring Tool," in *Proceedings of the 2nd Workshop on Narrative and Hypertext*, NHT '12, Milwaukee, Wisconsin, USA: ACM, 2012, pages 19–22, ISBN: 978-1-4503-1408-4. DOI: 10.1145/2310076.2310081 (cited on pages xvii, 54).

[154] Afshin Mobramaein and Jim Whitehead, "A Methodology for Designing Natural Language Interfaces for Procedural Content Generation," in *Proceedings of the 14th International Conference on the Foundations of Digital Games*, FDG '19, New York, NY, USA: ACM, 2019, 102:1–102:9, ISBN: 978-1-4503-7217-6. DOI: 10.1145/3337722.3341860 (cited on page 109).

[155] Nick Montfort, *Twisty Little Passages: An Approach to Interactive Fiction*. MIT Press, 2005, ISBN: 978-0-262-63318-5 (cited on page 22).

[156] Michael J. Muller and Sarah Kuhn, "Participatory Design," *Communications of the ACM*, volume 36, number 6, pages 24–28, 1993, ISSN: 0001-0782. DOI: 10.1145/153571.255960 (cited on pages 64, 162).

[157] Janet Murray, "The Last Word on Ludology v Narratology in Game Studies," in *DiGRA*, delivered as a preface to a keynote talk., Vancouver, Canada, 2005. [Online]. Available: https://inventingthemedium.com/2013/06/28/the-last-word-on-ludology-v-narratology-2005 (cited on page 30).

[158] Nicholas Negroponte, *Soft Architecture Machines*. MIT Press, 1975, ISBN: 978-0-262-14018-8 (cited on page 25).

[159] Graham Nelson, *The Inform Designer's Manual*, 4th edition. Dan Sanderson, 2006 (cited on pages 45, 50).

[160] T. H. Nelson, "Complex Information Processing: A File Structure for the Complex, the Changing and the Indeterminate," in *Proceedings of the 1965 20th National Conference*, ACM '65, New York, NY, USA: ACM, 1965, pages 84–100. DOI: 10.1145/800197.806036 (cited on page 13).

[161] Ted Nelson, *Literary Machines*. Mindful Press, 1982, ISBN: 978-0-89347-062-3 (cited on pages xviii, 14, 344).

[162] Ted Nelson, *Selected Papers*. 1977. [Online]. Available: https://archive.org/details/SelectedPapers1977 (cited on page 14).

[163] Jakob Nielsen, "Enhancing the Explanatory Power of Usability Heuristics," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '94, New York, NY, USA: ACM, 1994, pages 152–158, ISBN: 978-0-89791-650-9. DOI: 10.1145/191666.191729 (cited on pages 72, 124, 151).

[164] Jakob Nielsen, *Multimedia and Hypertext: The Internet and Beyond*, 1st edition. AP Professional, 1995, ISBN: 978-0-12-518408-3 (cited on page 15).

[165] Jakob Nielsen, "The Art of Navigating Through Hypertext," *Communications of the ACM*, volume 33, number 3, pages 296–310, 1990, ISSN: 0001-0782. DOI: 10.1145/77481.77483 (cited on page 14).

[166] Jakob Nielsen, *Usability Engineering*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994, ISBN: 978-0-08-052029-2 (cited on pages 73–75).

[167] Jakob Nielsen and Thomas K. Landauer, "A Mathematical Model of the Finding of Usability Problems," in *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, CHI '93, New York, NY, USA: ACM, 1993, pages 206–213, ISBN: 978-0-89791-575-5. DOI: 10.1145/169059.169166 (cited on page 173).

[168] Anthony J. Niesz and Norman N. Holland, "Interactive Fiction," *Critical Inquiry*, volume 11, number 1, pages 110–129, 1984, ISSN: 0093-1896. DOI: 10.1086/448277 (cited on page 21).

[169] Don Norman, Jim Miller, and Austin Henderson, "What You See, Some of What's in the Future, and How We Go About Doing It: HI at Apple Computer," in *Conference Companion on Human Factors in Computing Systems*, CHI '95, Denver, Colorado, USA: ACM, 1995, page 155, ISBN: 978-0-89791-755-1. DOI: 10.1145/223355.223477 (cited on page 63).

[170] Donald Norman, "Design as Practiced," in *Bringing Design to Software*, Terry Winograd, Ed., New York, NY, USA: ACM, 1996, pages 233–251, ISBN: 978-0-201-85491-6 (cited on page 63).

[171] Steven Poulakos, Mubbasir Kapadia, Guido M. Maiga, Fabio Zünd, Markus Gross, and Robert W. Sumner, "Evaluating Accessible Graphical Interfaces for Building Story Worlds," in *Interactive Storytelling*, Lecture Notes in Computer Science, Cham: Springer, 2016, pages 184–196, ISBN: 978-3-319-48279-8. DOI: 10.1007/978-3-319-48279-8_17 (cited on page 93).

[172] Steven Poulakos, Mubbasir Kapadia, Andrea Schüpfer, Fabio Zünd, Robert W. Sumner, and Markus Gross, "Towards an Accessible Interface for Story World Building," in *Eleventh Artificial Intelligence and Interactive Digital Entertainment Conference*, 2015. [Online]. Available: https://www.aaai.org/ocs/index.php/AIIDE/AIIDE15/paper/view/11583 (cited on pages xvii, 54).

[173] Peirce Edition Project, *The Essential Peirce: Selected Philosophical Writings*. Indiana University Press, 1998, volume 2 (cited on page 36).

[174] Vladimir Propp, *Morphology of the Folktale: Second Edition*. University of Texas Press, 2010, ISBN: 978-0-292-79249-4 (cited on pages 10, 11, 27, 81).

[175] Laia Pujol-Tost, "Did We Just Travel to the Past? Building and Evaluating With Cultural Presence Different Modes of VR-Mediated Experiences in Virtual Archaeology," *Journal on Computing and Cultural Heritage*, volume 12, number 1, 2:1–2:20, 2019, ISSN: 1556-4673. DOI: 10.1145/3230678 (cited on page 123).

[176] George E. Raptis, Christos Fidas, and Nikolaos Avouris, "Do Game Designers' Decisions Related to Visual Activities Affect Knowledge Acquisition in Cultural Heritage Games? An Evaluation From a Human Cognitive Processing Perspective," *Journal on Computing and Cultural Heritage*, volume 12, number 1, 4:1–4:25, 2019, ISSN: 1556-4673. DOI: 10.1145/3292057 (cited on page 123).

[177] Paul Ricoeur, "Narrative Time," *Critical Inquiry*, volume 7, number 1, pages 169–190, 1980, ISSN: 0093-1896. [Online]. Available: https://www.jstor.org/stable/1343181 (cited on page 9).

[178] G. Robertson, D. Mccracken, and A. Newell, "The ZOG Approach to Man-Machine Communication," *International Journal of Human-Computer Studies*, volume 51, number 2, pages 279–306, 1999, ISSN: 1071-5819. DOI: 10.1006/ijhc.1980.0310 (cited on pages xviii, 14).

[179] Robert W. Root and Steve Draper, "Questionnaires as a Software Evaluation Tool," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '83, New York, NY, USA: ACM, 1983, pages 83–87, ISBN: 978-0-89791-121-4. DOI: 10.1145/800045.801586 (cited on page 74).

[180] Lee Ross, David Greene, and Pamela House, "The "False Consensus Effect": An Egocentric Bias in Social Perception and Attribution Processes," *Journal of Experimental Social Psychology*, volume 13, number 3, pages 279–301, 1977, ISSN: 0022-1031. DOI: 10.1016/0022-1031(77)90049-X (cited on page 347).

[181] Virpi Roto, Effie Law, Arnold Vermeeren, and Jettie Hoonhout, "User Experience White Paper — Bringing Clarity to the Concept of User Experience," Dagstuhl Seminar on Demarcating User Experience, Tech. Rep., 2011. [Online]. Available: http://www.allaboutux.org/uxwhitepaper (cited on page 63).

[182] Jeffrey Rubin and Dana Chisnell, *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests*, 2nd edition. Wiley Publishing, 2008, ISBN: 978-0-470-18548-3 (cited on page 74).

[183] James Ryan, "Grimes' Fairy Tales: A 1960s Story Generator," in *Interactive Storytelling*, Lecture Notes in Computer Science, Cham: Springer, 2017, pages 89–103, ISBN: 978-3-319-71027-3. DOI: 10.1007/978-3-319-71027-3_8 (cited on page 27).

[184] James Owen Ryan, Michael Mateas, and Noah Wardrip-Fruin, "Open Design Challenges for Interactive Emergent Narrative," in *Interactive Storytelling*, Lecture Notes in Computer Science, Cham: Springer, 2015, pages 14–26, ISBN: 978-3-319-27036-4. DOI: 10.1007/978-3-319-27036-4_2 (cited on page 31).

[185] Johnny Saldaña, *The Coding Manual for Qualitative Researchers*, 3rd edition. SAGE Publishing, 2015, ISBN: 978-1-4739-0248-0 (cited on pages 142, 211).

[186] Katie Salen and Eric Zimmerman, *Rules of Play: Game Design Fundamentals*. MIT Press, 2004, ISBN: 978-0-262-24045-1 (cited on pages 31, 130).

[187] Elizabeth Sanders, "Converging Perspectives: Product Development Research for the 1990s," *Design Management Journal*, volume 3, number 4, pages 49–54, 1992, ISSN: 1948-7169. DOI: 10.1111/j.1948-7169.1992.tb00604.x (cited on page 64).

[188] Willem Saris, Melanie Revilla, Jon A. Krosnick, and Eric M. Shaeffer, "Comparing Questions with Agree/Disagree Response Options to Questions with Item-Specific Response Options," *Survey Research Methods*, volume 4, number 1, pages 61–79, 2010, ISSN: 1864-3361. DOI: 10.18148/srm/2010.v4i1.2682 (cited on pages 74, 202).

[189] Yotam Shibolet, Noam Knoller, and Hartmut Koenitz, "A Framework for Classifying and Describing Authoring Tools for Interactive Digital Narrative," in *Interactive Storytelling*, Lecture Notes in Computer Science, Cham: Springer, 2018, pages 523–533, ISBN: 978-3-030-04028-4. DOI: 10.1007/978-3-030-04028-4_61 (cited on pages 43, 123).

[190] Frank M. Shipman and Catherine C. Marshall, "Spatial Hypertext: An Alternative to Navigational and Semantic Links," *ACM Computing Surveys*, volume 31, number 4es, pages 14–18, 1999, ISSN: 0360-0300. DOI: 10.1145/345966.346001 (cited on pages 57, 67, 143, 147–149, 188, 191, 192, 225).

[191] Viktor Shklovsky, "Art, as Device," *Poetics Today*, volume 36, number 3, pages 151–174, 2015, ISSN: 0333-5372. DOI: 10.1215/03335372-3160709 (cited on page 8).

[192] Viktor Shklovsky, *Theory of Prose*, translated by Benjamin Sher, Russian Literature Series. Dalkey Archive Press, 1990, ISBN: 978-0-916583-64-4. [Online]. Available: http://www.dalkeyarchive.com/product/theory-of-prose (cited on page 8).

[193] Ben Shneiderman, "User Interface Design for the Hyperties Electronic Encyclopedia (Panel Session)," in *Proceedings of the ACM conference on Hypertext*, HT '87, New York, NY, USA: ACM, 1987, pages 189–194, ISBN: 978-0-89791-340-9. DOI: 10.1145/317426.317441 (cited on pages xvii, 15).

[194] Ben Shneiderman, Catherine Plaisant, Maxine Cohen, Steven Jacobs, Niklas Elmqvist, and Nicholas Diakopoulos, *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, 6th edition. Pearson, 2016, ISBN: 978-0-13-438038-4 (cited on page 72).

[195] Miguel Sicart, "Defining Game Mechanics," *Game Studies*, volume 8, number 2, 2008, ISSN: 1604-7982 (cited on page 37).

[196] James Skorupski and Michael Mateas, "Novice-Friendly Authoring of Plan-Based Interactive Storyboards," in *Proceedings of the Sixth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, AIIDE '10, Stanford, California, USA: AAAI Press, 2010, pages 174–179 (cited on pages xvii, 28).

[197] Mikael B. Skov and Jan Stage, "Designing Interactive Narrative Systems: Is Object-Orientation Useful?" *Computers & Graphics*, volume 26, number 1, pages 57–66, 2002, ISSN: 0097-8493. DOI: 10.1016/S0097-8493(01)00178-9 (cited on page 13).

[198] Nils Sørensen and Mette Pødenphant, "Narratification: Unifying Narrative and Gameplay," SIDeR '13, 2013. [Online]. Available: https://sider.au.dk/fileadmin/sider/0125-paper.pdf (cited on pages 30, 83, 85, 86).

[199] Ulrike Spierling, "Adding Aspects of "Implicit Creation" to the Authoring Process in Interactive Storytelling," in *Proceedings of the 4th international conference on Virtual storytelling: using virtual reality technologies for storytelling*, Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2007, pages 13–25, ISBN: 978-3-540-77039-8. DOI: 10.1007/978-3-540-77039-8_2 (cited on page 65).

[200] Ulrike Spierling and Ido Iurgel, "Pre-Conference Demo Workshop "Little Red Cap": The Authoring Process in Interactive Storytelling," in *Technologies for Interactive Digital Storytelling and Entertainment*, Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2006, pages 193–194, ISBN: 978-3-540-49935-0. DOI: 10.1007/11944577_20 (cited on page 65).

[201] Ulrike Spierling and Nicolas Szilas, "Authoring Issues beyond Tools," in *Interactive Storytelling*, Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2009, pages 50–61, ISBN: 978-3-642-10643-9. DOI: 10.1007/978-3-642-10643-9_9 (cited on pages 2, 65, 123, 155).

[202] Ulrike Spierling, Sebastian A. Weiß, and Wolfgang Müller, "Towards Accessible Authoring Tools for Interactive Storytelling," in *Technologies for Interactive Digital Storytelling and Entertainment*, Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2006, pages 169–180, ISBN: 978-3-540-49935-0. DOI: 10.1007/11944577_17 (cited on page 66).

[203] Laurence Sterne, *The Life and Opinions of Tristram Shandy, Gentleman*. Penguin Classics, 2003, ISBN: 978-0-14-143977-8 (cited on page 8).

[204] Ivo Swartjes, Edze Kruizinga, and Mariët Theune, "Let's Pretend I Had a Sword," in *Interactive Storytelling*, Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2008, pages 264–267, ISBN: 978-3-540-89454-4. DOI: 10.1007/978-3-540-89454-4_33 (cited on page 29).

[205] Ivo Swartjes and Mariët Theune, "The Virtual Storyteller: Story Generation by Simulation," in *Proceedings the 20th Belgian-Dutch Artificial Intelligence Conference*, BNAIC '08, Enschede: University of Twente, 2008, pages 257–265 (cited on pages xviii, 29).

[206] Nicolas Szilas, "A Computational Model of an Intelligent Narrator For Interactive Narratives," *Applied Artificial Intelligence*, volume 21, number 8, pages 753–801, 2007, ISSN: 0883-9514. DOI: 10.1080/08839510701526574 (cited on pages 28, 66).

