UNIVERSITY OF CALIFORNIA
SANTA CRUZ

DEVELOPING A COMPUTATIONAL THEATER

A dissertation submitted in partial satisfaction of the
requirements for the degree of

Doctor of Philosophy

in

COMPUTATIONAL MEDIA

by

Nic Junius

September 2024

The Dissertation of Nic Junius
is approved:

Prof. Elin Carstensdottir, Chair

Prof. Michael Mateas

Prof. Michael Chemers

Prof. Chris Martens

Peter Biehl
Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

List of Tables

List of Algorithms

Abstract

Developing a Computational Theater

by

Nic Junius

When we tell stories, we play with them. Theme, character, setting, and plot are some of the main sites of this style of play. Fresh perspectives on these elements allow them to be combined and interpreted and new ways, inspiring further play. Interactive narrative in computation has explored this idea from a number of perspectives, often running into the problem of how to invite and incorporate a naive audience member into a story. In this dissertation, I discuss my work re-examining the assumptions of interactive storytelling, drawing from theatrical practices to create AI systems of expressive performance, their use and impact on narrative design, and ways of supporting the development of experiences built around the idea of players as performers. This combination of design theory, practice, and evaluation blend together into *computational theater*: a new approach to creating interactive narratives and AI driven character behavior rooted in the aesthetics of theatrical performance. Through this approach, I illustrate how performance expands the ways a player can interact with and influence narratives through play with AI controlled characters as well as the resulting implications for the development of AI-based interfaces and simulations.

To the shardfolk and the coven

Acknowledgments

I would not have been able to do the research I present in this dissertation or get through my now three degrees and seven years of graduate school without support from so many people. More people than I can possibly name touched this work in large and small ways and gave me support through the hardest times. I thank all of you and want to name those who I owe special thanks to here.

The cats in the coven gave me the strongest support I could ask for, brightened every one of my days, and helped me spread my ideas far further than I ever thought possible. I owe each of you more than I can ever express in words. Freyja Katra Erlingsdottir is my sunshine and a designer I look up to so much and I'm so excited to make more things with her. Chiaki Hirai has kept me laughing about art and our home town and I will one day do the media criticism for you we keep talking about. Felicia Balthar rearranged her sleeping schedule because of me and has listened to more than I could ever hope to return.

Everyone in the Seabright Camerata is a friend I deeply treasure and have loved tossing ideas around with. Here's to many many more future discussions! Max Kreminski and Isaac Karth were my housemates through all of the degrees and domicile related disasters imaginable, frequent collaborators, and somehow we still haven't completed the publication graph yet. Kate Compton who laid the foundations for the Camerata with opera and whose work on Tracery inspired much of how I approached building Puppitor. Aaron Reed was the best game master I could ever ask for and whose

footsteps I have accidentally followed in between writing interactive fiction, working in the Expressive Intelligence Studio, and making a TTRPG my dissertation year. Jason Grinblat talked golems with me a few times between emergent narrative conversations and now it's all golems all the way down. Cat Manning was the source of so many great narrative design conversations and Shakespeare opinions. Jacob Garbe and his various ravens always make me want to add a bit of magic to whatever I'm working on. Melanie Dickinson's curiosity and excitement about storytelling and everything else we've ever talked about is an inspiration to me. Jasmine Otto always showed me a new perspective on my work and our shared interests. Barrett Anderson never fails to include me even when all I'm there for with the board games is the vibes. Each of you contributed to everything in here more than I could ever thank you for.

I could not have made *Tracks in Snow* without the help of so many collaborators, including a number of people already mentioned. Ronald Murray did amazing character and key art, the game would not exist as it is without you. C. Marshall did the best character designs, concept art, and background renders I could ask for. Isaac Karth's 3D models all made the world feel bigger. Angeleen Tan did lovely art of girls kissing for some very important CGs. Tamara Duplantis gave me an amazing set of music to work with and mix. Sophie Martin made fantastic tracks to bookend the entire game. Freyja Katra Erlingsdottir has kept my words from running too far away. Max Kreminski saved me from writing the worst parser ever. And of course, *Tracks* wouldn't exist without all the work done by Tom Rothamel and all the other Ren'Py contributors. I will finish the script now that the dissertation is complete!

This dissertation would not exist at all without my committee members. My advisor Elin Carstensdottir always pushed me to go bigger and provided me opportunities to do just that. Michael Mateas has been on every one of my committees and his work has been a huge part of the foundation of my own. Michael "Doc" Chemers is one of my longest time collaborators and far more responsible for everything in here than he realizes or will likely willingly claim credit for. Chris Martens helped me refine the way I framed my work and while I couldn't do everything we talked about, that just means there's even more we could collaborate on in the future.

I am so thankful for all the friends I've made in Computational Media: Kyle Gonzalez, Henry Zhou, Jared Pettitt, Cassandra Ravenbrook, Andrew Dunne, Bre Baltaxe-Admony, Asiiah Song, Alex Calderwood, Shi Johnson-Bey, Jack Kelly, Sam Conde, Sol Choi, James Fey, Alison Crosby, Nathann Latimore, Batu Aytemiz, and everyone else I don't have the space to name; you all made the last seven years so much more fun and fruitful. Special thanks to my colleagues in the Interaction Dynamics Lab Bjarke Larson, Katy Grasse, Shweta Sisodiya, Kevin Weatherwax, Yi Xie, Ryan Cai, Adam Carter, Angeleen Tan, and everyone else who has passed through and will join in the future.

I also want to thank my advisors from previous degrees Noah Wardrip-Fruin and Elizabeth Swensen whose guidance and feedback was foundational to my research journey. Jim Bierman was the best mentor a playwright could ask for. Thank you to all the members of the EIS diaspora, but especially Anne Sullivan, Gillian Smith, James Ryan, Ben Samuel, Josh McCoy, Stacey Mason, Stella Mazeika, and Adam Smith who

all paved the way for my research in one way or another. All my compatriots from DANM, but especially Michael Becker, Evie Chang, Rey Cordova, Dayna Diamond, Kavi Duvvoori, Jordan Magnuson, Zoe Sandoval, Ben Spalding, Squinky, and Patrick Stefaniak helped me grow as an artist. All the friends I've made and shared ideas with in the wider games and AI communities, especially Mike Cook, Florence Smith Nichols, Younes Rabii, Ollie Withington, Rogelio Cardona-Rivera, Markus Eger, Stephen Ware, Erica Kleinman, Nathan Partlan, Lara Martin, Matthew Guzdial, Emily Halina, Kristen Yu, Arunpreet Sandhu, Ian Horswill, Allison Parrish, and Emily Short.

All of the comrades I gained across the 2019-2020 wildcat strike, 2022 contract strike, and the 2024 unfair labor practices strike, thank you for helping me be a part of something much bigger than my own research and for fighting to make a more just university.

Thank you to my friends far and wide—Noelle, Ritty, Tori, and Holly—who have reminded me that I exist outside of my work and research.

And finally, thank you to my parents Anne and Andrew for giving me all the support through my seven years of graduate school, three degrees, and now moving halfway around the world.

Chapter 1

Introduction

Theater is a human process devoted to telling and retelling stories. The elements of this process come together in what we commonly refer to as a play. This single word encompasses the methods theater uses to put on a show and to work with narrative. As a performing art, theater is *playful*. From acting to directing to design to puppetry to playwriting to dramaturgy, these disciplines each approach the problem of a story by playing with it. They tease out thematic connections, find new perspectives, and seek new interpretations through the act of producing a play. Similarly games and interactive narrative those interacting with them players. It is a player's role to, again, explore the circumstances they find themselves in, to tease apart the connective tissue, push against its boundaries, have a dialogue with the work, and, of course, play.

I am not the first person to argue for the parallels between computing and theater, the entire field of interactive drama has existed since the 1980s [154]. Yet, I didn't title this dissertation in a way that relates to interactive drama. I don't view

myself as working in that tradition, with its desire to make the structure of the narrative the primary site of interaction. One of my purposes with this dissertation is to re-examine the assumptions of interactive drama and forge an alternate path of combining computation and theater. I therefore frame my work around *computational theater*, or the practice of creating systems and experiences to support the *playful interaction* with stories through performance.

Playful interaction is often associated with physical, embodied play [23,24,264] of some kind. It has been applied to storytelling itself, though with a more literary-focused, co-creative approach [143] than what I am describing in this dissertation. My use of the term here leans closer to the literary, co-creative approach as it too directly relates to the process of storytelling. To frame my dissertation, I view playful interaction as the act of engaging with a story and experimenting with its portrayal, details, and thematic elements.

These definitions bring me to the goal of my dissertation and, more broadly, the goal of computational theater, succinctly described by Steve Tillis [265]:

> *My virtual Hamlet is not itself of nature; neither, in its own way, would a stop-action Hamlet. But both, just like a marionette Hamlet or, for that matter, a living Hamlet, serve the same essential function (albeit by differing means): not so much the holding up of a mirror to nature as* ***the opening up of a window for human artistry.***

In other words, to use computation to open up a window for human artistry through play. Now, what exactly do I mean by this statement? Performance is an art form tightly linked with play. It is an art form about making things come alive [29, 123]. It is an art form which creates fleeting beauty, sometimes likened to a flower [223].

Performance is also an art form deeply connected to interpretation through play. In particular, performance can find new perspectives on narrative details only possible through communal play and then give them life on stage [19, 29, 158]. Computation has opened up even more possibilities for play and interpretation [91, 193, 273]. Through computational systems, we can not only invite play, we can hold conversations with our players through these systems [273], inviting a dynamic relationship to these systems and experiences in the process [193]. With computation, we can support players in their exploration of experiences, stories, and systems by making complexity approachable [273] and through shared authorship [143]. As performance based and interactive mediums, these two art forms share the need for someone to add their voice to the artifact, be it a play script or an executable file, to tell a story.

A play script becomes a set of constraints on performance, yet this doesn't mean there is no creativity to be found, quite the contrary even. Across performance practices, there is a common thread of finding the new in the old [223], of finding power in juxtaposition [19], of keeping a story alive by making even minute changes [158]. These aspects of creativity do not rely on being able to restructure a story or completely change dialogue. Instead, they revel in all the ways presentation can completely change the meaning of a line, the audience's relationship to a character, or the thematic focus of the entire narrative.

It is with all these theatrical concepts in mind I return to the other half of this dissertation, computing. More centrally, what specifically is it I do with computational theater? I build systems and interactive narratives inspired the qualities and aesthetics

of performance. These computational artifacts are, of course, an expression of the parallels I see between theatrical storytelling and interactive storytelling as well as proof of the efficacy of computational theater. Additionally, these artifacts encompass both AI systems and problem domains, often in a blurry way. Puppitor, the system much of this dissertation focuses on, is both a domain and interface, not an AI system unto itself. The AI systems I discuss are inexorably linked to Puppitor and minimal to the point of not being named. They are, however, important examples of my philosophy that a system can focus on doing one thing well without first having to model the universe and succeed at its appointed task.