[207] Nicolas Szilas, Olivier Marty, and Jean-Hugues Réty, "Authoring Highly Generative Interactive Drama," in *Proceedings of the 2nd international conference on Virtual storytelling: using virtual reality technologies for storytelling*, Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2003, pages 37–46, ISBN: 978-3-540-40014-1. DOI: 10.1007/978-3-540-40014-1_5 (cited on pages 28, 66).

[208] Mariët Theune, Sander Rensen, Rieks op den Akker, Dirk Heylen, and Anton Nijholt, "Emotional Characters for Automatic Plot Creation," in *Technologies for Interactive Digital Storytelling and Entertainment*, Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2004, pages 95–100, ISBN: 978-3-540-27797-2. DOI: 10.1007/978-3-540-27797-2_13 (cited on pages xviii, 29).

[209] David Thue, Vadim Bulitko, Marcia Spetch, and Eric Wasylishen, "Interactive Storytelling: A Player Modelling Approach," in *Proceedings of the Third AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, AIIDE '07, Stanford, California: AAAI Press, 2007, pages 43–48. [Online]. Available: http://dl.acm.org/citation.cfm?id=3022508.3022517 (cited on pages xvii, 54).

[210] *Time Machine (novel series)*. Bantam Books, 1984 (cited on page 3).

[211] Tzvetan Todorov, *Grammaire du Décaméron*. The Hague, Paris: Mouton, 1969 (cited on page 7).

[212] David Travis and Philip Hodgson, *Think Like a UX Researcher: How to Observe Users, Influence Design, and Shape Business Strategy*. CRC Press, 2019, ISBN: 978-1-138-36529-2 (cited on page 75).

[213] Marian F Ursu, Jonathan J Cook, Vilmos Zsombori, and Ian Kegel, "A Genre-Independent Approach to Producing Interactive Screen Media Narratives," in *AAAI Fall Symposium Series*, 2007. [Online]. Available: https://www.aaai.org/Library/Symposia/Fall/2007/fs07-05-029.php (cited on pages xvii, 54).

[214] Hans-Jörg Uther, *The Types of International Folktales: A Classification and Bibliography. Animal Tales, Tales of Magic, Religious Tales, and Realistic Tales, With an Introduction*. Academia Scientiarum Fennica, 2004, ISBN: 978-951-41-0955-3 (cited on page 10).

[215] Aaron Walter, *Designing for Emotion*. New York, NY, USA: A Book Apart, 2011, ISBN: 978-1-937557-00-3 (cited on page 65).

[216] Joseph Weizenbaum, "ELIZA — A Computer Program for the Study of Natural Language Communication Between Man and Machine," *Communications of the ACM*, volume 9, number 1, pages 36–45, 1966, ISSN: 0001-0782. DOI: 10.1145/365153.365168 (cited on pages xvii, 23).

[217] Max Wertheimer, "Laws of Organization in Perceptual Forms," in *A Source Book of Gestalt Psychology*, Willis Ellis, Ed., London: Routledge & Kegan Paul, 1938, pages 71–88 (cited on page 67).

[218] John Whiteside, John Bennett, and Karen Holtzblatt, "Usability Engineering: Our Experience and Evolution," in *Handbook of Human-Computer Interaction*, Martin Helander, Ed., Amsterdam: North Holland, Jan. 1, 1988, pages 791–817, ISBN: 978-0-444-70536-5 (cited on page 75).

[219] Stephanie Wical, "Memory Machines: The Evolution of Hypertext," *Online Information Review*, volume 38, number 5, pages 701–702, 2014, ISSN: 1468-4527. DOI: 10.1108/OIR-06-2014-0146 (cited on pages 16, 17).

[220] Terry Winograd, "Procedures as a Representation for Data in a Computer Program for Understanding Natural Language," Ph.D. dissertation, Massachusetts Institute of Technology, 1971 (cited on pages xvii, 23).

[221] José P. Zagal, Michael Mateas, Clara Fernández-Vara, Brian Hochhalter, and Nolan Lichti, "Towards an Ontological Language for Game Analysis," in *Proceedings of the 2005 DiGRA International Conference: Changing Views: Worlds in Play*, volume 3, 2005 (cited on page 87).

[222] Fabio Zünd, Steven Poulakos, Mubbasir Kapadia, and Robert W. Sumner, "Story Version Control and Graphical Visualization for Collaborative Story Authoring," in *Proceedings of the 14th European Conference on Visual Media Production*, CVMP '17, New York, NY, USA: ACM, 2017, 10:1–10:10, ISBN: 978-1-4503-5329-8. DOI: 10.1145/3150165.3150175 (cited on pages xvii, 54).

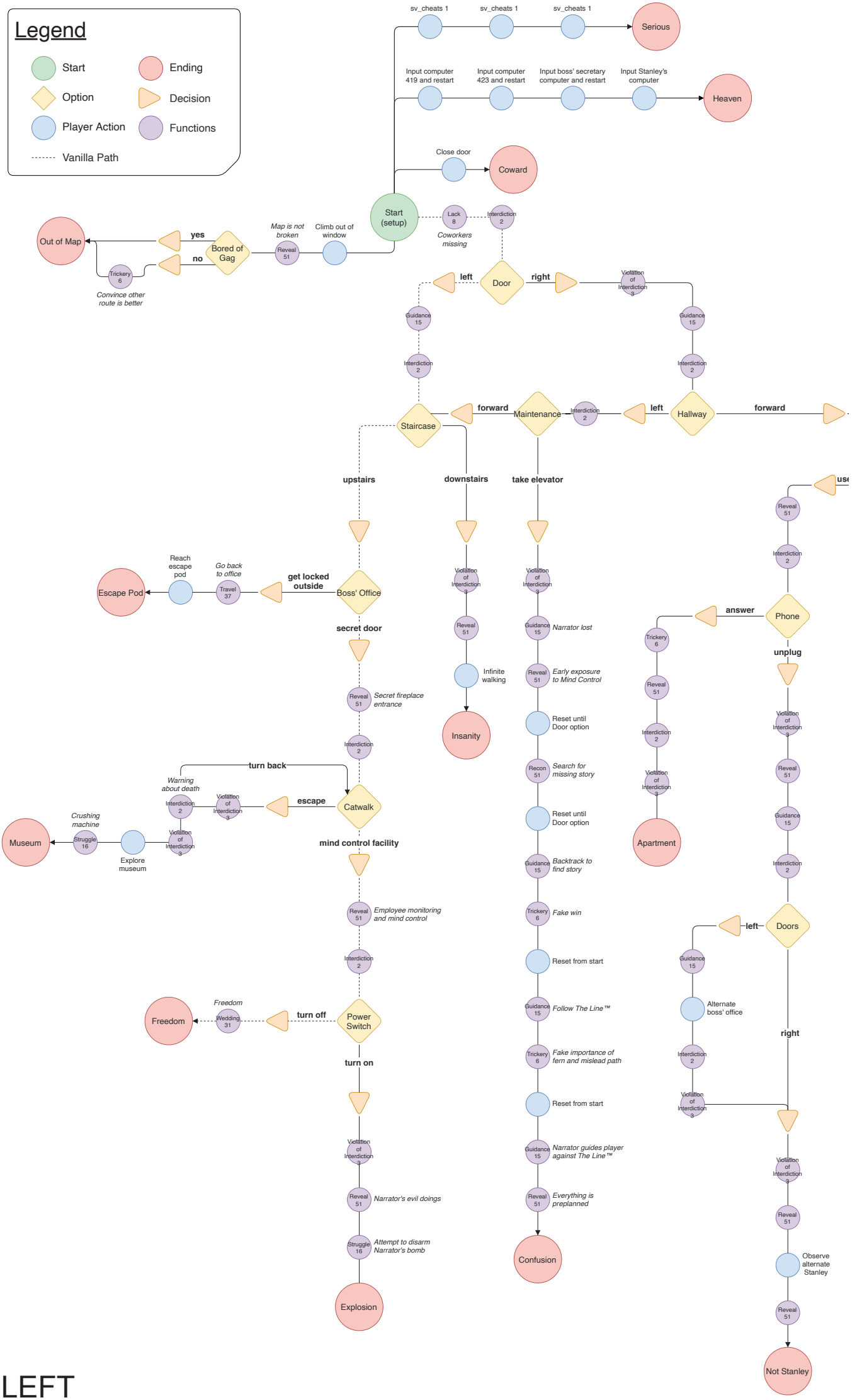
APPENDIX A

Modeling Video Game Narrative for Authorship

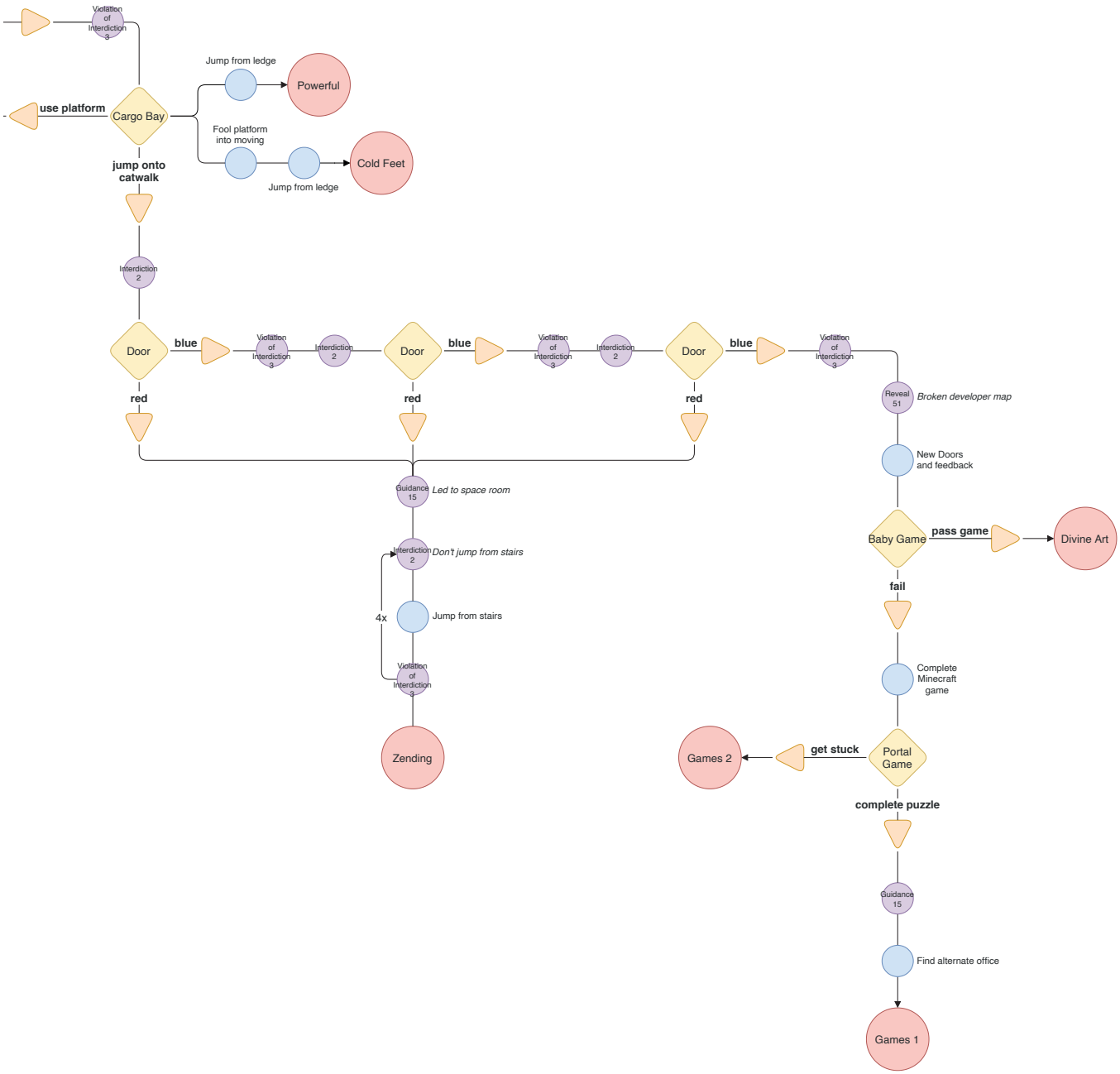
A.1 Propp's Morphology applied to *The Stanley Parable*

The pages that follow show a complete Proppian mapping of *The Stanley Parable* based on the modified functions and rules of Bostan *et al.* [29]. Due to the physical size of the diagram, it is split over multiple pages that join together. The corresponding alignment for a given page is noted with a label in the bottom-left.

The diagram makes use of several colors and shapes to represent different elements. The green circle labeled *Start* represents the initial starting point of the story. There is only one green circle as the player always begins at the same location. Red labeled circles denote a particular ending that the player can achieve by making appropriate choices from the start. Yellow diamonds represent an explicit branching choice given to the player within the narrative, and the orange triangles represent the potential decisions that can be made for a given choice. For example, the yellow diamond labeled *Door* represents the first major decision the player is given, where the narrator instructs them to pass through the left door (represented by the orange triangle labeled *left*), but they have the freedom to ignore the instruction and instead navigate through the right door (represented by the orange triangle labeled *right*). Purple labeled circles represent a Proppian function that the player experiences as they make their way through the narrative. Blue labeled circles represent a player action that is required to progress through the narrative to a particular ending that is unable to be mapped to a Proppian function. The dashed line, beginning at the green starting node and ending at the red *Freedom* ending, is considered the default pathway through the game, as it is achieved by directly following instructions given by the narrator at each decision point. This information is also available in the top-left of the first page in the *Legend*.


Legend
Start
Ending
Option
Decision
Player Action
Functions
Vanilla Path
sv_cheats 1
sv_cheats 1
sv_cheats 1
Serious
Input computer 419 and restart
Input computer 423 and restart
Input boss' secretary computer and restart
Input Stanley's computer
Heaven
Close door
Coward
Start (setup)
Lack 8
Coworkers missing
Interdiction 2
Out of Map
yes
no
Bored of Gag
Map is not broken
Reveal 51
Climb out of window
Trickery 6
Convince other route is better
left
Door
right
Violation of Interdiction 3
Guidance 15
Interdiction 2
Staircase
forward
Maintenance
left
Hallway
forward
upstairs
downstairs
take elevator
Escape Pod
Reach escape pod
Go back to office
Travel 37
get locked outside
Boss' Office
secret door
Violation of Interdiction 3
Reveal 51
Infinite walking
Insanity
Guidance 15
Narrator lost
Reveal 51
Early exposure to Mind Control
Reset until Door option
Recon 51
Search for missing story
Reset until Door option
Guidance 15
Backtrack to find story
Trickery 6
Fake win
Reset from start
Guidance 15
Follow The Line™
Trickery 6
Fake importance of fern and mislead path
Reset from start
Guidance 15
Narrator guides player against The Line™
Reveal 51
Everything is preplanned
Confusion
Reveal 51
Secret fireplace entrance
Interdiction 2
turn back
Warning about death
Interdiction 2
Violation of Interdiction 3
escape
Catwalk
Crushing machine
Struggle 16
Violation of Interdiction 3
Explore museum
Museum
mind control facility
Reveal 51
Employee monitoring and mind control
Interdiction 2
Freedom
Wedding 31
Freedom
turn off
Power Switch
turn on
Violation of Interdiction 3
Reveal 51
Narrator's evil doings
Struggle 16
Attempt to disarm Narrator's bomb
Explosion
Reveal 51
Interdiction 2
answer
Phone
unplug
Trickery 6
Reveal 51
Interdiction 2
Violation of Interdiction 3
Apartment
Violation of Interdiction 3
Reveal 51
Guidance 15
Interdiction 2
left
Doors
right
Guidance 15
Alternate boss' office
Interdiction 2
Violation of Interdiction 3
Violation of Interdiction 3
Reveal 51
Observe alternate Stanley
Reveal 51
Not Stanley

LEFT


Violation of Interdiction 3
use platform
Cargo Bay
Jump from ledge
Powerful
Fool platform into moving
Jump from ledge
Cold Feet
jump onto catwalk
Interdiction 2
Door
blue
Violation of Interdiction 3
Interdiction 2
Door
blue
Violation of Interdiction 3
Interdiction 2
Door
blue
Violation of Interdiction 3
red
red
red
Guidance 15
Led to space room
Interdiction 2
Don't jump from stairs
4x
Jump from stairs
Violation of Interdiction 3
Zending
Reveal 51
Broken developer map
New Doors and feedback
Baby Game
pass game
Divine Art
fail
Complete Minecraft game
Games 2
get stuck
Portal Game
complete puzzle
Guidance 15
Find alternate office
Games 1

RIGHT

APPENDIX B

Interface Paradigms & Workflow

B.1 Clustering Features

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 (color or 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 schematic-like interfaces 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.

Can Duplicate Content (`Yes/No`)
Is the user able to duplicate existing content to save time recreating them?

Structural Shortcuts (`Yes/No`)
Accelerators or shortcuts are present to quicken creation of 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`)
Can the program export for use in another system (e.g., existing reader/interpreter)?

B.2 Story Features

Game	Genre
Portal	Puzzle Platformer
Spec Ops: The Line	Shooter
Streets of Rage	Fighter/Brawler
Dishonored	Stealth
Alan Wake	Survival Horror
Project Diva	Rhythm
The Talos Principle	Puzzle Adventure
Assassin's Creed Origins	Action Adventure
The Beginner's Guide	Walking Simulator
Mass Effect 2	Action RPG
World of Warcraft	MMORPG
The Elder Scrolls V: Skyrim	Sandbox RPG
Halo Wars	Strategy
Tony Hawk's Project 8	Sports
Open Sorcery	Hypertext Narrative
Go! Go! Nippon	Visual Novel
Pathways of Destiny	Locative Hypertext

The following features were included in this story feature analysis.

Choices (`None/Diegetic/Extra-Diegetic`)
Are choices diegetic (without suspending story flow) or extra-diegetic (suspension of flow during choice)?

Resolve (`Linear/Branching/Perpetual`)
How does the story end? Is it linear, branching, or does it not end?

Narrator Type (`None/Dramatized/Undramatized`)
How dramatized the narrators are, based on Booth [28].

Narrator Significance (`None/Substantial/Insubstantial`)
Whether the narrator plays a key role in the story or not.

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.

Discoverable Narrative INEA (`Yes/No`)
Does the game feature Intangible/Narrative/Explicit/Active elements of Discoverable Narrative?

Discoverable Narrative TNEA (`Yes/No`)
Does the game feature Tangible/Narrative/Explicit/Active elements of Discoverable Narrative?

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?

Player Traits Alter Narrative (`Yes/No`)
Do player traits alter the narrative, such as morality points affecting which conversation options are available?

Mechanics as Metaphor (`Yes/No`)
Does the game use mechanics as a metaphor?

Environmental Storytelling (`Yes/No`)
Does the game feature environmental storytelling?

B.3 Demographics Questionnaire

What is your profession or field of study?
`Creative/Technical/Other`

What was your previous knowledge of the tool before this experiment?
`None/Little/Moderate/Plenty`

What is your previous knowledge of other similar tools before this experiment?
`None/Little/Moderate/Plenty`

What was your previous knowledge of IDN before this experiment?
`None/Little/Moderate/Plenty`

Was the short training video adequate as a brief introduction to the software?
`Yes/No`

B.4 Interview Questions

Tell me about how you felt the features in the tool were a hindrance to your goal of implementing your vision. What was it that frustrated you? 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 was it that satisfied you? 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.

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 the participants would change.

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 given more time.

B.5 Analysis Actions

Action Description (A)
Text 'spoken' by a narrator describing the state of the world with respect to a character's action or state.

Authored Branch Creation (B)
A purposely created divergence by the author to either provide explicit choice between distinct pathways or to be implicitly redirected along them based on a conditional check of the world state. Branches must be of some narrative significance. For example, an interactive object present in an optional location does not constitute a branch; if its presence in inventory caused a branch later on, that particular point would count, not its initial interaction. For adventure games, branching refers to distinct rooms that diverge from a common point where the player explicitly traverses, and implicit conditional traversal along one of two or greater pathways based on state checks. Blocking progression alone is not counted as a branch, unless in addition to the block, two or more distinct pathways are clear (such as different speech occurring on each option). This avoids simple blocks counting as branches.

Character Creation (C)
Adding a new character to the story. Characters must be distinct and not only mentioned in text. This includes animals and personified inanimate objects.

Documentation (D)
The author accesses documentation. This includes the official documentation as well as referencing back to the training video or searching the internet.

Guidance (G)
The author is guided by the moderator to avoid deadlock without explicitly asking for help. Guidance differs from a Query in that Queries have no following assistance.

Help (H)
The author explicitly asks the moderator for assistance.

Advanced Interactivity (I)
Adding behavior other than basic defaults to objects and the world. This includes adding custom rules to a world, custom attributes and commands, as well as more complex linking methods such as blocking pathways based on conditions.

Location Creation (L)
A location or room is initially specified. This includes renaming and repurposing of

temporary rooms in the case of the *Inform* and *Quest* templates, as well as using new 'Flow' nodes to segment stories in *Articy*.

Object Creation (O)
Adding a new distinct entity to the story that is not only mentioned in text.

Query (Q)
The author asks for clarification (but not assistance) on something in the program.

Character Speech (S)
A character explicitly speaks something. This can be done in text as long as the speaker is known in context.

Testing (T)
The author tests the story from within the program for previewing or debugging purposes.

Variable/Script Creation (V)
A variable or script is initially created for fine control over the story state.

World Building (W)
Present text to the player that describes and conveys the state of the world.

Notes
The World Building (**W**) action has a few caveats with its measurement. A single occurrence of **W** is an uninterrupted stream of descriptive text that terminates at its end or when interrupted by **S**. In the case that an author mixes **W** and **S**, each separate chunk of **W** is counted uniquely. Judgement must be given on a case-by-case basis as to whether the breaking **S** is substantial enough or can be considered a part of the **W**. The Action Description (**A**) is similar to **W** in that they both describe the state of the world. **A** differs from **W** in that it is 'spoken' from a disembodied actor with respect to the actions or state of a character. For example, 'You enter the forest…' or 'LRRH picks up the letter…' would be a form of **A** as it is spoken with respect to character actions rather than being a standalone descriptor of the world, which would instead be a form of **W**. In the case where a sentence contains clear aspects of **W** but begins as **A**, judgement must be taken as to which it primarily represents. Authored Branch Creation (**B**) refers exclusively to those that are intentionally created by authors. If a branch occurs not as a conscious decision, but by chance of playing around with the interface, then this is not counted. This is a subjective classification that is decided based on observation of the participant, comments from the interview process, and analysis of activity during video recordings.

B.6 Coding Categories

Unclear UI elements cause confusion.
Mention of an interface element or feature that is unclear, resulting in confusion or difficulty.

Inclusion of an intelligent assistant when writing code.
Mention of the desire for assistance when writing code, such as autocomplete or linting.

The graph system is good/helpful/etc.
Positive mention of a graph's functionality and helpfulness.

Elements of the interface are too small.
Comments on the size of elements in the interface being less than adequate.

Assistance to return to a specific state during testing.
Discussion of how a state-returning feature (play from here, Skein, Walkthrough) is helpful.

Using the tool is easy having gotten used to it.
Mention of the ease of use as familiarity increases.

Unclear terminology caused confusion.
Mention of terminology used in labels (and such) that resulted in confusion or difficulty of understanding.

Error messages are difficult to understand.
Complaints about the clarity or helpfulness of errors (including compile, runtime, and system).

Desire of a graph system when it wasn't present.
Mention of wanting a graph system where it did not exist, or if it did, it was not satisfactory.

Variables are useful.
Positive mention of the functionality of variables.

Support for more advanced dialogue functionality than built-in.
Mention of the desire for a dialogue system that extends the capabilities of the tool's existing functionality.

Difficulty in use of built-in syntax.
Complaints or concerns about the clarity and form of built-in syntax for scripting.

Populating the world with objects is easy.
Mention of ease of populating the game world.

Creation of the world structure is simple.
Praise for the ease of creating world layouts.

Hierarchies are useful.
Positive comments on the functionality of a hierarchy or outliner.

Variables should be easier to handle.
Mention of desire to simplify variables.

Creation of branches is easy.
Mentions of ease of creating branches regardless of the content.

Intuitive syntax.
Positive comments on the form of the built-in syntax for scripting.

The overall interface is overwhelming.
Complaints about the complexity of the interface and its lack of understandability as a whole.

Editing content is easy.
Praise for ease of editing content of elements such as object descriptions.

Shortcuts were useful to create layouts.
Positive mention of shortcuts for structure creation.

Support for more advanced game-centric features than built-in.
Mention of the desire for absent functionality that relates to game narrative, such as support for quest systems.

Good use of context menus.
Positive comments on the usefulness of context menus.

Intelligent syntax assistance is good.
Positive mention of existing intelligent assistance.