More broadly, these artifacts and the design theories they birthed expand the concept of interacting with a narrative. Playing with the structure of a story is only one approach and there is the potential for so many other perspectives on the matter. Storytelling in computation is both a technical and aesthetic practice. As such, there is a need to re-examine assumptions and experiment. You might even find something new and beautiful.

This perspective brings me to the four research questions of my dissertation in the area of narrative design, system design, and system evaluation:

- **RQ1:** How is theatrical storytelling applicable to computation and narrative design?
- **RQ2:** How do we translate theatrical practice into a computational model of character acting?

- **RQ3:** How do we create a narrative experience using an acting system as the interaction paradigm?

- **RQ4:** How do we computationally analyze and expose a system's acting choices and its effects on the performance of a character in an interactive narrative context?

Answers to each of these questions support the overall contribution of this dissertation in computational theater. I structure this dissertation in four parts. The first is Chapter 2, a characterization of the relationship between theater and computation. The second is Chapters 3 to 5, a discussion of narrative design which supports interpretation as an aesthetic of play and answering RQ1. Third, Chapters 6 and 7 provide a description of the narrative and system design of the dramatic visual novel *Tracks in Snow* and acting system Puppitor, answering RQ2 and RQ3. Finally fourth, Chapters 8 to 10 are a discussion of the relationship between technical and aesthetic elements of AI systems and describe the ergonomics of authoring with Puppitor indifferent development environments, answering RQ4.

Chapter 2 is the examination of the academic relationship between theater and computation. My goal is to characterize the evolution of the assumptions of work in that space while positioning computational theater as a way to create a new relationship between the two fields, parallel to the existing research. I cover the development of interactive drama as a field, the difference in its priorities and theatrical practice, and the ways theater and computation and theater have been combined outside of interactive

drama.

Theatricality is one of my major contributions supporting computational theater, covered in Chapters 3, 4, and 5. These chapters seek to expose the ways narrative design encourages critical engagement with a story. In Chapter 3, I define theatricality as a property of storytelling which uses reinterpretation as a core aesthetic of engagement. This aesthetic is found in both the theatrical production process and cyclical game narratives. Chapter 4 connects theatricality to the concept of story volumes, which argues that narratives can describe an entire possibility space of story lines. Importantly, this chapter emphasizes that theatricality and story volumes, while first applied to highly systemic games, are a property of design and storytelling approach rather than a purely systemic concept. Chapter 5 broadens theatricality from its focus on individual, singleplayer experiences to community driven modes of interpretation. This discussion rounds out the lens of theatricality as a view of narrative design and interaction applicable to many distinct approaches to storytelling.

I describe my narrative and system design practices as examples of works of computational theater in Chapters 6 and 7. Chapter 6 focuses on the high-level narrative design lessons learned from *Tracks in Snow*. In it, I posit *narrative frame data* as a particular approach to implementing performance and theatricality in a game with the express goal of facilitating playful interaction with stories. Additionally, I call out the implications of interaction design focused around performance and discuss the ways in which a game's structure can be used to support players' critical engagement with a story. Chapter 7 then details the design and architecture of Puppitor as a

system to enable cooperative expressive performance between human and AI controlled characters. I also describe some of Puppitor's integration with *Tracks in Snow*, both as a game interface and as a new domain for AI systems.

Following from the discussion of systems, Chapters 8 through 10 describe evaluation techniques focused on supporting of designer understanding of AI systems as well as lessons learned while authoring with Puppitor. Chapter 8 posits the *Expressive Response Curve* as a method of mapping out the expressive possibilities and player experience of a given Puppitor domain. The Aesthetic Comparison of Algorithms, discussed in Chapter 9, expands on this research and turns the evaluation on the algorithms driving decision making rather than the domains they operate on. The goal with the work in both these chapters is to give insight to the way underlying technical implementations of systems impact the end player experience. Following from this, Chapter 10 describes the authoring experience and ergonomics of designing characters with Puppitor for *Tracks in Snow* and the MIMIC crowd simulator. In particular, I highlight how project needs and external decisions change the approach needed for developing dynamic characters using Puppitor. The purpose of these three chapters is both to provide techniques to enable more insight into the complexities of AI systems and to better support developers in approaching technical decisions with intention.

I structured this dissertation and the approach of computational theater around my own research process of building-to-understand [171] and research through design [281]. I first identify a particular design space to operate within and formulate a high-level design goal. With this design goal in mind, I construct a system to explore

the chosen design space and implications of the high-level design goal. In parallel, I develop a specific storytelling experience using the system. These parallel forms of development allow for reflection on the initial design goal from both technical and aesthetic perspectives. Lessons from these reflections is then reincorporated into the system and experience design. These lessons in turn improve the understanding of the system's capabilities and the nature of the design goal as well as cash out to software tools and evaluation approaches intended to support further development using this iterative process.

Chapter 2

Background

One of my purposes with this dissertation is to call for a re-examining of how we view interactivity as it relates to storytelling. Much of my research, as someone working in both theater and computation, could fall under the label of interactive drama. After all, I do make dramas and they are interactive. But I have rarely completely embraced that umbrella for my own work. This chapter provides part of my answer: my work draws from different parts of theater than the work which labels itself interactive drama.

Interactive drama roots its idea of how to make a story interactive in structure. That is, for a player to have agency within a story, they must be able to change the outcome through their actions. This idea stems from Aristotle's *Poetics* and is the assumption underlying interactive drama as well as numerous other approaches to interactive storytelling. What results are stories, explicitly or implicitly, focused on the outcomes of actions.

My background is in playwriting. Even if I do not subscribe to the idea that

playwrights exist exclusively to create structure [26], I recognize that much of my role is to provide structure for directors, performers, and designers to then experiment, and dare I say play, with. My experience as a playwright leads me to question why interactive drama is so fixated on outcomes. It is my job as a playwright to worry about outcomes and structure and yet interactive drama asks its players to both perform a character and choose the shape of a story's structure.

I don't see anything inherently wrong with focusing the player's interaction on structural changes to stories. More, I want to provide new ways, alternative ways for players to interact with a narrative. Structure is not the only place for creativity and expression. My reason for bringing up my background as a playwright was not just to say that interactive drama is partially interested in the work of playwriting. Rather, it is a style of writing that makes a story capable of being played with.

When I was writing the initial draft of this chapter, I was asked if I had encountered any writing challenges working on the projects in this dissertation. Somewhat surprisingly I said no. My answer is not because of some special authoring system I created or because I can write a massive amount of content quickly. It comes down to how I approached the problem of making a story interactive. I threw out the assumption that for a player to have control over a story they must be able to change outcomes. Instead, I wrote my stories as I would any play.

Now I admit, I am the kind of playwright who likes leaving things open for productions and finds incredible joy in seeing where people take my stories. I only make as many decisions about my plays as are necessary for me to tell the story I want. My

scenes are usually dialogue heavy and I use stage directions for pacing and hooks rather than micromanaging a production. There's freedom as a writer to be able to say that *something* must happen at a particular time but then let others decide exactly what that something is. I don't say this out of laziness. I say this because it is what I mean when I say that playwriting makes a story capable of being played with.

A script provides the major structure of a story, the order of events, who says what, what kinds of relationships characters have. But it is an unfinished structure. Finishing the structure, turning a script into a play *is* highly interactive, as performers describe in section 2.2. By focusing so deeply on structure and outcomes, we have left this aspect of expressivity and creativity on the table. My desire to build systems and experiences around the interactivity of performance rather than structure is why I set myself apart from interactive drama. It is why I advance the idea of computational theater as an alternative to interactive drama, with its foundation in interactivity through structure. I define computational theater as the development of systems which use performance as the basis for their mode of interaction.

My goal with my research is to create new forms of interactive stories and play. As a result, I had to return to the foundational inspirations and assumptions of how to combine theater and computation. I am as much a thespian as I am a computer scientist and I owe it to both fields to understand them as deeply as I can while developing new systems and interaction paradigms. This dissertation is about my approach to narrative and system design. A huge part of that stems from living in both the theatrical and computational worlds. If there is one thing I want you to take away from this chapter

it is to be curious about your influences and inspirations. It might not be realistic for most people to work on theatrical productions and develop AI systems (or your field's equivalent), but it is something I want to see interdisciplinary research strive for. Doing so enables you to go back to the beginnings of a field and carve a new path. It's hard, lord knows I haven't completely finished what was originally supposed to just be my master's thesis project as I'm writing this dissertation, but it's worth it. You will be able to bring amazing new things into the world.

This section is devoted to understanding the history of interactive drama, how performance practices understand their role in storytelling, and the ways theater and computation have been combined beyond interactive drama. I have already touched on some of the history of interactive drama, but in the first section of this chapter I delve much deeper into why and how the discipline developed. The second section surveys a number of performance practices used in theater as well as touches on dramaturgy, as the latter is the main lens I use to approach media criticism. Finally, I discuss approaches to combining theater and computation beyond interactive drama, with an eye towards how they fit under my umbrella of computational theater.

2.1 Interactive Drama

The primary goal of this section is to define interactive drama using the systems that label themselves with the term. Part of this process involves understanding where interactive drama came from and how those initial priorities shaped its evolution as a

design methodology from those first inspirations to its current focuses. My goal with this dissertation is to posit a new approach to building systems grounded in theater practices. To frame my reasons for developing a new approach system design, this section explores the history of systems grounded in theatrical theory, namely Aristotle's *Poetics*. Of particular interest to me is the way the inspiration and grounding of early work in interactive drama became foundational assumptions even as new sources of inspiration took center stage on projects.

2.1.1 Aristotelian Drama

Brenda Laurel's dissertation [154] laid part of the foundation for interactive drama with the discussion of using Aristotelian drama as the basis for allowing a player to act out a character in a digital play. Laurel selected Aristotle's *Poetics* as the foundation for her ideas on the basis that it remains the most wide-ranging and complete theory of drama available, as well as the spring board numerous other, less complete, theories of drama [152]. The other reason for the emphasis on *Poetics* is the assumption that because Aristotelian theory is so expansive and coherently articulated it provides the ideal foundation for building computational models with theatrical qualities [152]. This manifests in Laurel's usage of Aristotle's four causes: *Formal* (the completed plot of a play), *Material* (the actions performed on stage), *Efficient* (the techniques used to stage the finished production of a work of theater), and *End* (the emotional experience of the audience) [152], as providing a guide to human-computer interaction in a similar way to drama. When translating the causes' application from drama into human-computer

interaction, the changes to definitions primarily come from the expansion of the scope of building computational experiences rather than producing a play. The formal cause now expressly includes the person performing the interaction. The material cause now includes the computer and its abilities to enact change in addition to the person involved. The efficient cause now includes the skills of people possibly far beyond those directly creating the experience thanks to the nature of code being built on top of other code. The end cause now focuses on the outcome of the interaction between human and computer rather than the audience's emotional experience.