B.7 Little Red Riding Hood Script

```
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 well-trodden. 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
As Little Red Riding Hood peered through the brush in search of 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

As Little Red Riding Hood was walking along the path, a friendly-looking 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
(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

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

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. 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

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, thanks for saving me and my granddaughter. 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, Wolf-free 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 two-for-one meal-deal. **THE END.**

[**END**]

APPENDIX C

A Pipeline for Authoring Tool Design

C.1 Persona

Catarina Drake Narrative Designer, Libertalia Games Ltd.

"I'm a gamer. Working in games writing is a dream for me. I enjoy the creative process of taking ideas and translating them into gameplay that players around the world can enjoy."

"I know my place in the pipeline. I want to work with systems that let me get my job done efficiently without having to battle against them."

About Catarina

Catarina is a storytelling whiz and an avid gamer. She studied Creative Writing in college and went on to pursue a Games Design degree at university, picking up some basic knowledge of programming entering various game jams. She entered the games industry as a designer, but her strong knowledge of storytelling resulted in her gradually taking on narrative designer roles. She has worked at a number of studios and has seen multiple AAA games through to release. Her passion is to take what writers envision and help to make that vision become a reality by any means possible.

Behavioral Considerations

- Enjoys working with game engines to help implement stories (Unity, Unreal, etc.).
- Is interested in programming but is not an expert.
- Enjoys working with writers due to her background but prefers to work on structure and implementation with them rather than on the writing itself.
- Wants to ensure that their knowledge of interactive narrative is properly used.
- Likes to work with a team rather than alone.
- Enjoys working with VO talent.

Frustrations

- Doesn't like it when most of her tasks become writing, but enjoys the occasional contribution.
- Dislikes working with tedious or mediocre tools that get in the way of implementation.

Tasks

- Work with the writing team to ensure that their vision of the game's narrative is correctly implemented and to communicate any technical limitations that they may be bound by.
- Work with the writers to create informed narrative structures for the game.

- Assist with planning and implementation of narrative such as with scripted events and sequences.
- Oversee and contribute to the game's narrative design documentation.
- Assist with implementation of dialogue and other additional narrative texts within the game.
- Be a point of contact and work with film/VO talent.

C.2 Participatory Design Information Sheet

Thank you for taking part in this experiment.

The goal of this experiment is to collaboratively design a user interface and user experience for an authoring tool for writing narrative structure for video games. The background research and technology for this tool already exist, so we will be collaborating only on the interface and experience. Primarily, this is focusing on the *visualization of the content* rather than the program as a whole.

The remainder of this document will provide various background information about the authoring tool. Please read it in detail and make sure you understand what it is saying. If you have any questions at all, please ask.

After reading the document, you will be given a piece of paper and asked to roughly sketch the type of interface that initially comes to your mind after hearing about a system like this. This can be in any form that you see fit; there is not a wrong answer. You can have as many sheets as you need. We will then briefly discuss your design to gather further insight.

Finally, you will be presented with two of my own prototype ideas and we will discuss those in the same light.

Thank you again for your participation!

Persona

Please see the persona above this appendix entry.

Tool Vision

The high-level concept of the tool.

This tool is designed for creating narrative structures for use in video games. It is aimed at Narrative Designers that take content and concepts created by writers, and create narrative structure that implements this vision into the game. This includes both creating

structures that control the game (such as delivery of dialogue) but also includes creating structure to respond to in-game happenings to provide dynamic narrative experiences.

Model Structure

The technology that the tool must implement.

The model is a three-layer hierarchical representation of video game narrative. Primary elements are Groups, which contain nested Sequences, which contain nested Events. Groups can also nest within other Groups. There is always one main Group at the topmost level. A Group is the highest level element which acts as a folder and controls the scope of its contents (i.e., its contents can only be active when it is also active). Sequences represent blocks of narrative content but mostly serve a structural and organizational purpose. Finally, Events are individual narrative actions; these tell the game to do things. Additionally, there exists Hubs and Returns, which work together with other elements to provide a cyclic set of sequences, such as dialogue wheels with different pathway choices in Mass Effect or Life is Strange. Sibling elements (i.e., those at the same depth in the hierarchy) can be connected together with Links. Linkage is not always sequential nor is required. For example, a single Sequence could link to none, one, two, or even fifty other Sequences. All of the elements, including Links, are tangible objects and each has many properties, including assigning scripts that can be edited to specify their behavior. A scripting system called Logic (which when implemented uses something like Lua or Python) is also present which can control the model and communicate bidirectionally with a game engine. That is, it can query the game using script from within the model to react to the game, or the game itself can control the model forcefully. Named variables keep track of the state of the model. Objects of importance to the narrative are represented as Entities and can have enumerated Tags to help identify them.

By 'tangible object' I mean something that can be seen and interacted with. An intangible Link, for example, would connect two elements together but would never be seen visually. A tangible Link, on the other hand, would have a clear line (or similar) that can be seen and interacted with by the user of the tool.

If you would like to know more about the model before continuing, please ask now. If you do ask, you will be given a section from a publication on this model which contains detailed descriptions of how it functions.

Model Features

The basics that the program should have.

Users need to be able to create the objects described above and edit all of their properties. These objects are structured hierarchically but can be displayed in any way as long as the structure and connectivity are understandable by users. All objects, including connections, should have editable properties somehow. Some properties modify an object's behavior, and it would be good to show this on the object itself. Connectivity between the objects should ideally be clear and easy to modify to enable rapid iteration and visualization of the structure. The tool needs to support some form of implementation of the Logic scripting system including editing and assigning to objects (via their properties). There needs to be some kind of management system for Variables, Entities, and Tags.

Your Task

You will now be given a piece of paper (if you need more, please just ask).

Let's say that an authoring tool following this description already exists. You start the program and open a save file of a story structure that you were working on. After the program starts, what do you see? What does the interface look like? Where are the objects and how do you show their hierarchy and connections?

Please sketch on the paper what kind of interface comes to mind based on this document's description. To reiterate, you are not expected to design a complete program here, only the *visualization* of the content described above.

The sketch can be **very rough** and is not expected to be perfect; it's just about getting ideas down quickly. Boxes and lines are perfectly acceptable.

Please consider annotating your design as it will make discussion a lot easier.

C.3 RITE Study Information Sheet

Thank you for taking part in this experiment.

The goal of this experiment is to refine a prototype user interface and user experience for an authoring tool for writing narrative structures for video games.

The remainder of this document will provide some context as to the underlying functionality of the prototype authoring tool design. Please read it in detail and make sure you understand what it is saying. If you have any questions at all, please ask.

After reading this document, the main experiment will begin. The concepts explained to you and questions asked will help in refining the user experience and support for video

game narrative authoring.

You are **NOT** being tested. The **application** is being tested.
There are no incorrect answers!

Thank you again for your participation!

Persona

This tool is designed for Narrative Designers that work with narrative structure. A Narrative Designer is typically more technical than a regular designer, working on integration of creative concepts into the game engine, and are experts in narrative structure. They have working knowledge of programming and scripting languages but are not experts.

Please see the persona above this appendix entry.

Overview

This tool is for authoring narrative structures chiefly for video games. It is designed to be as generalized as possible, which means it cannot contain game-specific, genre-specific, or engine-specific interactions. It is designed to be a **desktop application**; that is, used with a regular mouse and keyboard. Editing in the tool takes place in a flowchart-like editor using varying types of nodes connected with lines. The nodes are organized in a hierarchical structure. The behavior of these nodes is controlled by an extensible scripting language called Logic. This includes communication with the game engine. The main interface of the tool is broken up into three parts: the Outliner, the Canvas, and the Toolbar. This will be shown to you later.

Node Types

The main node types are **Groups**, **Sequences**, **Events**, **Hubs**, and **Returns**. A Group is like a folder that works mostly for structural organization; they can contain other Groups and Sequences. Sequences are a middle layer that can contain Events. Events are the leaf nodes that correspond to individual actions within a game and cannot have anything placed inside of them. Hubs are center points for multiple pathways to branch out and return using Return nodes, like in a dialogue wheel system, and both are siblings to Events. All nodes have a number of scripts that trigger at different times to let the game "do things" when various things happen. Examples of this include when a node is entered or exited. Some nodes have unique scripts, such as Hubs having a script for when they are returned to after following a pathway. Nodes can be arbitrarily grouped together using Frames; a purely organizational and visual way to make a node graph more readable.

Scripting with Logic

Scripting is done with a syntax called Logic. This scripting system can communicate directly with the game engine to query it and to instruct it to do things. In fact, Event nodes have a "do script" which instructs that node on what it actually does to the game. Similarly, all nodes have conditions that determine whether or not they can execute. In these scripts, you could query the game for player information, for instance.

Example Videos

We'll now look at three short example videos from Mass Effect (BioWare, 2007):

- A forced conversation with Joker, Kaidan, and Anderson.
- Triggered ambient dialogue between Adams and Presley.
- An optional conversation with Presley.

Live Questions

You will be given a number of tasks along with a partially interactive prototype of the authoring tool design. Due to the limited functionality of the prototype, you will be sometimes asked to describe how you would achieve a task rather than actually doing it. Please discuss this process out loud and use the prototype to assist in explaining.

C.4 Mandatory Functionality & RITE Tasks

Mandatory Functionality

Authoring

- *Canvas & Nodes*
  - Create and delete nodes
  - Move nodes around the Canvas
  - Group nodes with Frames
  - Toggle node display modes
  - Identify node types at a glance
  - Identify connected and conditioned pathways
  - Edit label and description of each node
  - Toggle states of each node
  - Create, modify, and delete Node Templates
  - Insert a Node Template
  - Navigate around the project
- *Outliner*
  - Identify active object in Outliner
  - Find collapsed objects in Outliner
- *Connections*
  - Add/remove links on a node
  - Connect and disconnect links with nodes
- *Scripts/Variables*
  - Create, modify, and delete internal variables
  - Create, modify, and delete Named Scripts
  - Insert Script Template into a script
  - Create, modify, and delete Script Templates
  - Assign Named Scripts to a node
  - Edit and assign node's Custom Scripts

Testing

- Test starting from any node
- Understand the purpose of the testing panels
- Navigate along a path in all directions
- Identify which variables impact outputs
- Tweak variables to enable/disable nodes
- Store/restore variable panel states
- Understand how to activate triggers
- Understand how to use the Recordings panel.

RITE Tasks

1. You want to add an extra dialogue option to the Event conversation chain inside the Joker Sequence. Walk me through how you would go about adding a new Event to the Canvas.
2. Describe to me how you would move the newly created node to a position you like.
3. Now walk me through how you would go about deleting the newly added Node.
4. You feel that some of the nodes in the Joker Sequence could be grouped together. You have made a marquee selection of these nodes. Walk me through how you would add a Frame that encompasses these nodes.
5. Nodes have different display modes (Simple, Extended, Artboard) to let you see more or less information. You want to see more information of the Joker Sequence. Walk me through how you would toggle the mode on the Joker Sequence from Simple to Extended, and then to Artboard mode.
6. You want to refine the description and label for the PresleyAdams Sequence. Walk me through how you would edit the properties of this node.
7. Show me how you would toggle the states of the same node.
8. Can you please identify the types of the three nodes in the Canvas?
9. The Joker Sequence is in Artboard form and you can see its internals. Can you identify the types of nodes inside it?
10. Can you please navigate into the Presley Sequence?
11. In the Outliner, can you identify each of the node types from the top down?
12. Can you now navigate to the Joker Sequence and identify which link has a dynamic condition attached to it?
13. You want to save time and create a reusable Event Node Template for dialogues. Walk me through how you would create a new Event Node Template using the Node Template Editor.
14. Walk me through how you would now add a Node Template to the Canvas.
15. The 'root' of the Canvas is the node whose contents is currently being displayed for editing. Can you please identify the current root of the Canvas (not its contents) and then find it in the Outliner? How did you find it?
16. Walk me through how you would find the Event with the label **Stealth systems** in the Outliner. At any point, it may or may not be visible in the Outliner.
17. You accidentally disconnected the Event labeled **Official story**. It should link to the three following Events labeled **I agree**, **You're overreacting**, and **Cut the chatter**. Walk me through how you would add three links to the **Official story** connection panel and how you would connect the links to the mentioned Events. (NOTE: Participant should perform this specific task on a desktop computer to use hover behaviors)
18. It seems the third connection was incorrect. It should connect to **Cut the chatter** instead. Walk me through how you would go about removing the connection and connecting it to the correct one.
19. Walk me through how you would remove these links completely from the connections panel if you had added too many.
20. Some dynamic behaviors require variables to be created in the tool. Walk me through how you would create a new Integer variable using the Variable Editor.
21. Please describe how you would modify the variables and delete them.
22. Named Scripts save time by creating reusable scripts. Show me how you would, using the Script Editor, create a new Condition script and insert some code into it.
23. Please describe your interpretation of the code editing panel.
24. Script Templates allow for reusable snippets to be inserted into other scripts, such as those you have just created. Walk me through how you would insert a Script Template into the Condition script you just created.
25. Walk me through how you would create one of these Script Templates.

26. Please describe your interpretation of the right-side panel.
27. Having created a Named Script in the Script Editor earlier, walk me through how you would change the assigned Condition of the **PresleyAdams** Sequence node to be this new Named Script.
28. A node can have Custom Scripts, which are one-time unique scripts assigned to them. Walk me through how you would edit and assign a Custom Script for the **Presley** node. (After completing) Describe your interpretation of the Scripts window.
29. Walk me through how you would begin testing from the **Introduction** Group.
30. Please describe your interpretation of the testing interface and describe its functionality.
31. Please navigate and follow the story until **Status report** is the current node.
32. Please identify which variables affect the outputs and tweak them to enable the disabled output. Continue the story after.
33. Variable states can be saved and loaded to avoid having to change variables each time. Walk me through how you would assign an existing state, update a modified state, create a new state, delete an existing state, and restore default values of variables. Please begin by identifying where variable states can be managed.
34. Activate and follow the **Presley** Sequence trigger.
35. Recordings allow you to save a path through the story (starting from any node and ending on any node you can traverse to) so that you can replay it later, including relevant variables states. Let's traverse to the **Status report** node. Walk me through how you would now save a new Recording of this path, and describe what the path would contain.
36. Assuming that we closed and reopened the Simulation mode at a random point, please now describe how you would load an existing Recording for playback, and describe how you would follow through the playback.

C.5 Design Notebook

The application design described in this document is agnostic of the operating system environment. However, keys found on Apple hardware are frequently used throughout the text in order to prescribe actual interactions for a modern operating system. These keys can be equally replaced with keys associated with non-Apple hardware. For example, Apple's Option (⌥) key is analogous to Alt, Command (⌘) is analogous to Ctrl, and context-clicking is equivalent to right-clicking.

Application Window

The main application window is divided into two panels split vertically. The left panel contains a hierarchical listing of the project's constituents ('Outliner'). The right panel contains a spatial hypertext graph-based editor ('Canvas').

Outliner

Hierarchy

The Outliner provides a hierarchical preview of the contents of the current project. Each item in the hierarchy is given a designated icon to help determine its type along

with a user-specified label. Nested elements are displayed with indentation represent the hierarchal relationships, with vertical lines used to assist with visual alignment. Hierarchy levels can be expanded and collapsed to save space, which can be toggled by clicking on an entry's icon. All Canvas objects except Frames are listed in the Outliner (see 'Frames' for more detail).

Sorting

The order of items in the Outliner is custom, although by default it will append items to the bottom of its respective hierarchical depth. Context options are present to sort alphabetically at any level of the hierarchy.

Rooting

When an item is the root of the Canvas (see 'Canvas' for details), the Outliner highlights this item with a shaded background. If the root changes, the highlighted item in the Outliner updates accordingly.

Clicking Items

Single-clicking an item in the Canvas will also select it in the Outliner. When an item in the Outliner is selected, a small indicator to the right of the item's label appears. Holding Command (⌘) when selecting additional elements will toggle their inclusion in a multiple selection. Double-clicking an item in the Outliner will reroot the Canvas to that item. If a leaf node that cannot be rooted is selected, then the Canvas will instead reroot to its parent and it will become selected in the Canvas. Alternatively, this behavior can be mimicked for any item by holding Option (⌥) when double-clicking an item. Context-clicking on an item in the Outliner will open a context menu for that item. Appropriate menu items appear, such as selecting the element, rooting to the element, renaming the element, deleting the element, alongside generic items such as sorting the children of the selected item. There is also an item present for changing the display mode of all direct child nodes as an alternative to the same action being done one by one in the Canvas. New elements cannot be added through the context menu of the Outliner (although this could be changed to add at the geometric center of existing nodes).

Dragging Items

Items within the Outliner, including multiple selections, can be dragged between levels of the hierarchy providing that the requested move was valid. They will retain their original position on the Canvas in their new parent and all incoming and outgoing connections will be destroyed.

Filtering

Below the item listing is a filter field. When the text of the filter field is not empty, the Outliner will only show elements that match by their name in a collapsed state (i.e., you could expand the children of the matching nodes even if the children do not match). A button to the right of the filter field provides a popup to determine visibility of items in the Outliner by type. The result of the filter affects only the Outliner and the Canvas remains unaffected. Optional advanced usage allows for visibility to be specified directly within the text field. Including `typ:grp/seq/evt/rtn/hub` controls the visibility based on the type of node.

Canvas

The Canvas is the primary editing portion of the interface. It is a spatial hypertext graph interface for creation, connection, and management of nodes. The Canvas is backed by a regular grid that can be panned and zoomed.

Rooting

Any non-leaf node can become the 'root' of the Canvas. When a node is the root of the Canvas, the workspace displays the children of that node and any editing done is for the children of that node. If a child node contains a further hierarchy, then they can instead be rendered as Artboards (see 'Artboards'). This means that for any given rooting, editing is done one layer deep (direct children), but it is possible to see two layers deep through Artboards, which provides more context to the editing process. Displayed in the top left of the Canvas is the label associated with the currently rooted node. To the left of this label is a button for surfacing up the hierarchy (i.e., rerooting to the current root's parent).

Clicking Items

Single-clicking on a node will select it. Multiple items can be selected by holding Command (⌘) to toggle the selected state of a node. A marquee selection can also be created by dragging from an empty space. Double-clicking on the label of a node will edit its label inline, and the same for the description when in Extended mode. Doing so on Artboards will have no effect. Context-clicking on an empty space with the Canvas will open a context menu containing items relating to creation of new nodes as well as graph-specific options such as being able to surface up from the current rooted item and fitting the view to the selected items. If a new node is selected from the menu, then it is created at the mouse position. Context-clicking on a node will open a menu with items particular to that type of node. Options always include the ability to select the node, delete the node, or open the properties for the node. For nodes that can be rooted, the option to open them is also present.

Moving Items

Nodes and Frames can be moved by clicking and dragging them. This applies to multiple selections made with a marquee too. Grid snapping can be enabled with an optional preference, forcing nodes to snap to the background grid. The size of the grid can also be controlled through the preferences. Grid snapping can be temporarily enabled by holding the shift key while dragging a node.

Traversal

Traversal of the Canvas is done with panning and zooming. Panning allows for horizontal and vertical movement of the view. Zooming allows for an arbitrary yet clamped magnification or minification of the view. If a selection is made on the Canvas, then using the corresponding context menu item, the view can be zoomed and positioned to encapsulate the selection.

Nodes

Creation

As mentioned in 'Canvas/Clicking Items', the Canvas' context menu can be used to create nodes. Alternatively, icon buttons in the main toolbar are present for Groups, Sequences, Events, Hubs, and Returns, each of which has a padded, dotted outline border indicating an irregular interaction. Authors can drag from these buttons into the Canvas to create a corresponding node where the drop target is, providing the node is allowed in the current root. When a node is created, it is given a default name such as 'New Event'. Upon being initially added to the Canvas, the node's label is editable by typing to allow the user the ability to quickly name the node. If an interaction takes place outside of the node, or the user submits the editable label, then the name will be committed but can be edited later.

Appearance

Nodes are the atomic elements that make up the hierarchy. Each type of Node has a specific icon and color associated with it that is used to help differentiate it within the interface. This applies to both Outliner and the Canvas rendering of Nodes. All Nodes except for Hubs and Returns have a label and description. An option in Node context menus allows toggling between rendering as a minified version (icon, label), a detailed version (icon, label, description), and an Artboard (where applicable).

Artboards

A node can be drawn as an Artboard if it has children. An Artboard is essentially a folder that provides a lightweight preview into its internal contents one layer deep (i.e., its direct children). The contents of an Artboard are not editable, although its contents can

be panned if it is too large to fit within the preview area. Any interactions with a node in Artboard form are the same as any other respective sibling in a different mode. An Artboard's appearance is different from that of a regular node. It has a shaded background with a header at the top containing the appropriate type icon and label. Descriptions are not rendered in Artboard mode to save space. The top-right of the shaded area has a button which when clicked will reroot the Canvas to the Artboard. The main body of the Artboard is below the header, which contains a non-interactive rendering of the internal contents of the node. The preview contents of an Artboard (i.e., its direct children) are not rendered as complex nodes but instead in a restricted form containing only a colored shape and respective icon. This is because rendering a full Canvas at this scale is impractical. The preview attempts to automatically arrange the nodes as cleanly as possible. Artboards can be resized to provide a bigger view but are constrained to a maximum and minimum size.

States

Nodes (including those in Artboard form) have icons overlapping their border that are enabled based on various states of the node. This applies to the concept of 'Topmost', 'Keep Alive', and 'Parallel' states. These icons help to relay important metadata about nodes that could alter the authoring decisions. For example, one may treat a parallel node differently from that in the middle of a chain. Similarly, if the node is an entry point to a parent, it is marked with an arrow pointing to its left side.

Scripts

All nodes have varying configurations of Functions, Conditions, and Selectors, determined by the type of node. From the node's properties panel (see 'Properties' below), Named Scripts can be assigned to these slots by selecting them from corresponding combo boxes. The combo boxes can be set to 'None' to ensure that no script runs at all. Alternatively, the combo boxes can be set to 'Custom', which will use a script unique to this particular node instance rather than a reusable Named Script. From the node's context menu, a Scripts window can be opened which is used to edit custom scripts unique to this particular node instance. These scripts are not shareable or reusable and exist only within this node. The Scripts window is a floating window that will remain open until manually closed. This window edits only its assigned node, but many different Scripts windows can be open for different nodes in parallel.

Scripts Window — General

Within this window is a maximized script editor. This is an augmented text editor for typing in code. It has syntax highlighting and fuzzy autocomplete of keywords. Keywords specific to a project that are extensions of Logic can be imported from file from

the application's menu. This configuration is a JSON file that includes keys and values for name and data type. The imported keywords are then accessible to syntax highlighting based on their type, autocomplete based on their name, and become exposed to External variables (see 'Simulation Mode/Variables Panel'). Above the code editing section is a toolbar containing a button and three combo boxes. Any changes made to the scripts are automatically saved.

Scripts Window — Script Templates

The button, when pressed, opens a floating panel to insert Script Templates. The floating Script Template Preview panel consists of a scrollable and searchable list, where each item in the list displays a code icon and a user label. When an item in the list is selected, a panel to its right displays the user label, user description, and a preview of the code that will be inserted. If a listing is double-clicked, its code will be inserted at the current caret position in the Script Editor. Below the preview also has two buttons labeled 'Copy' and 'Insert'. Pressing the Copy button will copy the active item's code to the clipboard but will not insert it into the Script Editor. Pressing Insert will perform the same behavior as double-clicking an item. Alternatively, a Script Template can be interacted with using a content menu activated by context-clicking the listing or a cell. This contains both Copy and Insert entries which have the same functionality as the corresponding buttons. When a Script Template is inserted, the floating window automatically closes.

Scripts Window — Combo Boxes

The first of the combo boxes selects the language being used (such as Python or Lua), which is saved per script. The second combo box lists the global color scheme which applies across the complete program. The final combo box lets the user select which script of this particular node they are editing, such as the 'Entry Function' or 'Exit Function' custom scripts. Selecting which script is being edited and inserting code does not automatically enable it. The corresponding combo box in the node's properties must be set to 'Custom' to ensure that this code will run.

Scripts Window — Errors

If errors are present within a given script, the culprit lines are highlighted in red. To the right side of the line, a darkened region contains a warning icon, and to its right, a truncated description of the error, which is helpful for those that are familiar with them. At the bottom of the script editor is a vertically collapsible error panel that displays in a table all errors including their line number and full text description. The gutter of this panel has a background shaded the same as the error bar, and an icon for each error, to better communicate that it is an error panel.

Properties

Properties of nodes are edited using a floating popup window. This window is accessible by the node's context menu. The window is deliberately designed to be a temporary floating popup attached to the location of the target node to save on interface real estate and ensure that the Canvas editing experience is not blocked or obscured by large interface elements. As soon as interaction takes place outside of the window, it automatically closes. Within the window, properties are edited using mostly standard form controls (text boxes, combo boxes, property grids). At the top right of the window is a button that allows the properties panel to be detached. This opens a unique window that is topmost to the interface. The properties in this new window target whichever node is currently selected in the Canvas. There is a padlock button at the top which locks the window from updating when the selection changes. The presence of this window does not change the property editing behavior of any other node.

Node Templates

A downside to the single leaf node design (i.e., Events) is that common nodes with a particular use-case are repetitive and sometimes somewhat tedious to recreate, as the same process must be followed for each instance and user error can be introduced. Node Templates mitigate this by providing custom configurations of nodes that act as archetypes and can be instantiated without the need to manually configure each one from scratch. A Node Template can set all properties of a node, including the contents of its scripts. This concept of Node Templates applies to Groups, Sequences, and Events, Hubs, and Returns. As an example, if a node for narrator speech was common, creating and assigning a new script for each node would be obnoxious. Instead, a Node Template could be created that sets the configuration of the node and also assigns or specifies any necessary script to make it work. This template can then be instantiated with the same properties which saves time and reduces the likelihood of human error. Node Templates are created in the Node Template Editor panel of the Manager and are inserted into the Canvas using a popup (see 'Manager/Node Template Editor' for details). Alternatively, there is a context menu entry when clicking on a node that allows that node in its current state to be quickly used to generate a new template of the node.

Frames

Frames act as visual containers for collections of nodes on a Canvas. Visually, they have a colored background as well as a customizable label for identification of its contents. Frames are purely organizational and have no impact upon the hierarchical structure of the project. They are not present in the Outliner as they do not contribute to nor modify the hierarchy. Nodes can be freely dragged into and out of a Frame's bounding box. When moving a Frame in the Canvas by dragging it, all nodes present within its bounding

area are also moved. This is done in real-time and has no impact upon the hierarchy. Simply moving a node out of the Frame's bounding area is enough to disable its influence over that node. Frames can overlap, although this is not advised. If a node is present in multiple Frame bounding boxes, then the node moves when either of the Frames are moved due to the calculation of what is being contained being done in real-time. Frames can be resized by dragging the edges, but there is a minimum and maximum size to avoid extremes. Frames are created like nodes, through the context menu of the Canvas. Alternatively, alongside the node icons in the toolbar is an icon button for Frames, too. If a selection is already made, then the created Frame will auto-fit to the bounds of the selection. If using the toolbar icon, single-clicking will create an encompassing Frame if a selection exists, otherwise a Frame must be dragged from the button. By context-clicking on a Frame, a context menu opens for that Frame with an option to open the properties as well as deleting the Frame.

Connections

The flow between nodes is determined by connections between them. A node can have any number of outgoing connections to any sibling nodes.

Connection Panel

Connections for a given node are handled with a floating panel to the immediate right of the node, which is centered vertically. This panel contains zero or more Links, which are represented as vertically stacked circles. Context-clicking on the floating panel will open a context menu for adding links. If context-clicking on a Link, the displayed context menu additionally contains options to delete the Link and to edit its properties. Hovering over the connection panel will reveal a '+' button, which when clicked, will add a new Link.

Making Connections

Connections are made by dragging from a Link's origin circle to the target destination node. Alternatively for accessibility, a single-click on a link will enter act as if the user was holding the mouse button, which is terminated by either pressing the escape key or clicking again on a target. If a valid connection was requested, then the connection is made, a line is drawn between the Link and the target node, and the Link's origin circle fills with the same color as the target node type. If a connected Link has a script assigned to its condition, then two slashes (//) intersect the center of the line. Clicking and dragging on an existing line will disconnect the line from the target, and, as long as the mouse button remains held, stays in the state as if a new connection were being made. In this state, the user can drag and drop the line onto another node to create a new connection faster. A special case exists for where this line is dragged onto another node

with only a single input when the shift modifier is held. In such cases, the targets of the two lines are swapped. This is useful for inverting two single connections. The design does not allow for this to happen when more than one input exists on a node as there is no distinction between the inputs (although the design could be modified to have explicit input slots).

Link Coloring

When a node is not selected, its output Link lines are colored by the color of their target nodes. This helps to visually identify what kind of node is coming next without actually seeing it. When a node is selected, all of its output Link lines highlight and animate a dashed line towards their target nodes. This is useful in situations where connections become too difficult to determine due to overlapping lines. When two nodes that are distantly connected are selected, all potential pathways between them highlight and animate in the same way selecting a single node does. This makes it easier to identify the potential ways of getting from A to B and can work as a debugging tool in the event that undesired pathways exist (or do not exist).

Subdivision

Connected lines can be subdivided for rerouting to make visual organization clearer and more personalized. When double-clicking on a line, a pin is created at that point, creating a subdivision. Double-clicking on the pin again will remove it. Clicking and dragging a pin can allow for moving it around, rerouting the line. These pins serve pure as reduction and cannot be branched out from.

Script Templates

Similar to Node Templates, a Script Template provides a reusable snippet of code that can be inserted into any other script. This means that typing complex or even just repetitive scripts can be made easier by the inclusion of these code snippets. Script Templates are created in the Script Template Editor (see 'Manager/Script Template Editor') and are used chiefly in the Script Editor (see 'Manager/Script Editor').

Manager Window

The Manager window is a single-instance popup window that contains a tabbed interface. It has a tab bar at the top of the window with four tabs, each representing a different page, with the active tab being highlighted. Tabs include the Variable Editor, Script Editor, Script Template Editor, and Node Template Editor. A button in the toolbar of the main window below the Canvas opens the Manager window. It was decided to use a combined Manager instead of separate windows to help bring together similar creation and management functionality that is not related to Canvas editing authorship. This also

reduces the amount of real estate taken up in the interface as the Manager only requires one button to launch and only one window to access four different panels. Another advantage of congregating the tabs is that with a single window open, all four editors can be accessed without having to open separate windows. A downside is that only one tab can be open at a time, but the benefits of this design outweigh this factor.

Variable Editor

The Variable Editor is a simple creation and management interface for variables. The panel contains a table view in detail mode (consisting of rows and columns). Columns include Name, Constant, Type, and Value. The Name column represents the unique name for the variable and is displayed as a text field. The Constant column represents whether the Variable can be changed in runtime and is displayed as a checkbox. The Type column represents the data type of the variable and is represented by a combo box. The Value column represents the initial value of the Variable and is represented either by a combo box for Booleans or a text field for integers and doubles. All properties are edited in place. Variables are created and deleted from within a context menu accessible by context-clicking on the table view. At the bottom of the window below the table is a section for filtering the Variable table listing. Typing text will filter by the Name column. A button next to the filter field allows for hiding and showing of given data types and truth value of the Constant column. Optional advanced usage allows visibility to be specified directly within the text field. Including `typ:bool/int/real` controls visibility for data type. Including `const:t/f` controls visibility for Constant truthiness. Including `val:</>/<=/>=` followed by a number will check and filter accordingly providing they are digit values.