By drawing primarily from Aristotle, there is also the focus on plot rather than character being the foundation for drama. Laurel characterizes this preoccupation with plot as "the characters [being] there because they are required to represent the action, and not the other way around" [152]. While action is indeed the foundation for drama, the Aristotelian view is less concerned with the details of those interactions, the stage action, than the way plot action contextualizes events and defines characters. When actions are seen as enough to define a character, the details of how a character performs those actions are not always deemed important when building systems. This is why this literature review and my prior work [129, 130] push for theatrical practice to be further explored, they are systems for understanding and enacting those character details. The rest of this section is devoted to exploring the ripple effect of this initial focus on plot action to the development of interactive drama.

How would one go about building an interactive story when drama requires structure and interaction is chaos? This is the question the Oz project's usage of a drama

manager sought to answer. This drama manager is concerned with the organization of important plot points, ordered using an evaluation function intended to capture an author's aesthetic goals [165]. In a live scenario, the drama manager listens until it recognizes a transition to a new beat, then it will try to change the state of the world such that the new beat is structurally satisfying [165]. This approach to imposing structure on interaction is directly inspired by Brenda Laurel's work [165] and ultimately has the goal of making the act of writing a scripted drama happen in conjunction with its performance. A side note, this approach to restructuring plot action fits extremely well into Aristotle's notion of the purpose of a playwright, to arrange action, not create it from whole cloth [26].

To create an actual interactive drama using the Aristotelian model as described by Laurel, there was the question of how the goal of a player having agency in the experience. Agency as defined by Janet Murray is when the actions a player takes have noticeable effects that clearly arise from those actions [197]. As part of creating a poetics for interactive drama, Michael Mateas places the player's role in the Aristotelian model as a character, one constrained by what actions are reasonable within a story world [167]. The goal of interactive drama at this point is to provide a play space within a story that only allows actions that fit within its context while also allowing a player to alter the direction of that story in noticeable ways. All of this is chasing the notion that a player experiences agency when the rules governing the world align with the narrative's suggestion of action [167]. Here is again where the focus on theatrical theory misses the importance of how stage action can provide agency to actors and by extension players,

without having to alter a plot's structure on the fly. One could even argue that the example of *Quake* as a high agency game [167] falls under the purview of stage action as its structure remains unchanged but the details of how a player progresses through an environment change every playthrough.

Façade [172] is the one end result of all this work in applying and altering Aristotelian dramatic theory to form the foundation of an interactive drama experience. One element of *Façade*'s design that remains relatively unique within interactive drama is its relatively tight coupling of its agents and its drama manager due to beats being a core part of the architecture and a relatively small unit of action [169, 170]. This led to the creation of joint behaviors as a way to coordinate actions across characters and allow for dramatically interesting moments between characters [173]. Given the explicit goals of making a structurally sound and satisfying dramatic experience, *Façade*'s focus on coordinating characters is very understandable, though it begs the question, why have a dynamic plot structure in the first place? The answer is of course that agency in an interactive drama comes from a player's ability to satisfyingly effect the plot *structure*, thus noticeably changing the outcome of the plot. This is the limitations of the Aristotelian view of drama and the attachment to agency at play, one that using stage action as the basis for interactive storytelling provides an alternative for.

2.1.2 Social Simulation Approaches to Interactive Drama

My view of the following systems is that they are the Aristotelian approach used by *Façade* combined with some other element of sociology or psychology to make

more improvisational experiences. Thespian models social norms around human conversation. *Comme il Faut* uses Goffman's sociological dramaturgy as its model for character interaction. Versu implements the constitutive of social practice. Villanelle is an outlier in its primary focus on authorability of characters. That is to say each of these systems, regardless of their distance to the conception of interactive drama as specifically rooted in the neo-Aristotelian theory as described by Laurel and Mateas, still operate within those assumptions. Plot action under a player's control is foundational to the experience, even if they are also capable of modeling stage action, just in these systems' cases, that plot action is more emergent than in *Façade*.

Thespian is an interactive drama system primarily focused on making authoring interactive dramas closer to the experience of writing for a traditional drama, using goal-based agents that fit their behaviors to a given script [238]. An agent's personality is defined by their various goals and the importance of those goals to allow the interpretation of actions from a given script into generalized actions within a story world [238] and make any author defined goal weights more accurate to the agent's perception of a script [239]. Thespian's usage of a single script to then generalize into a dynamic plot is reminiscent of the process of tuning a drama manager's heuristic in the Oz project using an original story to test that heuristic's effectiveness [165], just distributed across the characters in the story rather than a single drama manager. Thespian is primarily concerned with social norms around conversation. As a result most of the available actions are specifically dialogue related, its primary use case being language training [237]. Characters use the connotation of these actions to update their beliefs about the state

of the world and the goals of other characters, then taking actions to push the world towards a more desirable state [238]. Thespian's fairly granular approach to conversation, when combined with its pedagogical goals, leads to a very tight relationship between any action taken and the direction the plot through a scene will take. So while the system does operate with a certain respect to stage action, it is less about finding the expressiveness of the space within a plot than understanding the consequences of actions.

Comme il Faut (*CiF*) is a system for autonomous characters rooted in Goffman's dramaturgical analysis of social behaviors as the foundation for the development of interaction around social games [180]. Focusing on social games as the primary level of interaction allows for more generally applicable collective actions built around social norms and characters' usage of those norms in order to present themselves in particular ways to accomplish their goals [181]. The outcome of social games in *CiF* is the modification of each involved character's internal social state as well as the state between those characters [178]. All of this design comes together to create the social logic governing an entire story world [178] rather than the constrained plot space of *Façade*. With that said, the final version of *Prom Week* [179], the game *CiF* was built to support, did feature specific plot lines following particular characters and their social goals, framed as social puzzles [177]. *Prom Week* and *CiF* offer a far less stage play oriented version of interactive drama than *Façade*, though the assumption about dynamic outcomes of a plot being integral to the experience remain as endings are determined by which goals are accomplished or failed [177]. The social games *CiF* uses operate in the domain of

plot action as the results of the interaction within the social game are the system's primary concern, not the play of the game itself. As a result, even if a characters' behaviors within a social game defines them in the Aristotelian sense, their performance of the games themselves becomes flattened to the resulting outcomes, which, as Alex Mitchell notes, can push the stories being told into the background [188].

Versu is another multiagent simulation system designed to support interactive dramas and is focused around modularity using social practices as its core model of character behavior [72]. Of note is how Versu's design approach is directly in conversation with *Façade*, including its push towards autonomy, a very hands off drama manager, and its usage of text to surface more easily expose simulation elements and character than would be reasonable relying on animations [72]. The reason for a strong autonomy approach was rooted in similar considerations to *CiF*, the desire to build up the social foundations of the story world, in Versu's case it is with the constitutive model of social practices [72]. The social practices themselves constitute affordances available for characters to perform, multiple of which are allowed to operate concurrently [72]. The ultimate goal with Versu was to create a system that allowed any character to be playable and characters to be cast in different roles within a story, one where any small action, including doing nothing would be noticed [72]. In this way Versu is rather unique in that it does operate using stage actions, as how action is taken is as important as what the action is in the first place. Versu is also explicitly an improvisational drama [72] rather than a scripted one, so plot structure remains extremely malleable which can make understanding characters' decision making require some degree of replay, though

learning their response patterns is certainly doable, as Alex Mitchell describes [188].

Villanelle is a system designed around making the authoring of autonomous characters more easily accessible to non-experts by focusing on language design [164]. The system is primarily intended to work within interactive fiction contexts due to the active hobbyist community and relative approachability of the medium to less technically experienced authors [164]. The two primary features of Villanelle's approach to the problem of authorability in generative narrative are: a language-based model of authoring explicitly designed for approachability partially through its foundation in behavior trees rather than the highly complex and bespoke languages found in *Façade* and Versu, a practice-driven view of the design by engaging directly with the interactive fiction community to understand their needs around creating more autonomous characters [163,164]. Behavior trees were chosen as the foundation of character behavior due to their relative ubiquity in the game industry and ease of construction [163]. Villanelle's focus on authorability means it is far less concerned with challenging assumptions made about character behavior than making the existing assumptions about that behavior more approachable for non-experts to use. As a language, one focused on character expression, it is capable of having characters perform both plot and stage action, though most of the examples used focus on actions themselves (plot action) rather than how they are done (stage action).

If *Façade* took the middle ground between fixed structure and fully autonomous characters with its focus on beats and joint behaviors, there remains a desire for more modular and open ended character interaction. Some of this is of course to make for

more expansive character interaction, but one of the underlying assumptions remains that changing the outcome and even high level structure of a plot is something desirable. In fact they lean even more heavily into making altering the plot central to the experience, as these systems are characterized by their relatively hands off drama managers, something I might add that *Façade* explicitly avoided as part of its goal of having a well structured plot [170]. Improvisation is certainly a part of theater, and Versu even explicitly wants to be an improvisational drama, something it succeeds quite well in, but what these approaches leave behind is in fact theater. They mostly ground their additions and changes to the Aristotelian model of interactive drama in psychology and sociology, which for their aims makes sense. These moves are, however, my reason for proposing computational theater in the first place, my dissatisfaction with the focus on plot action and the shift away from theater that has characterized interactive drama.

2.1.3 Planners for Characters and Story Generation

Rather than being built to implement specific dramatic, sociological, or psychological theory in a system, planning based systems use a particular technical approach, the eponymous planner, as the foundation for their implementation of character or storytelling behavior. A planner fundamentally is based on starting from some initial state and constructing a sequence of actions to reach a specified goal state [279]. Generally a planning problem is composed of a specified initial state of the world, a goal state of the world describing the elements that need to be changed, and a library of actions describing everything the planner can do in the world [279]. A solution is reached when

the planner finds a sequence of actions that, when executed in the correct order, will make the world line up with the given goal state [279].

Doing a full survey of planning based approaches for story generation and character control is beyond the scope of this paper as there are numerous approaches [14, 31, 40, 44, 99, 211, 222, 278] all falling at various points on the the spectrum of strong story (primarily focused on unified plot [275]) to strong autonomy (primarily focused on consistent character behavior [275]). The three systems highlighted in this section represent three points on the spectrum of strong story to strong autonomy. The interactive storytelling engine from *Madame Bovary* on the Holodeck exemplifies the strong autonomy approach. Glaive [275] represents a balance between both ends. Sabre [274] represents a strong story approach.

Planning based approaches focus on the technical foundation which, while allowing implementation of particular models of character, means they do not necessarily need a specific theoretical grounding outside of computation. With the exception of Villanelle, the systems in the previous section call themselves interactive dramas even with their drift away from theater as their primary source of inspiration. The systems in this section have even less connection to theater and do not always connect themselves to interactive drama, or even specific domains outside of computation at times. I have included them in this section because they share the goals of reactive characters and plot to the more explicitly interactive drama systems and provide an example of when an approach to modeling causality, under the default assumption of the existence of an objective world state, is used to drive character behavior.

The interactive storytelling engine found in *Madame Bovary* on the Holodeck is rooted in Heuristic Search Planning (HSP), with each character running their own HSP planner as part of the system's character based approach to interaction [45]. The system describes narrative state as the feelings of their main characters about their current situation, modeled using the inventory of feelings described by Gustave Flaubert, the author of *Madame Bovary*, with each feeling having three possible discrete states [45]. Character behavior is mostly driven by their psychological state, feelings turned into planning goals, which in turn motivates the characters to interact with each other using associated planning operators [45]. The storytelling engine found in *Madame Bovary* has some similarities in approach to the social simulation work that *CiF* and Versu would eventually use in their own character-based approaches. The engine's basis for character knowledge is, however, rooted in the literary rather than sociological and its foundation of character interaction exists somewhere between the rigid joint behaviors of *Façade* and the generalized, formalized exchanges of *CiF* and Versu. As part of the engine's explicit goal of reaching for the Holodeck, it does engage with both plot and stage action, though plot action is still prioritized systemically, with the characters' focus on acting on their feelings rather than how those different feelings might be used with the beats of the original novel.

Glaive is an intentional planner designed to reason about character intentionality and conflict by representing alternate worlds [275]. Intentional planning is a middle ground between the strong story and strong autonomy ends of the planner spectrum to allow individual character interaction while still serving higher level narrative

goals [275], which bares at least some similarity to *Façade*'s approach to combining its characters with its dynamic plot. Glaive uses failed plans as part of how it explains why characters took the actions they ultimately did [275]. Beyond the plans themselves being recognizable as something found in traditional story forms and characters being able to pursue their goals in balance with higher level narrative objectives [275], there is little direct connection to the interactive drama systems that narrative planners share some lineage for. Where we can still see those roots, however, is the focus on plot action from Aristotelian dramatic theory, including the systemic explanations for those actions.

Sabre is a strong story planner that coordinates multiple characters to fulfill narrative goals without anyone breaking character [274]. Characters in Sabre are primarily defined by their beliefs, and while they each have utility functions to push them to act to fulfill their goals, the story planner itself is the soul decision maker and will control when characters are allowed to act, in service of its high level goals [274]. One of the planner's primary modes of changing the state of the world is choosing actions to perform [274]. The characters involved must each have a justification for performing that action and can be observed by any relevant characters, who will in turn change their beliefs based on that action [274]. The other method of changing the world are triggers which fire when their preconditions are met and are often used to update characters' beliefs [274]. As a system explicitly modeling the deep theory of mind, Sabre further pushes for character internality, which can lead to stage action as Stanislavsky's volitional objectives describe, but the focus is not character expression of emotion as

an extension of their actions. Like with Glaive, there is very little direct connection to interactive drama work beyond the extreme plot focus, though Sabre's plot-centric approach does have at least some passing resemblance to the Oz project's drama manager in the way it orchestrates events.

Narrative planners share a lineage with interactive drama systems but prioritize a particular technical approach to problem solving and the operationalization of theories into computational models. As a result planners have drifted much further away from their dramatic roots than their interactive drama contemporaries. A shadow of Aristotelian dramatic theory remains, however, because they primarily tend to operate over plot action. Their reliance on knowledge representation as part of decision making means they tend to create characters with a lot of internality, sometimes in an explainable way as with Glaive. These systems often have relatively few ways of directly surfacing that internality directly. What this leads to is computational characters exhibiting qualities of Stanislavsky's method that have been critiqued for their myopic focus on the actor's psyche rather than what and how the actor performs [29, 158].

2.1.4 Non-Aristotelian Interactive Drama

The last two interactive drama systems I discuss fall outside of the categories I've focused on so far as they don't draw as heavily from Laurel's work or Aristotelian drama in general. In IDtension's case this is because the theory of narrative the system uses has very different priorities. In Mirage's case, one of its major systemic focuses is implementing particular craft knowledge rather than a specific overarching theory.

As a result this subsection is less of a commentary on the development and lineage of multiple systems and more of a commentary about the systems themselves.

IDtension is an interactive drama system that attempts to create a dialogue with the audience and very specifically does not consider the player a co-creator [259]. Rather than focusing on allowing character behavior to build an emergent narrative or plot to be dynamically built from scenes, IDtension focuses on narrative properties and the simulation of narrative laws, in a similar way to physics simulations, to create stories following those properties [259]. The actual model IDtension uses is grounded in three layers of narrative: the discourse layer which attempts to convey messages to the audience, the story layer which describes the sequence of events making up the narrative, and the perception layer which draws the audience to understand and engage with the narrative [259]. The system itself models character actions as speaking or performing an act that is significant to the narrative [259], in other words, still plot action, even if it is not directly drawing on the neo-Aristotelian model of drama. Authorial values and audience reaction are all also explicitly modeled within the system as part of its goal of trying to communicate a degree of morality through the story produced [259]. IDtension attempts to be a synthesis of multiple approaches to storytelling as part of its view of narrative theory and yet still has a similar view of plot to the Aristotelian approach to interactive drama, that to engage a player, the plot must bend around their will. The attempt to model the dialectical aspect of drama, while unique, leads to an overly simplistic view of that dialogue, boiling it down to the idea that narratives define some actions as good and some as bad and this is dependent on whose system of values

is being used [259]. The most curious part about the desire to model this dialogue with the player is that it appears to operate under the assumption that the dialogue itself must be modeled and is not more foundational to the practice of storytelling.

Mirage is an interactive drama experience adapting the legend of the house of Argos into the new form and adding a player to the story in the form of Archemedis [68]. The systems supporting the experience include a story engine, characters, a director, and a user model [68]. The story engine is similar to the work done for *Façade* with the addition of the user model as part of its selection criteria [68], so we remain in the realm of Aristotelian theory to an extent. The director's job is to choose goals for the characters, who choose behaviors to perform in pursuit of accomplishing their given objectives following principles drawing from Stanislavsky's volitional objectives [68]. The user model's primary purpose is to allow characters and the other systems to reason about what a player is trying to express through their actions [68]. For example, the length of a pause before a player taking violent action will determine how violent the systems think the player is, shorter pauses being interpreted as more violent [68]. Additionally, *Mirage* features a dynamic lighting system inspired by theater and film practices to allow the lighting of scenes to be based on character locations and help better reflect the tone of scenes [69]. *Mirage* provides me with an interesting conundrum when it comes to its approach to interactive drama, it still very explicitly roots itself in the Aristotelian approach with how it treats plot, but it does draw much of its design from more practice based parts of theater and film and does try to support dynamic stage action though the importance of adverbial additions to action [67]. This is most

apparent in its Expressive Lighting Engine [69] in how the system is there to support and embellish the actions taken rather than drive the plot forward. While the focus on dynamic plot makes me hesitant to call the entire system a work of computational theater, *Mirage* more than many of the systems discussed in this section has elements devoted entirely to supporting stage action such as the Expressive Lighting Engine and elements of its character behavior are certainly works of computational theater.

Interactive drama as a methodology has drifted from its roots in Aristotelian dramatic theory and, noticeably, it is the earlier systems, *Façade*, *Mirage*, and IDtension that have the most grounding in specifically theatrical practice and theory. As priorities shifted, new theories were drawn upon to allow for more emergence with systems like Thespian, *Comme il Faut*, and Versu as well as more focused technical approaches like with *Madame Bovary*'s story engine, Glaive, and Sabre. My goal with my work is not to discount any of these directions. Instead, it is fueled by my desire to see more systems supporting and understanding stage action. This desire is the reason I have focused my work squarely on creating computational systems using an understanding of theatrical performance and production. Computational storytelling and character interaction has largely left the creativity and expressive ability of performance on the table. My hope is that through the work I present here, and the broader idea of computational theater, I can at least nudge the field towards picking up this aspect of storytelling so brimming with potential.

2.2 Theatrical Practices and Puppetry

This section provides an overview of theatrical practices and their goals of supporting and teaching the expression of stage action. Some of these acting practices, like elements of Stanislavsky and Laban's writings, have found their way into computational work semi-regularly. Other practices such as the Viewpoints and Nō have only rarely found their way into computation. As stage action is my primary concern with the creation of computational theater, the focus is on performance practices, primarily acting and puppetry. Acting is of course well known and studied and the practices included are only a slice, a well trod and practiced slice but a slice all the same, of the approaches used. Puppetry is far less studied as its own practice, as some of the authors gathered here lament, and I have included discussions of it in this literature review for its unique perspective on stage action as distinct from acting as well as its applicability to computation, particularly character expression. Finally, I have included a discussion of dramaturgy, as it is a practice devoted to shaping stage action in relation to the text of a script. To put this section directly in conversation with interactive drama, as Yamazaki Masakazu (one of the translators of the Nō treatises) notes, Aristotle's *Poetics*, while concerned with the narrative structure of a play, disregards how that play will be performed [223]. My goal with this section is to highlight the creativity inherent to stage action and its importance in creating characters and telling stories. The creative opportunity in the performance of a story underpins all my

2.2.1 Stage Acting or Humans Performing Stage Action

I took the phrase "stage action" from Stanislavsky [158] and while not every acting practice uses that exact phrase, the importance of using action for expression remains a common thread throughout the discipline. How these expressions are performed and what aspect of human nature they try to capture has changed over time, though the process of acting itself repeatedly returns to the goal of continuously finding novelty and additional nuance in a text to then convey [158,223]. In traditional Western theater, the changes in what is being captured has been rooted in the constantly developing understanding of human psychology and emotion [224]. This subsection covers Zeami's Nō drama [223], Stanislavsky's Method of Physical Actions [158], Laban and Lawrence's Effort [175], and Bogart and Landau's Viewpoints [29] and how each views expression on stage as a creative process.

Nō theater, as described by Zeami Motokiyo, was developed over fourteenth and fifteenth century Japan where it transformed from country entertainment into high art over the course of his life [223]. Unlike many Western theater practices, Zeami placed a huge amount of weight on the audience's engagement, to the point of making it the actor's responsibility to perform in such a way that even a person with no knowledge of the art on display will find it enjoyable [223]. Central to Zeami's discussion of performance is the idea of the flower as the central aesthetic of theater, something that produces novelty (making the old look new) and must be actively cared for throughout practicing the art itself [223]. When discussing acting itself, Zeami notes that it is not

sufficient to plainly reproduce human action realistically through imitation nor is it sufficient to only attempt to emotionally become a character [223]. A balance between the two is necessary, still drawing from realism but allowing the expression of emotion to guide parts of the performance, even likening actors in this state to puppets for the similarities in the need to pursue something beyond the manipulation of an object, human body or otherwise [223]. Zeami's view of the aesthetic of theater as constantly needing to pursue novelty within the bounds of the stage is the core of my interest in stage action as a part of interactive storytelling. Just because a play has been performed numerous times does not mean there is nothing left to explore and it is part of an actor's job to help the audience appreciate the performance or story whether it is their first or hundredth time experiencing it.