Script Editor

The Script Editor is a basic code editor and management interface for reusable Named Scripts for Functions, Conditions, and Selectors. The intended purpose of these scripts is to be shared and reused throughout many nodes, in contrast to scripts entered directly into a node which are unique and cannot be shared. The panel contains a headed cell-based listing on the left. Each cell represents either a Function, Condition, or Selector. A cell has an icon to help identify the type of script with a user-specified label to its right. Below the icon and label is a text field for an optional user-specified description of the script. The label and description can be edited in place within a cell by double-clicking the elements. To the right of the listing is a code editor. This code editor is the same as the one found in the Script Window when editing scripts unique to nodes. The only difference is that the combo box selecting the script to edit is not present. When a cell is selected in the listing, the cell's background changes color to indicate its selected state and the code editor updates to display the respective script. Scripts are created and deleted using the listing's context menu accessible by context-clicking on it or one of the

cells. When added, a new cell becomes the active selection. When deleted, the closest cell upwards becomes active. Alternatively, Scripts can be created using a '+' button in the listing's header which opens a context menu for selecting the type of script to create. At the bottom of the listing is a filter. Typing text will filter by both label and description. A button next to the filter field allows for hiding and showing of any of the three script types. Optional advanced usage allows for visibility to be specified within the text field. Including `typ:fun/con/sel` controls visibility based on script type.

Script Template Editor

The Script Template Editor is a panel for creation and management of Script Templates. Script Templates are not linked to a particular project and can be freely imported and exported as desired. On the left of the panel is a headed listing containing a code icon and user label for each cell. At the bottom of the listing is a search panel which filters the entries by their label and description. To the right of the listing is a panel that changes contents based on the active selection in the listing. This panel contains a text field for the label, a larger multiline text field for an optional description, and the same code editor found in the Script Editor but without the button to insert Script Templates. Context-clicking on the listing or a cell opens a context menu. This has options to create new Script Templates, delete the selected Script Template, remove all Script Templates, export all Script Templates to an external file, and loading Script Templates from an external file. Alternatively, Script Templates can be created using a '+' button in the listing's header which opens a context menu for creating a Script Template.

Node Template Editor

The Node Template Editor is a manager for creating archetype nodes for use in the Canvas. Node Templates are not linked to a particular project and can be freely imported and exported as desired. On the left of the panel is a headed listing with each cell containing an icon representing the type of node (Group, Sequence, Event, etc.), and a label to its right. At the bottom of the listing is a search panel which filters entries by their label and description. To the right is a panel that changes based on the active selection, but the majority of it remains the same. All properties of a node can be configured in this panel, which is laid out vertically, stating with the node's label and an optional multiline description. Following this is the attributes table adjoined with Activations and three checkboxes for Keep Alive, Parallel, and Topmost (Entry is not included as that is a per-parent setting). Below is a repeating set of combo boxes and code editors for the Condition, Entry Function, Do Function (if present), Exit Function, and any other necessary functions for specific node types. If a combo box has 'None', then no script is attached, which is the default option. If a named script is selected from the combo box, then it is used. If 'Custom' is selected in the combo box, then the script entered in the accompanying code editor is used. Context-clicking on the listing or one of its cells opens

a context menu. This has options for adding a new Node Template by type, deleting a selected Node Template, clearing all Node Templates, exporting all Node Template to an external file, and loading Node Templates from an external file. Alternatively, Node Templates can be created using a '+' button in the listing's header which opens a context menu for selecting the type of Node Template to create. In the main toolbar below the Canvas is a button that opens a popup for inserting Node Templates into the Canvas. This floating window contains the same listing as above, but its context menu is restricted to a single 'Insert' entry which when clicked will insert the corresponding Node Template into the center of the Canvas. The panel to the right that previously contained editable controls have been replaced with static or similar controls (such as text boxes being replaced by labels). Below the preview controls is an 'Insert' button which has the same functionality as the context menu entry. Dragging a Node Template cell from the listing allows for a given Node Template to be dragged to a location that the user desires instead of just the center. Inserting a Node will close the window. Alternatively, the context menu used for adding nodes contains an entry that expands to show all Node Templates. This is useful when the user knows what they are looking for and does not care for detail of each template.

Simulation Mode

Simulation Mode provides a lightweight simulation of the structure and content of the project without having to rely on an external engine. Some limitations apply to the simulation as a result of the execution of Logic being deferred to an external engine. Additionally, due to the simulation window handling one node at a time, it cannot accurately depict multithreaded behavior; any interruption will redirect the currently active node. The Simulation Mode window can be accessed by choosing 'Test from here' on any node's context menu. Doing so will open a floating Simulation Mode window and prepare its contents to display the selected node. The window is divided into three panels: a panel to preview the current node's properties and content, a panel for manipulation of internal and external variables, and a panel hosting connections to other nodes.

Preview Panel

The preview panel shows a preview of the currently active node. At the top is a shaded header with static text showing the current node's label, and beneath that, static text showing the node's absolute path from the root for enhanced context. To the left of the labels is two buttons. The first button, when clicked, will go back to the previously active node in this instance of the simulation run. If there is no previous node available, this button is disabled. The second button, when clicked, will go back to the parent of the currently active node. Rightmost on the shaded header is a section for Recordings (see 'Recordings' in this section). The feature is unrelated to the preview, but is a convenient location to place it. The remainder of the panel is split vertically. The right side contains

static checkboxes for the node states and static text for the number of activations. Below this is a non-mutable attributes table. The left side contains a listing of all Scripts for the node in the form of a static combo box showing the script name followed by a short preview of the contents of the script. If no script is assigned, the code preview is omitted. The panels are scrollable in the event they cannot show all content.

Variables Panel

The variable panel consists of a detail view table and a footer. The table has columns for the variable name, data type, and value. Only the value field can be edited. The content of the table is divided into two header sections: Internal and External. Internal variables are those that are created within the tool. External variables are function and variable references extracted from the project-specific extension of Logic imported into the tool. External functions and variables have a known name and data type and are simulated by letting the user treat them as if they were being fulfilled externally. For example, an external function `player_in_area()` returning a Boolean can be simulated by allowing the user to specify the value that is returned from the function as an External Variable. Rows in the table are usually banded but will be highlighted when they could impact the current node, its outputs, or its sibling triggers. As of right now, the highlight is uniform for all variables regardless of what they impact, but in the future it could be changed to have a different color per impact (i.e., a color for the current node, a color for outputs, and a color for triggers). Below the table is a footer containing a search panel which filters the variables based on their name. Those that do not match will be temporarily removed from the table until the search is altered or removed. On the right side of the footer is a checkbox labeled 'Bring related to top', which when active will sort the table such that all variables that impact the current node, its outputs, or its sibling triggers are brought to the top. To the right is a button labeled 'Restore default', which when clicked will restore all variables to their default state. The user is first prompted with a confirmation dialog for this action as it cannot be undone. At the far right of the footer is a button labeled 'Manage', a button labeled 'Save', and an editable combo box. Together, these manage states of all variables in the table. A state is a snapshot of the value that all variables in the table take. By selecting a state from the combo box, the state is applied to all variables present in the table after a confirmation dialog is presented to the user. Any further modifications to a state after loading do not automatically save. Instead, new states can be created by typing the desired name into the combo box and submitting. If the name is valid and unused, a new state is created with the current snapshot of the variables in the table. If the desired name already exists, then the user is prompted to overwrite and update the existing state. Clicking the save button has the same effect as submitting into the combo box, but is more explicit. Clicking the Manage button opens a popover containing a dropdown populated with all states and a single button to delete

the state currently displayed in the dropdown. Removing a state from the list does not reset the values in the table, even if the state being deleted is currently applied.

Entry, Outputs, and Triggers Panel

This panel has a cell-based list view split into three sections: entry, outputs, and triggers. If the current node is a non-leaf node and contains an entry point, then that entry is listed here. There can only be zero or one entry in this list. The outputs list all links outgoing from the current node. The triggers list two types of parallel node. The first is children of the current node that are parallel. The second is siblings of the current node that are parallel. This way, nodes that could trigger while siblings run are captured, as well as supporting situations where there are only parallel contents and no siblings within a non-leaf node. Each cell has an icon representing the type of node (Event, Sequence, etc.). To the left of the icon is a line indicating that it's an outgoing connection. Nodes with conditions have two slashes (//) intercepting the line to indicate visually that a condition is in place for the given output. This does not apply to triggers as they do not connect as outputs and they always have conditions. To the right of the icon is the node's label. To the far right of the cell is an icon indicating that the node can be followed. If the node's condition determines that the node cannot be followed, this indicating icon and the type icon become grayed out. Clicking on a cell, if the node's condition is satisfied and allows so, will follow the node and make it the new current node of the Simulation Mode window.

Recordings

Recordings are a feature of the testing environment for saving and replaying particular pathways through the story. The Recordings panel is broken up into two sections. The first section contains a Manage button, a Save button, and an editable combo box listing the states. This panel operates identically to the states section of the Variables Panel described above. The second section contains playback controls including beginning («), back (<), reset (undo), forward (>), and end (»). The panel is placed below the 'Entry, Outputs, and Triggers' panel. When the Simulation Mode window opens, no Recording is selected. If the user selects a Recording from the combo box, they will be alerted that this process will play back the Recording from its origin to its end, and reset all variables to their values at the start of the Recording. If the story structure or variables no longer permit the state from playing, a different prompt will show explaining to the user that the Recording is no longer valid. If during playback an obstacle is met (such as a followed node now being disabled whereas when the Recording was made it was enabled), the user is prompted that the Recording will now terminate as the original path cannot be followed. They are then free to continue as they see fit. When a Recording is being played back, the user can interact with the testing window as normal and make temporary changes, but if a link is followed with a state different from that expected from

the Recording, the user is queried whether or not they wish to continue, and if they do, playback is abandoned and the user continues testing manually. The playback controls are particular to Recordings being played back. The beginning («) button restarts the Recording. The back (<) button goes back one step in the Recording. The reset (undo symbol) button resets the current step if any modifications were made. The forward (>) button goes forward one step in the Recording. The end (») button goes to the very end of the Recording. If a stage is not available (such as not being able to go any further back or forward), then the corresponding buttons are grayed out. The reset button is always grayed out unless the participant makes modifications during active playback. At any point during testing (including during a Recording that has gone off track), the user can enter a name into the combo box and submit or press Save, which will result in the current session, up until the point of the save request, being stored. This includes traversal from the node that the testing environment was opened at to the current node at the time of saving, and the initial state of the variables upon starting the testing session. The initial state of the variables is the state upon the first traversal, which means that the user can load a variable state before making their first navigation to have the Recording work with a specific configuration, or instead of using a state, configure the variables manually instead.

APPENDIX D

Principle Validation

D.1 Survey Questions

As many of the survey questions reuse the same scale, they are enumerated once below.

Timespan	Effectiveness	Importance	Usefulness	Agreement
< 5 hours	Extremely Effective	Extremely Important	Extremely Useful	Strongly Agree
< 40 hours	Very Effective	Very Important	Very Useful	Agree
< 2 months	Moderately Effective	Moderately Important	Moderately Useful	Neutral
< 1 year	Slightly Effective	Slightly Important	Slightly Useful	Disagree
> 1 year	Not at all Effective	Not at all Important	Not at all Useful	Strongly Disagree

Demographics

Summary: Type is *Timespan.*
Question: *What is your estimated experience with creative writing?*

Summary: Type is *Timespan.*
Question: *What is your estimated experience with games storytelling?*

Summary: Type is *Timespan.*
Question: *What is your estimated experience with interactive/game narrative authoring tools?*

Summary: Type is free text entry.
Question: *Please list the authoring tools that you have used for creating interactive stories.*
Extra text: *If you use a proprietary authoring tool, please list it as "Proprietary" to avoid breaking any NDAs.*

Metaphor Testing

Summary: Targets point 1. Present in workflow 1. Type is *Effectiveness*.
Question: *How effectively do you think that the Canvas flowchart can be used visually to help understand the structure and flow of the story?*

Summary: Targets point 2. Present in workflows 1/2. Type is *Importance*.
Question: *How important do you think information provided by nodes (labels, descriptions, states, etc.) is in helping to understand the structure and flow of the story?*

Summary: Targets points 3/4/5. Present in workflow 1. Type is *Effectiveness*.
Question: *How effectively do you think that the clearness of connections between nodes helps with understanding the structure and flow of the story?*
Extra text: *Examples of clearness in the guides include obvious and thick lines, coloring of lines based on their target node type, and overlaying vertical bars on lines to identify them as dynamic.*
Notes: These three points were merged into a single question. Points 4 and 5 are specific examples of point 3, also contributing to clarity of connections, but differ enough in the authoring tool design to warrant their own supporting statement.

Summary: Targets points 6/7. Present in workflow 1. Type is *Effectiveness*.
Question: *How effectively do you think that visualizing paths between nodes by highlighting and animating their connecting lines helps with understanding the structure and flow of the story?*
Notes: These points have been merged as while each point demonstrates a separate use-case within the authoring tool design, they share an overarching concept of visualizing connections using augmentations of lines to better understand the structure.

Summary: Targets point 8. Present in workflow 1. Type is *Importance*.
Question: *How important do you think navigation around the Canvas flowchart (zoom, pan, fitting to selection, opening a node, etc.) is in helping to understand the structure and flow of the story?*
Extra text: *Consider if your story was large and can't fit on the screen all at once.*

Fast Track Testing

Summary: Targets point 1. Present in workflow 1. Type is *Importance*.
Question: *How important do you think being able to start testing from any point of the story (rather than just from the beginning every time) is to keep testing sessions focused and quick?*

Summary: Targets points 2/3. Present in workflow 1. Type is *Effectiveness.*
Question: *How effective do you think allowing authors to modify and manage the story state during testing (tweaking variables, creating and applying States) is at helping to keep testing sessions focused and quick?*
Notes: These points have been merged as although they differ enough in the authoring tool design to warrant separate supporting statements, they share a concept of making and managing tweaks while testing.

Summary: Targets point 4. Present in workflow 1. Type is *Effectiveness.*
Question: *How effective do you think providing the ability to record and play back testing sessions is in helping to keep testing sessions focused and quick?*

Structure

Summary: Targets point 1. Present in workflow 2. Type is *Effectiveness.*
Question: *How effectively do you think that the Canvas flowchart allows you to organize nodes together as you personally see fit? For example, by dragging nodes into visual groups based on distance.*

Summary: Targets point 2. Present in workflow 2. Type is *Effectiveness.*
Question: *How effectively do you think that grouping tools, such as Frames shown in the guides, help authors to personally organize their story content?*

Summary: Targets point 3. Present in workflow 2. Type is *Importance.*
Question: *How important do you think navigation around the Canvas flowchart (zoom, pan, fitting to selection, opening a node, etc.) is in supporting authors to personally organize their story contents?*
Extra text: *Consider if your story was large and can't fit on the screen all at once.*

Summary: Targets point 4. Present in workflow 2. Type is *Importance.*
Question: *How important do you think being able to easily and quickly identify the type and unique instance of a node is when organizing them into a personalized structure?*
Extra text: *Examples of this include the node color, icon, label, and any other additional attributes such as the description or states.*

Experimentation

For the next four questions, the phrase experimenting refers to authors making changes by exploring different structures and configurations. Examples of this include changing

the number of outputs a node has, changing where an existing connection goes, adding or removing nodes, and so on.