Stanislavsky's Method is one of the more ubiquitous acting practices of twentieth century theater [29, 158, 223], though the most common association it carries, that of volitional objectives (a character's desires) and the superobjective (the force driving that character's actions throughout a script) [158], is not what my discussion focuses on. This version of the Method is both extremely internally focused and incomplete when it comes to Stanislavsky's view of stage action [158]. Stanislavsky's goal with the development of his Method was to allow actors to maintain a feeling of liveliness and interest regardless of how long those roles would be performed, rather than relying on something fleeting like passion or the novelty of a new character [158]. This was where the idea of inhabiting a character's psychology developed from, but even the focus on objectives did not always produce the desired effect [158]. This dissatisfaction

led to the further development of the Method of Physical Actions with its focus on expression through action and the relationships between characters and performers on stage [158]. To Stanislavsky, actions always have the goal of motivating a reaction from another character, characterized as changing the flow of energy between the characters involved in the action [158]. In order to keep performances alive and actors engaged, even after the outline of scenes has been constructed and set in rehearsal, Stanislavsky stresses the importance for the whole cast to explore the space within a production's fixed choices, lest their acting become mechanical and no longer supporting the actions of the plot [158]. As with Zeami, Stanislavsky's concern is with maintaining the quality and liveliness of performances over time, though who this is for is a little different. For Zeami it was for both actor and audience. For Stanislavsky, this focus on novelty is primarily for the actor, with the audience's experience being treated more as a bi-product. Another contrast between these two approaches is how much importance Stanislavsky places on the relationship between actors on stage as the foundation for maintaining novelty while Zeami considers it each individual performer's job to find and maintain their own novelty. While they differ in aesthetic goals and approaches, the foundation between the two remains the same, maintaining the strength and expression of performing stage action over productions and careers, be it 1300s Japan or 1900s Russia.

Laban and Lawrence's view of effort focuses on building a language to describe the way people, trained dancers or not, move through space and time [175]. The learning process, as Laban and Lawrence describe, is primarily about teaching awareness of the

motions a person performs, with a focus on identifying extraneous movements which may detract from a performance, and they stress that anyone can learn this awareness, even through self-teaching (though it will be a slower process without someone to guide the process) [175]. For Laban and Lawrence, it is vital to train in combining movements and efforts as this skill can lead to creative connections of movement and actions [175]. Effort itself is divided into four different psychological aspects: time, weight, space, and flow, each having the potential to convey a struggle with or indulgence in that effort [175]. Time is characterized as the presence or absence of rapidity in a movement [175]. Weight is characterized as the presence or lack of bodily force and the amount of energy spent overcoming one's own body weight [175]. Space is characterized as restricted and calculated movement or the flexibility and ease of movement [175]. Flow is characterized as free when movements are difficult to stop or interrupt and bounded when movements can be stopped instantly [175]. Importantly, the expression of effort is a part of actions, no matter how small those actions are [175], sharing some similarities to Stanislavsky's view of actions' relationship to the flow of energy. As the aspects of effort are intended to be descriptive of the way people move, and anyone can move between indulging in or struggling against any of the aspects at any time, describing even every day behavior as simply energetic or gentle is seen as overly reductive [175]. Effort is about providing language and awareness of physicality in general rather than specifically for the stage and as a result is less focused on expression specifically related to plot action when compared to Nō and the Method of Physical Actions. Even with its incorporation of the study of the everyday, Laban and Lawrence's effort remains concerned with the way

physical expression through motion conveys a person's feelings, which can of course be used as part of additional creative practices like storytelling.

The Viewpoints as described by Bogart and Landau is born from a desire to question as much as possible about the art of performance, a dissatisfaction with the internally focused version of Stanislavsky's Method that had become pervasive in the United States, and the creation of language describing physical performance in space and time [29]. Rather than the internal, the viewpoints specifically focus on the way action arises from groups of actors engaging each other in space [29]. Part of this shift towards group dynamics and action was moving away from asking what individual characters want or even what a director wants and towards collectively trying to find an answer to what the play itself wants from the members of a production [29]. More than anything the Viewpoints is intended to foster collaboration between actors to explore how they can discover actions on stage by indulging in possibilities rather than being restricted under some central authority, including the text of a play [29]. The nine Viewpoints of time and space as defined by Bogart and Landau are as follows: tempo is the rate at which a movement occurs; duration is how long a movement or sequence of movements continues; kinesthethic response is a spontaneous reaction to external motion; repetition is the repeating of some action on stage; shape is the contours or outlines a body or bodies make in space; gesture is a movement involving part(s) of the body; architecture is the physical environment's effect on movement; spatial relationship is the distance between things (especially bodies) on stage; and topography is the landscape and design created using movement through space [29]. More than any other practice in this section, the

Viewpoints are focused on fostering the collective creativity of performers and freeing them from the individual responsibility of understanding everything about the play being performed. With this intensely collaborative approach, there remains a focus on finding novelty within expression and physicality on stage, that novelty now explicitly coming from a group. Much of the language used when describing the philosophy of the Viewpoints is centered around the opening up the possibilities of a text by closing off avenues of exploration.

The commonality across each of these theatrical practices is the power that understanding that physical action carries a huge amount of expressive power, whether it is tied to a script or not. This knowledge developed over lifetimes of performances is the foundation of my interest in developing computational theater as a methodology. We need not contort plot action around a player's actions to give them the sense that they are a part of the process of telling a story. Instead we can support them with the understanding that stage action is as an act of creativity and expression unto itself which can shape a story in its own unique ways.

2.2.2 Puppetry and Stage Action

Acting practice's focus on stage action is rooted in the assumption that a person will ultimately be performing that stage action still using the constraints of their own body. Even when the goal of acting practice is changing a person's relationship to their body, as Zeami [223], Bogart and Landau [29], and Laban and Laurence [175] describe, there are still limits on how far that relationship can be pushed. With pup-

petry, the performance is extended outside the performer's body, sometimes very far outside, and additionally is mediated by the puppet itself through its own physical limitations [268] and further solidifying the role being performed [132]. The mediation of performance and stage action is the foundation of my interest in including puppetry as part of this discussion of stage action. Not only does puppetry provide an additional performance focused perspective beyond acting, but because unlike stage acting, which can require a large degree of translation work to apply directly to the design of computation [68, 118, 127], puppetry and purely digital characters have parallel relationships with mediated performance [132,265]. This subsection covers more traditional aspects of puppetry and its relationship to other parts of performance craft with Stephen Kaplin's model of performance [132], Basil Jones' discussion of the narrative underlying puppetry performance [123], and Eric Bass's explanation of visual dramaturgy [19], as well as Steve Tillis's view of puppetry's relationship to new artistic practices enabled by computation [265].

One of the foundations of performance is the stage persona that will be present whether the performance is a recreation of an offstage personality or as an object such as a puppet [132]. Dissatisfied with the focus of describing puppetry primarily through its materiality, Stephen Kaplin proposes instead to focus on the dynamic between a puppet and a performer [132]. He defines *distance* as "the level of separation and contact between the performer and the object being manipulated" where one extreme is absolute contact, where the performer and object are the same, and growing through layers of psychic, body, remote, and temporal levels of contact [132]. The other aspect

of his model is *ratio*, or the number of performing objects to the number of performers, where a one-to-one ratio is a single object and single performer, a many-to-one ratio is multiple objects controlled by a single performer, and a one-to-many ratio is a single object controlled by multiple performers [132]. Kaplin beings characterizing distance at first as a slight shift in presentation, widening in psychic distance as roles become more distinct from a performer, until the role begins to manifest as a physical object like a mask, or at further remove, a puppet [132]. Once the role has become a completely separate object, it shifts the focus of the performance off of the performer [132]. Kaplin notes that as the distance increases between a performer and a role, increasing amounts of technology are necessary to keep the two connected [132]. Traditional rod and marionette puppets still feature a physical connection between performer and puppet while the puppeteers controlling animatronics' motors and servos may not even be in the same room [132]. This distance becomes even more extreme when applied to motion capture performances which can in some sense close the distance as characters are once again directly controlled by a human body, but the amount of technology and computation required make the realities of puppeteering characters live like this a difficult proposition [1] [132]. This model of distance between performer and role is important to contextualizing the performance of stage action when mediated beyond the ways acting practice tends to view roles. By their nature as a physical manifestation of a role, puppets constrain certain aspects of a performance by needing to maintain increasingly

[1] Kaplin wrote this in 1999 and as both processing power and software has improved since then, we do in fact have individuals, such as vtubers, able to do live motion capture performances with a computer, phone, internet connection, and some specialized software.

complex connections between themselves and the performer. What is gained, however, is the ability for a performance to move beyond the constraints of the human body and for roles to make suggestions about how to perform them.

Puppets are dead and actors are alive [123]. This is the fundamental difference between actors and puppeteers described by Basil Jones, that a puppet's struggle is the performance of life [2] regardless of any other textual or directorial choices made, and it is the puppeteer's job to maintain that fragile life to keep the audience suspending their disbelief [123]. The underlying quest for life in every puppet performance can turn the mundanities of life into dramatic struggles, and with enough mastery, the puppeteer's performance of even the mundane can draw audience attention away from the the larger scale actions of plot and story [123]. The puppet's tenuous grasp on life then serves as an invitation of sorts to the audience to interpret the performance and seek meaning in any action the puppet takes, potentially to the point of drowning out spoken words [123]. In this way, Jones sees a fundamental tension between playwright and puppeteer, text is often an authoritarian presence in theater [3] and puppeteers' performances can, in a sense, rebel against this assumption with the importance of every action a puppet takes conveying the struggle of being alive [123]. Acting still tends to consider itself subservient to the script, both Zeami and Stanislavsky explicitly say that all actions *must* come from the text. The tension Jones describes when puppetry and

[2] Jones also describes the designer/maker of the puppet as partially being responsible for a puppet's ability to feel alive.

[3] In a production of *War Horse*, Jones notes that the scenes heavily featuring the puppet horses needed to be "written" by the puppeteers experimenting with their performance rather than a more standard rehearsal process involving written text.

script meet not only demonstrates the power of stage action to shape storytelling, to the point of surpassing the text at times, but highlights how adding a layer of mediation to performance can add extra weight and interest to every action performed. This is something I have felt with my own work in designing expressive characters [127, 128] which have drawn attention away from the text as they are played with. More broadly, Jones' observation connects to the common phenomenon of players ignoring text in games when there are systems to play with, though a discussion of that in detail is beyond the scope of this literature review.

When discussing visual play-making, as distinct from the physical play-making discussed in the Viewpoints, Eric Bass describes it as another reaction to the omnipresence of text's authority in theater and a way to put the play back in plays [19]. Puppetry is a practice fundamentally rooted in the material and as a result, a puppet through its existence on stage is making a statement about its character and does not need words to do so [19]. Bass reminds us that images have the power to be dramatic storytellers without the need for text and that, if the actions on stage simply echo the text, a powerful source of dramatic tension is lost [19]. When stage action and text are in tension, they can elevate each other and add far more complexity than simply illustrating each other [19]. Like Jones, Bass sees puppetry as opposing the frequent authority placed on text above performance, though he specifically points out how this opposition can be used to strengthen the text as it is performed for an audience. Again mediated performances further emphasize stage action and, importantly, the materiality inherent to puppetry calls attention to itself in a way that invites playing with and around the text

of a script by giving performance itself additional weight to bring it more into balance with the importance of the text.

Media figures, as described by Steve Tillis, are characters created through computer graphics and stop-action [4], performances enabled by technological mediation [265]. When setting the idea of media figures in relation to Hamlet's rant about acting holding up a mirror to nature, Tillis comes to the conclusion that one of the strengths of these figures is in their ability to perform stylistically as opposed to naturalistically, and this is their connection to puppetry [265]. Furthering this connection, Tillis notes the similarity between digital characters' articulation variables to the articulation points of a puppet, down to equivalent joint types [265]. Continuing the parallels, Tillis describes how the development of inverse kinematics (a physics simulation technique allowing one to grab part of a model and drag it into place followed by its related articulation points) as enabling control of gestural movement that is analogous to traditional string or rod puppets [265]. The parallels continue as Tillis describes the process of making both a puppet and digital character walk, defining the speed and path of the traversal and combining that with gestures that perform steps independent of the traversal itself [265]. Thanks to all of these commonalities, Tillis describes digital characters as virtual puppets [265], connecting the previous discussions of stage action using puppetry directly to computation. While Tillis specifically discusses three-dimensional, rigged polygonal figures in relation to puppetry (and does not include cel animation under the umbrella of media puppetry [265]), the implications of this connection reach much further than

[4] Tillis views animatronics as separate from this idea of media figures but sharing many similarities.

simply the animation of rigged characters. It begs the full potential of mediated stage action, like puppetry, to be brought into conversation with interactive storytelling practices and I view my positing of computational theater as a step towards facilitating that conversation.

By bringing puppetry into the discussion of stage action, I hope to bring an under-discussed aspect of the performing arts into conversation with computation and interactive storytelling. I also see its mediated mode of performing stage action as further proof that the emphasis placed on plot action by interactive drama as an approach to interactive storytelling lacks important context withing performance. Puppetry also highlights interactive drama's text focused origins which, as Jones and Bass articulate, constricts the ability to play *with* the performance and even the story itself.

2.2.3 Facilitating Stage Action with Dramaturgy

Dramaturgy itself isn't a practice dedicated solely to stage action, it touches everything about theatrical storytelling. I've included a brief overview of it as both part of my general interest in bringing the practice to computation (which falls under my umbrella of computational theater) as well as to explore the parts of it that directly focus on supporting stage action. As with puppetry, dramaturgy has seen relatively little scholarship focused on it until fairly recently but as dramaturgs remind us, the practice happens with or without someone dedicated to the role [48]. So while dramaturgy isn't the literal performance of stage action, it underpins the performance of that stage action and is necessary, with or without a dedicated dramaturg, to make the performance

happen in the first place. This section uses Michael Chemers' *Ghost Light* [48] to provide an overview of what dramaturgy is and how it fits in with acting and performance as the book is a broad introduction to the practice.

So what is dramaturgy and what is a dramaturg? Chemers describes dramaturgy as both the aesthetic architecture (the structure, themes, goals, and conventions) of dramatic literature and the practical philosophy of theater practice used to create a performance [48]. Connecting this understanding of dramatic literature to the transformation of that literature into live performance to engage an audience is research to support the other processes working towards the same goal [48]. A dramaturg is then the person in the theatrical production process who takes a high level view of the process of transforming a script into a performance while also providing a depth of knowledge and criticism to help contextualize and inform decisions made about that process of transformation. So while dramaturgy isn't something that itself *performs* stage action, it plays a huge part in understanding stage action's relationship to the rest of a theatrical production. This connection is why I conclude this section with a (brief) discussion of the practice. It is not enough to simply pursue stage action by itself, not if we want to tell stories and support interaction using the practice through computation. Acting and Puppetry provide details about how to perform stage action as part of a creative process, dramaturgy is what contextualizes and grounds the performance for both the production and the audience.

Chemers characterizes acting as requiring deep discipline and self-control in addition to instinct and quick decision making and it can be challenging to connect the

theory focused work of dramaturgy to performance [48]. This is especially true when rehearsal times are short and actors are inexperienced working with dramaturgs [48]. One of the most important aspects of the way a dramaturg supports performance is by providing context beyond that of the pure script and beyond what an actor could reasonably be expected to research themselves. This can include explanations of elements like period slang and in-jokes for a production of *Guys and Dolls* or what the ailments befalling Petruchio's horse in *The Taming of the Shrew* actually are, to allow an actor to make choices about how to play with the material in a more deliberate way [48]. If actors know what they are saying and doing, whether or not the audience can catch every nuance, they will pick up on humor, innuendo, and far more subtleties of the performance than if the actor was not given the support and context provided by a dramaturg.

Though I cannot hope to cover all of an art practice spanning thousands of years within a single section of a literature review, I have provided what I feel is an approachable overview to stage action as viewed by theater. Acting practice offers numerous approaches to performing and understanding stage action's relationship to both the body and mind of a performer as well as different ways of connecting to the text of a script. Puppetry provides a different, mediated view of performance, one that is sometimes seen as in tension with the text of a script due to the attention a puppet commands, as well as more direct connections to computation thanks to computation itself offering new ways of mediating performance. Dramaturgy's concern with the process of producing and performing a script refocuses us back onto the relationship between

text and performance, including ways performance can make a script legible to an audience without the deep knowledge and understanding of the script the production has, a modern framing with parallels all the way back to Zeami's concern for his audience's enjoyment. By introducing stage action and theatrical practices to computation, my goal is to advocate for a better understanding of performance as an alternative approach to interactive storytelling, both at a practical and theoretical level.

2.3 Theater and Computation

This final section is devoted to the discussion of ways computation and theater have been used together beyond interactive drama and better position my own work and computational theater in this space. Here, I look at existing work with a focus on stage action and how techniques supporting it have been used within computation as well as how computation has been brought into the theater and performance is necessary. When defining computational theater, my primary concern was with the similarities of storytelling processes in both mediums. Not everything covered in this section is primarily concerned with storytelling, that doesn't necessarily preclude it from existing within the umbrella of computational theater. I've already described the Expressive Lighting Engine as a work of computational theater, not because the system itself tells stories, but because it supports the *process* of telling stories. This section covers work in robots and AI performers, roleplay and transformation, and computationally assisted performance.

2.3.1 Robots, AI, and Puppetry

While working directly with robots is generally beyond the scope of my own work with theater and computation, the parallels robots and puppets share when brought on stage are important for helping broaden the applicability of computational theater beyond simply describing work similar to mine. The work discussed in this section primarily leans on the second half of computational theater, further understanding theater as a process for storytelling, largely thanks to the highly physical and material nature of robots. This section discusses Japanese robots on stage as described by Cody Poulton [212], Elizabeth Jochum and Todd Murphey's robotic marionettes [122], and Mikhail Jacob, Alex Zook, and Brian Magerko's work on the Viewpoints AI [118]. These three projects offer a view of the sliding scale between the more heavily theatrically focused and more heavily computationally focused.

When I say theatrically focused I mean that the way the computational elements are used is primarily in a theatrical way, not the extent or complexity of the computation on display. This is why I describe the plays that the roboticist Ishiguro Hiroshi and playwright and director Hirata Oriza have created as falling towards the theatrical end of the spectrum. These plays feature semi autonomous robots, with more recent shows featuring more human like androids, capable of reacting dynamically to human actors. The robots require no shortage of engineering and computational support, dancing in and out of the uncanny and by doing so further blur the lines of what is human and alive beyond what a traditional puppet could [212]. This question of

what is human and what is alive, while deep, is a primarily theatrical and philosophical question and less about how computation itself is impacting theater. Because of this primarily theatrical concern, I don't consider the work of Ishiguro and Hirata as falling under the umbrella of computational theater.

Between the two ends of the spectrum is Jochum and Murphey's robotic marionettes built as part of the *Pygmalion* project [122]. Rather than focusing on traditional approaches to robotics where motors articulate joints in increasingly complex ways (automating a puppet as with animatronics), the robotic marionettes focus on automating the puppeteer and using a traditional marionette with this automated puppeteer [122]. Not only does this approach to robotic control offer a new, expressive way to control a robot and allow for more expressive kinds of movement, it offers new ways for puppeteers to perform their craft and control their puppets [122]. *Pygmalion* and its robotic marionettes does fall under the purview of computational theater as it exemplifies both using theatrical practices to inform the design and implementation of computational systems focused on the performance of stage action as well as providing opportunities to further explore puppetry and performance with objects beyond the standard confines of as stage.

Also between the two ends of the spectrum of theatrically focused and computationally focused is Jacob, Zook, and Magerko's Viewpoints AI, an implementation of Bogart and Landau's Viewpoints to allow an AI actor, a projection rather than a robot, to interpret live human gestures [118]. The system formalizes seven of the nine Viewpoints of space and time (kinesthetic response and architecture still needing to be

implemented), with tempo, shape, gesture, and spacial relationship having the most associated predicates, symbols describing some discrete state of the world which is used to allow the AI's decision-making algorithms to reason about what kind of performance a human is putting on [118]. The Viewpoints AI in its complete form is an installation where a human and the AI take turns moving to collaboratively create a performance for an audience [118]. Like the *Pygmalion* marionettes, the Viewpoints AI uses computational systems to inform and, specifically, implement a computational system as well as developing a new mode of performance, directly collaborating with an AI, in front of an audience, and as Jacob, Zook, and Magerko note, the system can support new kinds of game interfaces, not just installation art. For each of these reasons the Viewpoints AI is a work of computational theater.

Computational theater is both concerned with building systems to support stage action as part of interactive storytelling experiences and using computation to explore new territory in theatrical storytelling. This is why simply putting a robot on stage is not, by itself, computational theater. Simply having a robot act as a robot on stage necessarily use computation to push theatrical practices to ask questions they fundamentally couldn't ask before. I view that approach as largely an extension of what puppeteers have been practicing for hundreds of years. Using theater to dramatically change the approach of building robots or creating new modes of interaction? That is computational theater, especially if these systems and approaches open up new kinds of theatrical performances. I've included these discussions in this literature review specifically to expand my understanding of the scope of computational theater beyond

my very specific approach to operating under its umbrella.

2.3.2 Transformation and Roleplay

Going all the way back to Zeami, roleplay is an integral part of theater that has been combined with both analog and digital computation in the form of tabletop and live action roleplaying games and video games respectively. Even with these connections, role playing games do not necessarily embrace stage action as part of having players perform their roles. This section focuses on a particular strand of research in transformative play that supports players in their roleplaying in interactive narratives using theater techniques [95, 260, 261]. If my interest in further connecting computation and theater is mostly focused on performance, the work in this section prefers to approach the connection with more internally focused goals of immersing players in characters.