Summary: Targets points 1/4. Present in workflow 2. Type is *Importance*.
Question: *How important do you think having a clear view of your story nodes and being able to easily identify and understand connections between them is when experimenting?*
Notes: These points were merged together. The former states that spatial hypertext graphs are useful for experimenting due to the clarity of structure and connectedness, and the latter states that it's important to be able to identify the state of connections. These two points are stating that clarity of nodes (context) and the links between them (connections) are important to experimenting.

Summary: Targets point 2. Present in workflow 2. Type is *Importance*.
Question: *How important do you think being able to quickly identify a node's type and the unique instance of that node is when experimenting?*

Summary: Targets point 3. Present in workflow 2. Type is *Importance*.
Question: *How important do you think navigation around the Canvas flowchart (zoom, pan, fitting to selection, opening a node, etc.) is when experimenting?*
Extra text: *Consider if your story was large and can't fit on the screen all at once.*

Summary: Targets point 5. Present in workflow 2. Type is *Importance*.
Question: *How important do you think being able to quickly and easily create, modify, and remove connections between nodes is when experimenting?*

Branching

Summary: Targets point 1. Present in workflows 1/2. Type is *Usefulness*.
Question: *When creating or modifying branches, how useful is the extra information given by the Canvas flowchart (connections into and out of nodes, showing nearby nodes) in helping you to understand the context around where your branch is/will be placed in your story?*
Extra text: *A "branch" is when one node has more than one possible output, and the story may go either way. Visually, this may look as simple as one node connected to two or more others, or it could be complex like a Hub and Return system.*

Summary: Targets point 2. Present in workflows 1/2. Type is *Importance*.
Question: *When creating or modifying branches, how important is it to be able to easily create new connections between nodes and redirect existing connections between nodes?*

Contented Authoring

Summary: Targets point 1. Present in workflows 1/2. Type is *Agreement.*
Question: *I believe that I could use the prototype's features in a range of different ways to build specific structures (such as a branching dialogue tree).*
Notes: This is challenging to ask as it's something that participants did without knowing. This question answers the point as it asks whether, using the existing node and script setup, they can envision creating structures in different ways.

Summary: Targets point 2. Present in workflows 1/2. Type is *Agreement.*
Question: *By having multiple ways of achieving the same goal (e.g., being able to create a node from the Canvas right-click menu or alternatively by dragging from the toolbar icons), I am less likely to be caught in repetitive routines using the same methods.*

Seriousness Check

Summary: Present in the anonymous online survey only as a data validity method.
Question: *It would be very helpful if you could tell us at this point whether you have taken part seriously, so that we can use your answers for our scientific analysis, or whether you were just clicking through to take a look at the survey.*
Notes: Two options were available: *I have taken part seriously* or *I have just clicked through, please throw my data away.*

D.2 Interview Questions

General

Please can you explain to me what 'story structure' means to you?

Metaphor Testing

When you create interactive stories, how do you go about testing the story?

Tell me about any other ways that you check your structure.

Fast Track Testing

How do you handle testing large stories where it would take a long time to test from the start each time?

Structure

Tell me how you go about personal organization and management of your story content within authoring tools.

Experimentation

Tell me about the features and functionality you consider to be most important when editing your story within an authoring tool.

Branching

Tell me about the importance when specifically working with branching structures?

Contented Authoring

This principle does not have a topical question assigned as its underlying concept is inherently passive without the conscious knowledge of the user (i.e., users may not at all be aware that they become contented or break from a contented workflow) and describes a specific behavior, making it difficult to gather valid data without biasing and leading the user. The corresponding survey questions are able to approach this principle by providing questions in the form of related examples for each point that accurately reflect the underlying concepts without biasing or leading participants as would an active discussion of the topic. However, it is still possible to ask unstructured questions to an individual participant if their survey responses targeting this principle warrant investigation.

D.3 Coding Categories

Story Structure

High Level
Abstract descriptions such as the Hero's Journey, Three Act Structure, general ark, sequential events, and so on.

Low Level
Detailed description at a lower-level including nodes, links, branches, relationships, and so on.

Enhanced Lines

Thinks that animated lines between nodes help to understand connectivity
Participant mentions that animated lines between nodes help to better understand their connectivity.

Thinks that animated lines between nodes are helpful when zoomed out
Participant mentions that animated lines between nodes are particularly helpful when zoomed out.

Desire for animated lines to also highlight logical pathways
Participant mentions the desire for animated lines between nodes to include a visual representation of logic-driven pathways rather than just structural pathways (such as showing that a path between two nodes is dependent upon visiting other nodes elsewhere first).

Uses enhanced lines to help understand connectivity
Participant mentions that enhanced lines that are not animated (such as color or thickness) between nodes help to better understand their connectivity.

Adding animated lines is technologically taxing
Participant mentions how implementing animated lines in an authoring tool is taxing from a technological point of view.

Users complained when animated lines were removed
Participant mentions how removing animated lines caused authors to complain.

Graphs (Internal)

Uses a graph to visually follow structure
Participant mentions visually following a graph to understand the structure.

Disorderly graphs can be a hindrance
Participant speaks negatively about convoluted/complex/spaghetti graphs being a hindrance or otherwise confusing.

Uses a graph for a visual high-level structural overview
Participant mentions using the graph to visually understand an overview of the high-level structure of their story.

Uses attributes to enhance context
Participant mentions the use of attributes (labels, colors, shape, etc.) to help enhance context surrounding that element.

Uses a graph to visually identify logic

Participant mentions using a graph to visually identify logic in the structure such as conditions or use of variables, if exposed.

Uses a graph to visually inspect for errors

Participant mentions using a graph to visually inspect for errors such as missing or incorrect links between nodes. While this is inherently a part of visual testing, this node codes explicit mentions of this practice.

Desire to see a graph while editing

Participant mentions desire to be able to see a graph visualization while editing (in an interface that isn't graph-based but features a graph somewhere, or that isn't a graph-based interface but would like a supplementary graph present).

Uses a graph to visually gauge the density of content

Participant mentions the use of a graph to visually gauge the amount of content in their story.

Physical limitations of graphs make them challenging to properly use

Participant mentions encountered difficulties using graphs due to physical limitations of the graphs.

Graphs described as intuitive

Participant describes graph-based editing as intuitive.

Graphs described as overwhelming

Participant describes graph-based editing as overwhelming.

Nested hierarchy avoids oversized graphs

Participant mentions that the ability to nest nodes within each other avoids graphs that become clustered and oversized (i.e., it avoids too many nodes on one graph becoming difficult to handle).

Uses a graph to remind them of the structure when revisiting it

Participant mentions using a graph to remind them of the structure when revisiting their story after not having seen it for a while.

Using a visual graph helps to encourage nonlinearity

Participant mentions that working with visual graphs helps to encourage writing in a nonlinear way.

Lack of exposed attributes increases mental bandwidth understanding the story

Participant mentions how a lack of exposed attributes increases the mental bandwidth required to understand the story.

Thinks that graph-based systems are efficient
Participant mentions how they find graph-based systems to be efficient, particularly over other approaches.

Uses color to differentiate node function
Participant mentions the use of node color in graph-based systems to differentiate node function (e.g., nodes with the same color have the same function).

Node attributes may desync from their internal content
Participant mentions some form of potential desync about a node's actual behavior and its described behavior in its attributes (such as in a public-facing label).

Thinks that node attributes allow for idiosyncratic rules
Participant mentions how node attributes (shape, size, color, labels, etc.) allow for idiosyncratic rules to be adopted by the authors.

Using node attributes to represent entity state
Participant describes using an attribute of a node (such as color, border, shape) to represent the state of an entity.

Node Layout & Organization

Organizes nodes into groups using Frames (or similar)
Participant describes organizing nodes into groups in a graph based using Frames or similar.

Organizes nodes spatially into pseudo-groups
Participant describes organizing nodes into pseudo-groups in a graph based on spatial proximity without the assistance of grouping tools like Frames.

Organizes nodes in an arborescent structure
Participant describes laying out nodes in an arborescent (tree-like) structure in a graph.

Organizes nodes into nested groups
Participant mentions nesting of nodes within other nodes (or containers) to create hierarchical organization.

Organizes nodes matching their temporal position
Participant mentions organizing nodes based on their temporal positioning (e.g., a node that is later in time is spatially lower in a tree structure).

Uses grouping of nodes to indicate a shared status
Participant mentions using grouping nodes together to signal a status of those within the group (e.g., nodes in a group have their content finished or belong to a specific character).

Aligns nodes to a snap grid

Participant describes aligning nodes to a snap grid rather than being freely positioned.

Explicitly organizes nodes to mimic hypertext structures

Participant explicitly describes laying out nodes in a graph to spatially mimic hypertext structures (e.g., a split join being represented by a diamond-like layout).

Organizes node group spacing relative to content size

Participant mentions using the physical space between nodes on a graph as a metaphor for the size of the content. For example, a branch with only one node would be physically closer to its parent than one with 20 nodes.

Organizes nodes to mimic the environment in the story

Participant describes organizing nodes in such a way that they visually mimic the underlying environment structure described by the story (for example, if a location is north of another, the node would be above the other on a graph).

Thinks that node layout is an intrinsic element of the narrative itself

Participant describes that the layout out of nodes is inherently a part of the narrative experience rather than being exclusively for personal organization.

Using node spacing to represent character relationships

Participant mentions the use of spacing between nodes to represent character relationships.

Mapping (external)

Plans out the story externally and then recreates it in an authoring tool

Participant mentions planning of the story externally and then using that as a template to recreate the plan within an authoring tool.

Created an external map due to authoring tool limitations

Participant describes creating an external story map because the authoring tool had limitations that they could overcome by doing so outside of the tool.

Uses an external map to follow structure

Participant mentions following an external map to understand the structure.

Organizes external map elements into groups

Participant describes laying out elements of an external map into groups.

Organizes external map in an arborescent structure

Participant describes laying out an external map in an arborescent (tree-like) structure.

Uses an external map to track progress in the authoring tool
Participant mentions using an external map to track progress within an authoring tool (e.g., crossing off paths on the external map when tested in the authoring tool's implementation).

Annotates external map with narrative reminders
Participant describes annotating an external map (e.g., drawings) to supplement and remind them of the narrative.

Attempts to mirror the external map layout in a graph-based authoring tool
Participant describes that if using a graph-based authoring tool, they would try to mirror the external map structure directly.

Finds discrepancies between external maps and actually implementing it
Participant describes finding discrepancies between their planned external map and the actual resulting implementation in the authoring tool of their choice.

Attempts to mirror an authoring tool's graph design in an external map
Participant describes that they try to mirror an authoring tool's graph design in their external maps.

Testing

Testing from arbitrary points of the story
Participant mentions testing their story from an arbitrary point other than the very beginning.

Testing from the beginning of the story
Participant mentions testing their story from the very beginning.

Desire to tweak variables while testing
Participant mentions the desire to tweak variables while testing.

Testing story by sending to others
Participant describes testing the story by sending it (in whole or in part) to others.

Testing large stories by focusing on specific parts
Participant mentions testing a large story by primarily focusing on the specific parts of the story.

Works on small parts and immediately tests them
Participant describe working on small bits of the story and testing them before moving onto other parts.

Moves variables if they appear prior to desired testing location
Participant describes having variables needing to be set before the part they wish to test and

solving this by literally moving the variable temporarily to a nearby node (for example, just prior to a choice they wish to test) to test it.

Thinks the Recordings feature is useful for larger or more complex stories
Participant explicitly says how the Recordings feature would be helpful when dealing with larger or more complex stories. (This is particularly true for participants who didn't find it useful with smaller stories.)

Playtesting in a game engine
Participant mentions integrating and testing their content outside of the authoring tool environment and in an actual game engine.

Thinks the Recordings feature saves time
Participant mentions how the Recordings feature of the prototype saves time.

Uses built-in grammatical rules for testing
Participant describes using built-in grammatical rules for testing (e.g., syntax errors raised by the compilation process).

Wants a clear understanding of the story before starting to test it
Participant mentions that they want to have a clear understanding of their story (its contents and structure) before testing it.

Desire for a way to run automated validity checks
Participant mentions the desire for a way to quickly check for errors and validity (in text-based authoring tools, this could be a grammar checker, and in a node-based, system this could be checking for unused nodes or logic errors).

Desire to observe variables as they change throughout testing
Participant mentions that they would like to observe variables as they may or may not change throughout a testing session.

Thinks that different granularity of testing requires different testing methods
Participant mentions that testing stories at a high level requires (i.e., a large chunk) a different testing method to smaller parts (i.e., a small chunk).

Uses Recordings to store different potential player pathways
Participant mentions the use of the Recordings feature to store different pathways for different types of potential player, and using those to see what kind of experience each would have, and whether they work.

Uses syntax highlighting to visually check for errors
Participant describes using syntax highlighting (usually of a text-based authoring tool) to check for errors (such as seeing an uncaught quote as color doesn't change back).

Testing the game story by sending it to QA
Participant mentions the game story being tested by sending it to or in some way including a dedicated QA department.

Testing the reader's experience with the story
Participant mentions testing explicitly for the sake of the reader's experience with the story (narrative flow and the reader's enjoyment rather than structural correctness).

Finds it difficult to test the use of variables properly
Participant expresses difficulty in testing use of variables.

Would store Recordings for the purposes of historical preservation
Participant mentions that they would use the Recordings feature of the prototype to store recorded readings for the purpose of historical preservation and reproduction (that is, storing a specific reading of a hypertext so that it can be replayed or at least recovered much later).

Would use Recordings for debugging and exploration
Participant mentions that they would use the Recordings feature of the prototype to debug and explore the story's pathways.

Would use Recordings to recover previous playthroughs
Participant mentions that they would use the Recordings feature of the prototype to recover a previous reading for debugging or insight.

Uses an in-engine interactive debugging tool to test state variations
Participant mentions using an interactive debugging tool within an engine to test state variations on the story.

Uses built-in automated validation checks to check logic
Participant describes using a built-in automated validation checker to check story logic.

Uses in-engine save games to test varying story states
Participant mentions using in-engine save games to load a specific state of the game world which allows them to test that specific configuration with the story.

Uses OBS to record testing sessions for debugging or review
Participant mentions using OBS software to record testing sessions for debugging or review.

Using built-in testing loses context
Participant mentions the use of built-in testing having less context than testing the content within its real environment (such as testing a text-based dialogue rather than seeing it spoken by characters in the game).

Usually wants to test from the beginning when working with dialogue
Participant mentions how they usually want to test from the beginning when working with dialogue.

Considers revising of writing a form of testing
Participant describes the revising and editing of writing a form of testing.

Runs automated tests for preservation of old hypertexts
Participant mentions running automated tests on older hypertexts to make sure that new software updates do not break them.

Uses external scripts to automate testing
Participant describes writing custom scripts outside of an authoring tool to automate the testing process.

Would use Recordings feature to validate the current story revision
Participant mentions the use of existing Recordings against the current revision of the story to see where they may not function as (previously) expected.

General Authoring Workflow

Ease of connecting content is important
Participant mentions how connecting content together (in any form including text and nodes) is important or essential to them.

Ease of creating content is important
Participant mentions how creation of content (in any form including text and nodes) is important or essential to them.

Works out high-level structures before implementing details
Participant describes figuring out high-level structures of the story before filling in the details.

Inadequate authoring tool functionality limited them creatively
Participant mentions how the authoring tool was limiting them creatively (e.g., distracting, cumbersome, blocking) due to inadequate features or functionality.

Being visual and graph-based is important
Participant mentions how the authoring tool being visual and being chiefly graph-based (i.e., nodes and lines) is important or essential to them.

Chose an authoring tool based on its genre-specific capabilities
Participant describes choosing an authoring tool to use based on its genre-specific features that match the type of story they want to tell.

Ease of content organization is important