What then is transformative play? Transformative play is a strand of research dedicated to understanding and supporting transformation, articulated by Janet Murray as the digital inviting players to masquerade as the roles they put on [197], as the core aesthetic of interaction [261]. Like my positing of computational theater in this literature review, transformative play does not see linearity and scripts as opposing interactivity [261]. Focusing on transformation as a part of roleplaying a character was inspired by Stanislavsky's more internally and textually oriented work on having actors become their roles [260]. Notably, this approach to engaging the player requires a well defined character for them to inhabit, otherwise they may not be able to understand

the character enough to fully experience transformation [260]. These ideas have facilitated the creation of a participatory VR theater experience with the explicit goal of giving players the experience of being an actor on stage by having a linear script for them to work with [95]. With its focus on inviting players to immerse themselves in a character using techniques grounded in Stanislavsky's method, yes, transformative play as a design methodology, at least in the form discussed here, falls under the scope of computational theater. The work in transformative play, particularly its relative lack of interest in branching and choice based narratives, skews very close to my own work with theatricality [129] and character expressivity [127, 128, 130], with the major difference being transformative play's focus on the player feeling like they are someone else or in another world while my work tends to focus on performance as an aspect of storytelling and character.

2.3.3 Computationally Assisted Performance

This final subsection returns to bringing computation into the theater with computational performance, defined as the middle ground between traditional performance and purely digital storytelling [231]. While there is certainly overlap between computational theater and computationally assisted performance, my main purpose here is to highlight the distinction between the two areas even with all of their overlap. Computational performance makes an assumption of computational referring to the digital, whereas computational theater makes no supposition that computation must be digital. This difference in views on computation is in turn where much of the distinc-

tion between the two areas comes from as the form of computation is much more of an aspect of computationally assisted performance while computational theater is more concerned with the use and place of computation in connection to performance. This subsection discusses three computationally assisted performance experiences, *Coffee: A Misunderstanding*, *Séance*, and *Bad News*.

While I characterize computationally assisted performance as focusing more heavily on the form of the computation (being digital) than how I've defined computational theater, computationally assisted performance still has a relatively broad view as shown by the three aforementioned examples taken from Samuel et al's taxonomy in "Computatrum Personae" [231]. *Coffee: A Misunderstanding* is a group performance where two audience members are brought on stage and their dialogue is chosen by the remaining audience and facilitated by a host [231]. As a theatrical experience enabled by computation that is built around audience participation above all else, I would characterize *Coffee: A Misunderstanding* as a work of computational theater in addition to being computationally assisted performance as the presence of computation allows for untrained actors to still perform in an engaging way for an audience. *Séance* is a three player physical puzzle game where participants are given character roles and work together with a spirit medium to solve puzzles as a wizard behind the scenes triggers cues to transform the everyday looking objects in the game to unlock new puzzles [231]. While *Séance* does rely on the acting skills of the person playing the spirit medium, it is primarily a puzzle game and the interaction between the computational elements and players is basically one way, as feedback and gating for further progression in the

narrative. With this in mind, I don't consider *Séance* to be a work of computational theater. The experience primarily uses stage action to frame the gating of puzzles and the way it uses computation falls more in line with stage craft than providing a new perspective our collective understanding of theatrical storytelling. *Bad News* is a combined social simulation and theatrical improvisational experience where players interact with the residents of a procedurally generated town to inform a particular person of a death in their family [232]. The conversations players have with the residents are facilitated by an improvisational actor and are then used by a wizard, hidden from the players, to update the social simulation and give the actor new information and context for their performance [232]. While *Bad News* and *Séance* share similarities in approach, I call *Bad News* a work of computational theater unlike *Séance*. The reason being, that the social simulation in *Bad News* is a new way of supporting improvisational performance rather than acting primarily as part of puzzles for players to solve. I have spent much of this literature review lamenting the lack of focus stage action receives compared to plot action. Suddenly saying *Bad News* (and even *Coffee: A Misunderstanding*) are works of computational theater may seem contradictory to that, as they both allow participants to change the direction of the plot. With that said, they both position themselves as improvisational and, importantly, both use computation to change the way improvisation is performed.

2.4 Summary

Well now that I've gone in three fairly distinct directions in this chapter, let's do some sense-making. One of my argumentative goals in putting these disparate seeming sections together is to juxtapose the way theater practitioners discuss their craft with one of the common ways computation has codified its usage of theatrical theory. I do this both to highlight the novelty of my own work and to emphasize, as much as I can, one of the reasons I see for one of the common approaches to interactive narrative focusing so heavily on structure. My ultimate purpose with this dissertation is to highlight another avenue of interactivity we as practitioners and designers in computation have access to.

This ultimate goal is the reason for the third section in this chapter as well. I'm not the only person or researcher pursuing computation and performance, even if I would say I'm the only one pursuing it using the exact means I describe in this dissertation. The pragmatic reason for my inclusion of all this work is to again position myself as doing something unique even within the space of computation and theater. My more high minded reason is to show that my project of defining computational theater as a practice and approach to research extends beyond just me and my idiosyncrasies. It is in fact something being practiced right now, even if we didn't notice or have the language for it!

This literature review is also designed to help explore the implications of this definition of computational theater and contextualizing its relationship with both traditional theater and existing work in computation drawing from theater

This chapter is also here to contextualize computational theater's relationship with traditional theater and computation, as well as further explain why I am not working purely in interactive drama. To this end, I provided overviews of acting and puppetry practices as well as dramaturgy's role in supporting performance. Through my exploration of these aspects of theatrical production, I wanted to expose the intricacies of how performance is a creative process in it of itself, even when fully constrained by the structure of a script.

Additionally, I wanted to contrast the development and philosophy of interactive drama, its roots firmly in Aristotelian dramatic theory, with understanding that performance practices have of their craft. Play scripts do not branch in the same way that interactive drama and many other computational approaches to interactive storytelling take. Yet we as members of the audience will still go and see a new production of a play thousands of years old, written for a completely different cultural moment. We can still be surprised and intrigued by stories we have experienced dozens or hundreds of times even if the words don't change. This quality of theater fascinates me and is really what I see as the project of computational theater: to bring the understanding of how to make the old feel new from theater into the world of interactive storytelling as computation understands it.

Theater is a primarily human and analogue form of storytelling that none the less is built upon creating repeatable processes to tell a specific version of a story and keep that story feeling alive. Even if the show itself is not itself hugely interactive beyond an audience's ability to interpret its elements and their presence in the same

room as the cast, the process of rehearsal and performance for the members of a production is intensely interactive and creative. Computation in the form of interactive drama frequently focuses on the narrative arc of a story as the site of play rather than the processes in theater which already do alter the understanding of a script. When interactive drama uses stage action, it is primarily to highlight the changes in narrative arc and structure. Looking at other work connecting theater and computation provides some glimpses into the possibilities of shifting our focus away from plot structure and towards playing within a plot structure using stage action, often by embracing elements of the Viewpoints philosophy: that whatever happens on stage is the performance, even if that embrace isn't explicit.

Interactive storytelling in general, when drawing from theater, has mainly focused on writing and acting, both of which can provide valuable foundations for interaction. When operating in a purely digital realm, if we are asking a player to embody and perform a character, we are asking them to perform stage action and ultimately put on a mediated performance. Acting does deal in mediated performance but it is just one element of acting. Puppetry is mediated performance. My goal with this chapter and dissertation is to not just call for more usage of stage action as part of interactive storytelling, but to expand the understanding of performance beyond acting. Puppetry and digital character performance have numerous parallels, described by both puppeteers [265] and roboticists [122]. Broadly, by putting forth the notion of computational theater, I hope to not only bring stage action into a more prominent role as part of interactive storytelling, I hope to encourage drawing from more diverse

aspects of theater to build foundations for that exploration of stage action.

The umbrella of computational theater fits more than just me underneath it and I hope I can make it inviting to work underneath its ideas. Much of the rest of my dissertation is devoted to delving into what it means to make work using computational theater as the guide, both at the theory and development level. From the vantage point I carved out here, I hope to have given you enough grounding in the way theater understands enaction as a creative process to follow me into deeper discussions about narrative design and system building.

Chapter 3

Theatricality in Interactive Storytelling

What does it mean to be theatrical outside of the theater? This is the question that frames all my work in this chapter and the two that follow it. My interest in this topic grew out of a combination of generally feeling dissatisfied with the way theatricality was commonly defined through presence in the same space. We colloquially describe lots of other media as theatrical even if it's recorded and mediated in a different way than theater. But what does that mean and what are those aesthetics? Having an answer to that is part of my goal with my work here analyzing existing games.

My answers here are, in retrospect, deeply in conversation with Sarah Bay-Cheng's [20, 21], and by extension Heiner Goebbels' [87], writing on mediated performance. Much of the ink spilled on theater's relationship to the digital world and performance is laced with anxiety about the loss of live performances in shared spaces during the height of the pandemic shutdowns. We are now in a post-shutdown if not post-pandemic era. Theater has returned to its shared physical spaces, but the ques-

tion of whether co-location is necessary for something to be theatrical, let alone *theater* remains. Bay-Cheng concludes that mediated performance and digital performance is certainly a disruptive force to the traditional understanding of theater, but that what it ultimately means for theater is still being written, as, with disruption comes the potential to wholly expand theater's understanding of itself [21]. Next to this space of possibilities is where I find myself with computational theater.

My discussion in these next three chapters views theatricality as distinct from shared physical presence in a positive light. I must also point out the distinction between presence and liveness. Both concepts share an overlap but one can have liveness without co-presence, for example video games, or presence without liveness, for example film. Disentangling these concepts is beneficial to both computation and theater. Viewing computational systems as something live allows us to examine these systems through the lenses of theater and gain new insight about how these systems engage their players.

Performance practices and the theatrical production process are intensely creative in how they tell stories, but much of this creativity is left implicit behind what an audience sees: the final polished production night after night. By understanding liveness and presence as two distinct properties of the theater, we can expand the stage beyond the confines of a theater. We can create works of theater which open up the mysteries and joys of the production process to people who would otherwise be unable to experience them. With computational theater, I hope to make computation another avenue for theater to continue to develop its storytelling practices and be able to play with a broader idea of liveness than shared presence in a theater. Computation provides

another place for the interactive and interpretive aspects of performance and theater to flourish.

I don't claim that what I have already built for my dissertation or what may come after is a replacement for live, co-located theater. In fact I think anyone claiming to be able to replace one art form with another is engaging in fallacy. My work here exists as much to support theater scholars in affirming their knowledge is applicable beyond the confines of their own discipline as it does to expand computation's understanding of interactivity.

My goal with this dissertation is, as many performance practitioners and scholars put it, about finding the new in the old, as applied both to computation and theater. Computation is in fact quite good at facilitating enaction and supporting its players in performing the ritual of interacting with systems. Theater's wealth of knowledge surrounding the interactivity of stories and storytelling are applicable far beyond the stage even where this applicability may not be obvious.

Ultimately what I'm saying here is that it's going to be okay. Theater has survived thousands of years through many an existential threat because there truly is no replacement for sharing an experience with our fellow humans in a single space. Likewise, the new modes of storytelling and play that computational systems has enabled will not be going anywhere because I am choosing a different direction in my work. What is our job as researchers if not to experiment and create new things in conversation with the old? Maybe this comes off as a bit of an artistic perspective in what is engineering work at the end of the day, but that's what computational media is: a blending of the

artistic, humanistic, and scientific perspectives. And besides, to understand my work, I see it as necessary to look at engineering as art and art as engineering.