Participant mentions how organizing content (in any form including text and nodes) is important or essential to them.

Plans out the narrative in a large Word document

Participant mentions planning out the narrative in a single gigantic Word document.

Tends to work on a detailed section before moving on

Participant mentions working on a small detailed part of the story and working on it until happy (e.g., at least a first pass of the content) before moving on.

Thinks that connections are hard to follow text-based authoring tools

Participant mentions how when using text-based authoring tools, connections are hard to follow or understand.

Uses writing structures or hypertext patterns intentionally

Participant mentions conscious use of writing structures or hypertext patterns in the authoring process.

Deferring complex tasks can lead to forgetfulness

The participant describes leaving complex tasks for later, but accidentally forgetting about them as a result.

Desire for guidance when writing scripts

Participant mentions the desire for assistance or guidance of some kind when writing scripts (particularly in authoring tools that are not text-based).

Desire for the ability to annotate story structures within a tool

Participant mentions the desire to annotate the story structure within an authoring tool (for purposes such as personal recording or tracking which ways you've been or where you've gotten up to).

Desire to know which variables go with which branch

Participant mentions the desire for some way of knowing which variables are used by which branches, or similarly, which variables a specific branch uses.

Ease of access to external media is important

Participant mentions that it is important to be able to access external media from within the authoring tool.

Ease of navigation is important

Participant mentions how navigation around content is important or essential to them.

Ease of use is important
Participant explicitly mentions how ease of use of a tool is important.

Finds tools with code familiar due to programming experience
Participant mentions authoring tools that include programming (like Twine using JavaScript) feeling more at home than other solutions due to their experience with programming.

Indecisive authoring methods
Participant describes bouncing between methods or being otherwise inconsistent.

Likes to be able to add custom functionality to script-based languages
Participant describes writing their own custom functions in text-based editors like Inform or Inky.

Thinks integration of the story into a game engine is troublesome
Participant discusses how integrating their story into a game was met with challenge.

Thinks that natural language tools are like writing in broken English
Participant expresses concern at authoring tools that use natural language as they feel like writing in broken English.

Thinks that varying methods of the same task help combat ADHD
Participant mentions how having varying methods of achieving the same task aid in combatting ADHD.

Unclear authoring state causing slowdowns
Participant mentions an issue with the authoring tool's state presentation causing slowdowns in authoring (such as unclear links making the user pause to figure them out).

Uses autocomplete to speed up making connections
Participant mentions using an autocomplete function to speed up creation of connections.

Uses syntax highlighting to understand structure
Participant describes using syntax highlighting (usually of a text-based authoring tool) to understand story structure via text being colored in blocks.

Breaks narrative down into iteratively smaller chunks when planning
Participant mentions breaking down the narrative from the largest scale becoming iteratively finer in detail as granularity becomes smaller (such as overall structure > levels > missions > quests).

Creates mood boards to assist in ideation
Participant mentions the creation and use of mood boards to assist in generating ideas for the story.

Narrative content can be impacted by external resources
Participant mentions how narrative content can be impacted by external resources often out of the control of the writers.

Uses internal tracking/documentation over dedicated story authoring tools
Participant mentions the use of tracking and documentation over the use of dedicated story authoring tools.

Works on narratives with trivial levels of interactivity
Participant mentions how their typical work doesn't involve complex interactivity.

Being able to visualize links between content is important
Participant mentions that it's important to them to be able to visualize links between content (i.e., as opposed to just knowing that they are defined, being able to see them onscreen).

Likes being able to drop into programming for lower-level control
Participant mentions that they like being able to write code in their story to some capacity for lower-level control over the story.

Likes inclusion of high-level programming languages
Participant mentions that they like the inclusion of high-level programming languages in authoring tools.

Ability for content attributes is important
Participant mentions how changing attributes of content (e.g., the color or label of a node) is important or essential to them.

Uses a word processor to write a high-level version of the story
Participant describes using a word processor to write a high-level version of the story.

Uncertain of where the story is going during the writing process
Participant mentions that they do not always know where the story will go while writing it (that is, even with a plan, some things are still unknown).

Navigation

Thinks navigation is more important for large projects
Participant mentions that navigation is more important when projects are larger.

Large navigation jumps cause confusion
Participant describes large navigation jumps when editing (that is, on a larger scale rather than local editing) makes them feel overwhelmed or confused.

General Layout & Organization

Uses colors for content organization
Participant mentions the use of colors to represent some form of personal organization.

Organizes Genarrator slides into a custom order
For participants that had used Genarrator, mention of organization of the slides into a custom order that meant something to them as opposed to being random or in the default order.

Separates text-based narratives into different files
Participant describes separating text-based narratives into separate files for personal organization.

Uses comments in Word documents for story annotation
Participant mentions using comments within Word documents (as in the comment feature, not just inline text) for annotating the story in some way.

Uses styles in Word documents to separate
Participant mentions the use of styles in Word processors to separate the meaning of content from each other.

Notes

Notes for chapter 1

1. For example, *Ulysses* by James Joyce, published in 1922, purposefully appears unstructured and chaotic, to this day remaining a staple of nonlinearity in traditional fiction.

Notes for chapter 2

2. Instead of sjuzet, Mieke Bal uses 'story', whereas others described in this thesis have used the same word to describe fabula. This highlights the challenges of translating phrases, particularly in subjects so nuanced as narratology, between at least Russian, French, and English. Neither translation is incorrect.

3. In his book *Literary Machines* [161], Nelson describes *transclusion* as the inclusion of a part or all of an electronic document into one or more other documents via hypertext reference. He further introduces *transdelivery* to specifically refer to the inclusion of content from a different location, and *transquotation* for explicit quotation that remains connected to its origins.

4. As explained by Ben Shneiderman and Catherine Plaisant in their article "Hypertext Research: The Development of HyperTIES" available at `http://cs.umd.edu/hcil/hyperties` as of 4 February 2022.

5. This is mentioned by Shelly Jackson in her own explanation and traversal of *Patchwork Girl* as part of the *Pathfinders* project by the *Electronic Literature Lab.* A video of the traversal is available on YouTube (`https://youtube.com/watch?v=ZHUR6phu0rc`) as of 4 February 2022.

6. *InvisiClues* was a series of books that accompanied *Infocom* games such as *Zork I.* They contained spoiler-free hints made possible by using invisible ink that was revealed using an *InvisiClue* pen. In addition to clues and answers, they provided maps for users to gather their bearings. This can be seen in some ways as a physical attempt to mitigate the *disorientation* problem of being lost in a hypertext.

7. See "Façade: An Experiment in Building a Fully-Realized Interactive Drama" presented by Michael Mateas and Andrew Stern at Game Developers Conference 2003.

8. *Radiant Quests* are scattered throughout the base game as a way of enhancing the variance and longevity of the game. Further *Radiant Quests* can be authored in the game's *Creation Kit* modding tools.

9. As described by Adams in his 1999 article on *Gamasutra* titled "The Designer's Notebook: Three Problems for Interactive Storytellers" available at `https://www.gamasutra.com/view/feature/131821/the_designers_notebook.php` as of 4 February 2022.

10. See "Formal Design Tools: Emergent Complexity, Emergent Narrative" presented by Marc Leblanc at Game Developers Conference 2000.

11. Presented as the *Game Narrative Triangle* by Fraser Allison in his blog post available at http://redkingsdream.com/2010/07/the-game-narrative-triangle via Archive.org's WaybackMachine.

12. *Shelley's Heart* was a locative narrative story created and executed in Bournemouth, UK, as part of a celebration of Frankenstein's 200th anniversary. Information about the project is available at https://shelleysheart.com as of 4 February 2022.

13. Quick-time events are short-lived sequences presented to a player in a game in which they must respond to some given prompt requesting player input. The result is typically binary in that the player either achieves the requested prompt or does not.

14. Carson's article from 2000 on *Gamasutra* titled "Environmental Storytelling: Creating Immersive 3D Worlds Using Lessons Learned from the Theme Park Industry" lays the foundations for environmental storytelling in games based on his own experience with theme park design. Available at https://www.gamasutra.com/view/feature/131594/environmental_storytelling.php as of 4 February 2022.

15. For example, at the Game Developers Conference 2010, there were two talks on environmental storytelling ("Environmental Narrative: Your World is Your Story." by Richard Rouse III, and "What Happened Here? Environmental Storytelling." by Matthias Worch and Harvey Smith). Moreover, since 2010, the same conference has hosted a dedicated *Game Narrative Summit* in which environmental storytelling techniques periodically arise.

16. An article by *Polygon* outlined the mental stress that *Missile Command* had on its designer even several years after its release, ultimately caused by the narrative driven by the game's mechanics. Available at https://polygon.com/features/2013/8/15/4528228/missile-command-dave-theurer as of 4 February 2022.

17. In this article, Rohrer explains the motivation behind the game in relation to his own real life. He follows by describing various mechanics from the game and documenting how he intended them to be a metaphor for various thematic topics, although he makes it clear that there is no incorrect interpretation of the metaphors. Available at http://hcsoftware.sourceforge.net/passage/statement.html as of 4 February 2022.

18. Specifically, Jason Rohrer questioning Rod Humble about his demonstrated use of game mechanics as a metaphor in *The Marriage*, directly querying as to why he chose to use mechanics as metaphor in his design. The phrasing suggests that either the two were already familiar with the term, meaning it has an older origin, or that Rohrer had used 'air quotes' when coming up with the term, but I cannot discern which is true. Available at http://northcountrynotes.org/jason-rohrer/arthouseGames/seedBlogs.php?action=display_post&post_id=jcr13_1175084349_0 as of 4 February 2022.

19. For example, in a 2008 interview with *IndieCade* curator Sam Roberts (available at https://destructoid.com/indiecade-night-journey-braid-and-the-importance-of-indie-games as of 4 February 2022), he highlights how Jonathan Blow's *Braid* makes extensive use of mechanics as metaphor and shares his disappointment on how this technique is lacking in other contemporary games. We also see brief discussion of mechanics as metaphor in Emily Short's reflective article on a talk by Jonathan Blow (available at https://gamasutra.com/view/news/116287/Analysis_Challenge_Game_Design_Conflicts_Storytelling.php as of 4 February 2022) where she likens Blow's concept of 'dynamic meaning' to mechanics as metaphor.

20. "Mechanics as Metaphor — I: How Gameplay Itself Tells a Story" available at https://youtu.be/4QwcI4iQt2Y as of 4 February 2022, and "Mechanics as Metaphor — II: Creating Narrative Depth" available at https://youtu.be/pP_qNm-96Dc as of 4 February 2022.

21. See, for example, Gregory Weir's 2008 article published on *Gamasutra* titled "Opinion: Grim Fandango and Diegesis in Games" (available at https://www.gamasutra.com/view/news/112122/Opinion_Grim_Fandango_And_Diegesis_In_Games.php as of 4 February 2022) and Marcus Andrews' 2010 article on *Gamasutra* titled "Game UI Discoveries: What Players Want" (available at https://www.gamasutra.com/view/feature/4286/game_ui_discoveries_what_players_want.php as of 4 February 2022).

22. The *Authoring for Interactive Storytelling Workshop* has run annually alongside the *International Conference on Interactive Digital Storytelling* for several years. The 2020 proceedings can be found online at http://narrativeandplay.org/ais/2020 as of 4 February 2022.

23. Mentioned in an interview with John McDaid published in the *Pathfinders* book available at http://dtc-wsuv.org/wp/pathfinders/authors-works/john-mcdaid-uncle-buddys-phantom-funhouse as of 4 February 2022.

24. As reported by GameSpot in the "15 Most Influential Games of All Time" article available at http://gamespot.com/gamespot/features/pc/most_influential/p14.html via Archive.org's WaybackMachine.

25. As listed under the "Programs authored (or executed) with this title" section at https://spectrumcomputing.co.uk/index.php?cat=96&id=6857 as of 4 February 2022.

26. As listed under the "Programs authored (or executed) with this title" section at https://spectrumcomputing.co.uk/index.php?cat=96&id=6825 as of 4 February 2022.

27. Published in the *Society for the Promotion of Adventure Games*, issue 44, available at http://spagmag.org/archives/backissues/SPAG44 as of 4 February 2022.

28. As documented in *The Hugo Book* available at http://ifarchive.org/if-archive/programming/hugo/manuals/hugo_book.pdf as of 4 February 2022.

29. As mentioned by Norman in an interview as part of *Adaptive Path's UX Week 2008 Conference* available at https://adaptivepath.org/ideas/e000862 via Archive.org's WaybackMachine.

30. See "Tools: Making a Better Game" presented by Jim Brown at Game Developers Conference 2010.

31. The original article appears lost to time, but is available in a Web format at http://vm.ibm.com/devpages/jelliott/evrrt.html via Archive.org's WaybackMachine in their 08 December 2015 capture.

32. This article is available on the NNGroup website at https://nngroup.com/articles/end-of-web-design as of 4 February 2022.

33. In cognitive psychology, 'chunking' refers to individual pieces of information that are grouped together into a more meaningful whole. That is, a collection of basic familiar bits of information are combined to be stored as a single item rather than several single pieces of information.

34. For example, the *Nielsen Norman Group*, founded by Jakob Nielsen and Don Norman, offers an 'Expert Review' service, part of which uses Nielsen's own *Ten Usability Heuristics*.

35. For example, as part of the submission preparation for both mobile and desktop platforms being deployed on *Apple's App Store*, it is the responsibility of the developer to "make sure your apps follow [Apple's Design Guidelines] before you submit them", which is then verified by *Apple* as a heuristic analysis during the review process. Applications that violate these guidelines may be rejected.

36. Nielsen discusses this in his own blog post which presents all of the heuristics, descriptions and examples, academic references, and personal reflection on the heuristics. Available at https://nngroup.com/articles/ten-usability-heuristics as of 4 February 2022.

37. The complete directory of guidelines is available online at http://hcibib.org/sam/contents.html as of 4 February 2022.

38. The *Usability Body of Knowledge*, a part of the *User Experience Professionals' Association*, explains that contextual inquiry provides data that is more realistic than synthetic due to users being interviewed in their own environments instead of in a lab. Available at http://usabilitybok.org/contextual-inquiry as of 4 February 2022.

39. As explained on *usability.gov*, which is operated by the *U.S. General Services Administration*, it is possible to mix the two methods together despite their differences. Available at https://usability.gov/how-to-and-tools/methods/contextual-interview.html as of 4 February 2022.

40. In the field of user experience, "you are not the user" is a common mantra and widely accepted axiom based on the *False-Consensus Effect* [180]. The effect refers to the tendency to believe that others will behave the same as you in a similar context. As a developer creating software, this means projecting your own understanding of your own design onto actual target users, which is almost always not accurate.

Notes for chapter 3

41. Quoted directly from *Inform's* official website available at http://inform7.com as of 4 February 2022.

42. *Spacewar!*, released in 1962 for the PDP-1, is a multiplayer space shooter which is believed to be the first game to widely spread. *The Sumerian Game*, released in 1964 for the IBM 7090, is an early text-based strategy and resource management game comparable to gamebooks like *Choose Your Own Adventure* where the player navigates a story by making explicit choices.

43. Statistics retrieved from *The Elder Scrolls Wiki* hosted by *Fandom* available at https://elderscrolls.fandom.com/wiki/Books_(Skyrim) as of 4 February 2022.

Notes for chapter 4

44. Also known as *bit rot* as defined by *The Jargon File v4.4.8* available at http://catb.org/jargon/index.html as of 4 February 2022.

45. Presented as part of the *Redcap* project available at http://redcap.interactive-storytelling.de/authoring-tools/emo-emma as of 4 February 2022.

46. 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.

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

48. *Articy* boasts several prominent game studios among its users including *THQ Nordic*, *Limbic Entertainment*, and *ZA/UM* as shown at https://www.articy.com/en/showcases as of 4 February 2022.

49. As of 10 January 2021, the *Interactive Fiction Database* (available at https://ifdb.tads.org as of 4 February 2022) shows more than 5000 stories written using a version of *Inform*.

50. See, for example, the *Redcap* project, where Little Red Riding Hood was used extensively to demonstrate several authoring tools. Available at http://redcap.interactive-storytelling.de as of 4 February 2022.

Notes for chapter 5

51. Presented as part of the article "UX Research Cheat Sheet" available at https://nngroup.com/articles/ux-research-cheat-sheet as of 4 February 2022.

52. Presented as part of the article "Personas Make Users Memorable for Product Team Members" available at https://nngroup.com/articles/persona as of 4 February 2022.

53. See "Designs on Writing: Improving the Game Writing Process" hosted by Tom Abernathy, Steve Jaros, Dalan Musson, and Mac Walters at Game Developers Conference 2008.

54. See "Balancing Act: Narrative and Mission Design in Assassin's Creed III" presented by Philippe Bergeron and Corey May at the Game Narrative Summit of Game Developers Conference 2013.

55. *Microsoft Digital Image* is a discontinued image editing program succeeded by newer *Microsoft* services.

Notes for chapter 6

56. As reported on *Twine's* latest GitHub repository available at https://github.com/klembot/twinejs as of 15 August 2021. It's also worth noting that this is only one component of the *Twine* platform — the Electron browser port — as the underlying language models and other variants of *Twine* are in separate repositories, each with their own comparable situation.

57. In the context of surveys, satisficing is a concept where participant answers become biased or otherwise less useful as a manifest of attempting to lowering cognitive burden. For example, a participant may rush an online survey, select random responses, choose socially desired responses, or other similar patterns for a variety of reasons typically associated with lowering cognitive load.

58. *Craig* is an open-source bot (available at https://github.com/Yahweasel/craig as of 4 February 2022) for *Discord* which allows for recording of conversations. For this study, I used the official publicly available build of *Craig* (available at https://craig.chat as of 4 February 2022).

59. Specifically, a form of survey satisficing referred to as non-differentiation or straight-lining that may occur when a series of questions asks for ratings on the same response scale.