The rest of this chapter covers more specifics about my view of theatricality and its relationship to narrative design. I begin with defining my concept of theatricality. I then discuss its relationship to existing commercial games and computational narrative work. Finally, I perform a close read of the narrative roguelike *Hades* paralleling the progression structure of a theatrical production to demonstrate the applicability of theatricality as a lens for narrative design.

3.1 Introduction to Theatricality Beyond the Theater

One ongoing strand of work in game studies seeks to advance our collective understanding of how videogames make meaning by analyzing what games share with theater. Much of this analysis has centered on the importance of *live performance* as a shared feature between both art forms. These analyses position the player of the videogame as being roughly analogous actors performing on stage. But is this the only way to understand videogames as theatrical?

In my view, liveness and co-presence of the audience in the same physical space as the performers is certainly part of what makes theater unique. However, I contend that, independent of live performance, theater is also characterized by its need for *production*. The processes of theater are built upon the creation of a ritual to tell a particular story in a particular way [48]—a ritual that is necessarily repeated by the

actors both prior to (in rehearsal) and during (on stage) the run of a play. It is on the construction of meaning through the theatrical ritual, and the continual *reinterpretation* of a play's meaning that this process implies, that I focus here.

With this in mind, I define *theatricality* as a property of creative works that repeatedly reinterpret and recontextualize themselves over time as and as part of the act of experiencing the work. This definition of theatricality stands in contrast to most established definitions of theatricality, which tend to focus on what does and does not happen on stage during a single performance of a play. From a playwriting perspective, a script is theatrical insofar as it orchestrates the live performance of character action [242]. Meanwhile, from a dramaturgical perspective, theatricality "takes advantage of those qualities of the theater that no other medium can reproduce" to highlight and support, through performance, the moments of change defined within the script [48]. Under either of these conventional definitions, for something to be theatrical, it must be performed live and in front of an audience.

In my dissertation work, I argue, however, that a key aspect of theatricality—which these conventional definitions overlook—is the continual reworking of a play's meaning that takes place between the company members during the theatrical production process. From this perspective, much of the work of theatricality takes place between performances and over the course of multiple performances and productions; what happens off stage, or on stage in the absence of a full audience, must also be considered.

By de-emphasizing the need for live performance on stage, I seek to instead call

attention to an aspect of theater practice that often goes under-discussed when theater is used as a lens to analyze videogames. That the work and pleasure of reinterpreting and recontextualizing a narrative through play is itself interactive storytelling. This interaction can be a more traditional production of a script or, as I argue in this chapter, and ultimately dissertation, a mode of audience interaction with an interactive narrative beyond changing the story structure or performing a role.

Videogames, which frequently adopt a cyclic narrative structure, seem especially prone to exhibiting this form of theatricality. However, not all videogames are created equal in this regard: though all games are subject to a degree of *external* theatricality imposed on them by repeated play in varying real-world contexts[1], some games also make use of *internal* theatricality. These games deliberately present and re-present a core experience to the player in a gradually evolving way, allowing the player to sample repeatedly from the story volume [93] of possible playthroughs of the core experience and refine their understanding of the game's meaning through reflection on the variations they have seen.

To better understand how this internal form of theatricality operates in games, I perform a close, dramaturgical read of the internally theatrical videogame *Hades* [255]. By adopting a view of theater that focuses on the theatrical production process, I gain insight into how meaning-making strategies from theater, beyond the use of live performance, may operate in games. Additionally, this new perspective allows us to

[1] As discussed in Mitchell et al's work on replay, for instance [191]. For example, playing *Freespace 2* [271] in the United States when it was released in 1999 and in 2019 highlight very different aspects of its thematic priorities.

better understand the process of reinterpretation and recontextualization internal to the play experience of a theatrical game.

Hades is an isometric narrative roguelike game released by Supergiant Games in 2020. The player controls Zagreus, a son of Hades, as he attempts to escape the underworld using a selection of mythical weapons and boons granted by the Olympians. Like many roguelikes, *Hades* exhibits a cyclic structure and a process-oriented (rather than a results-oriented) gameplay focus: the player repeatedly tries, fails, and tries again to escape from the underworld, hopefully gaining ground on each successive playthrough as they acquire the skills needed to make further progress. *Hades*, however, goes further than other roguelikes in its embrace of theatricality. Its narrative meta-progression revolves primarily around a sequence of major recontextualizations of the core gameplay, and the player's motivation to keep playing is tied in part to this evolving narrative context. The structure of *Hades*, both in its narrative and its usage of procedural elements, draws on theatricality as we have defined it. At a high level, the structure of an individual escape attempt remains completely fixed from one attempt to the next—but the context and meaning of these escape attempts is shifted dramatically by two major turning points in the narrative metaprogression, and more subtly by the player's choice of gear, short-term goals, and recent interactions with characters in the House of Hades prior to the start of each attempt. As the game's main narrative unfolds, what begins as a story about family drama eventually becomes a story about the cast of a play discussing their thoughts on the show they are performing every night. With this shift in narrative focus, *Hades* repeatedly changes its answer to the production question

of "why this play now?" [48].

Hades is not the only game to use a looping structure, nor does it draw as many direct parallels to theater as other commercial games. However, its dynamic, player-directed progression is what distinguishes its narrative structure from other games using this theatrical aesthetic in their narrative design. For this reason it is important to situate *Hades* within this broader context.

3.2 Other Theatrical Videogames

Nier: Automata [209], an action RPG released in 2017 by Platinum Games, makes extensive use of theatricality. In Routes A and B, the player first experiences the events of a protracted war between machines and androids from two different characters' perspectives. Then, from the third playthrough onward, the game turns the idea of repeatedly replaying the same events on its head and continues the story with a much more rapid-fire shifting of character perspectives. Events foundational to *Nier: Automata*'s plot were written as a stage play in 2015 by Yoko Taro, the game's director, [219,277] with newer versions slightly changing characters and events [218], foreshadowing the eventual game being built upon recontextualization of its narrative structure and environments. The game's focus on slowly recontextualizing character roles is at its most condensed in the form of the PODs, small rectangular floating robots with arms that accompany the player characters. They begin the game as rather innocuous communications-and-utility devices, later becoming justification for why resources are

shared across characters, facilitating much of the perspective switching that makes up the latter parts of *Nier: Automata*'s narrative, and finally becoming Rosencrantz and Guildenstern-esque [248] characters making sure the two protagonists do not cross each others' paths until the end of the game.

Pathologic [112, 113], a survival-horror game originally released in 2005 and remastered in 2015 by Ice-Pick Lodge, opens in a theater. Its three playable characters first discuss their approaches to dealing with the plague infesting a town, then set off and spend the next twelve days investigating its source. While the game's systems offer a fairly detailed economy and player state simulations, it is incredibly forward about how artificial everything is. Each in-game day ends with a brief theatrical performance foreshadowing future events, though how much these plays make sense is heavily influenced by which characters have been played up to that point. The choice of playable characters determines the player's relationship to the town, most importantly whether or not they are an outsider, drastically changing which parts of the town's culture they are privy to as well as the general arc of the plot they are participating in. This includes the answer to the source of the plague. Whether the town or the strange structure on its outskirts are the root cause depends on whose perspective the events were seen from. Even though each of the three playable characters' stories exist in parallel, whichever the player chooses has their perspective ascended to the reality of the plot.

VA-11 Hall-A [251], a visual novel bar-tending game released in 2016 by Sukeban Games, may not have the direct, explicitly acknowledged theatrical influence of *Nier: Automata* and *Pathologic*, but its structure is built around routine in much the

same way that *Hades*'s is built around performance. Every day for Jill, the player character, begins with her arriving at the titular bar before it opens, queuing up the day's music before both halves of her shift start, and ends with her relaxing in her apartment and potentially buying something for decoration. Even on the day the bar is shut down during an eruption of police violence, the flow of the day remains the same as the context of other characters visiting the bar shifts towards finding a safe place off of the street and not just somewhere to relax. While the routine *VA-11 Hall-A* establishes can be a source of comfort, once Gabriella, Jill's ex's sister, starts coming to the bar, the routine becomes a source of anxiety until Jill can no longer avoid having a painful conversation with her.[2]

This selection of games serves to illustrate that some videogames (including *Nier: Automata* and *Pathologic*) have deliberately positioned themselves in conversation with theater. Additionally, I hope to highlight that even games which do not explicitly invoke theatrical influences can still use theatrically cyclic narrative structures to realize their aesthetic goals. In each of these games, repeated re-expression of the same narrative skeleton while varying some details of content or presentation is essential to the creation of the intended player experience. *Hades* strikes something of a balance. The game has a degree of overtly theatrical inspiration, thanks to its usage of Greek mythology, and the slightly more common style recontextualization that comes from familiarity with a location or routine in a videogame. After *Hades*'s credits roll, its characters do talk about putting on a show for the other gods, but there is nothing

[2] *VA-11 Hall-A*'s presentation of a comfortable routine being upended by personal drama is almost the direct inverse of *Hades*'s narrative recontextualization.

quite so forward as the Greek chorus-style POD interruptions in *Nier: Automata* or the literal theater in *Pathologic*. That said, the game's usage of the House of Hades and the other realms of the underworld are reminiscent of a theater's layout, discussed in more detail through the close read itself.

3.3 Related Work

Theatrically influenced analyses of videogames have tended to focus either on the player's role in shaping the narrative structure of an interactive story [152, 167, 197] or in their performance and participation within an experience [130, 260, 261]. Even when the way games can recontextualize themselves has been discussed [13, 138], the focus has been on the phenomenon present in the games themselves rather than on the act of recontextualization and reinterpretation the games are performing.

3.3.1 Performance

The common ground between my view of theatricality and the work emphasizing performance that Tanenbaum [260, 261] and myself [130] have done lies with the understanding that reinterpretation and recontextualization are integral components of the theater production process as a whole [48]. Some amount of reinterpretation is necessary to avoid performance becoming an act of pure reproduction [158, 223]. Something that the above papers gloss over concerning Stanislavsky's work is the importance of an actor's ability to recontextualize themselves within any given performance, even if the role and the production itself remain the same, these are an actor's adjustments [158].

Both Tanenbaum et al. [261] and some of my early work [130] focus more on the mechanics and emotional side of actor's adjustments being a part of a performance. While both strands of work discuss the collective nature of the way these adjustments function on stage, they do not take the perspective that these adjustments are interpretive work as much as performative.

The Viewpoints as described by Anne Bogart and Tina Landau have largely been used in computational system design to help mediate performance with a computer [118,127] while one of the more subtle goals of the practice are ignored: collectively, as a theatrical production, being able to explore the question of "what is [the production]?" [29]. In other words, how does the production's understanding of what they are making change along with the collective context of that production? While the Viewpoints is largely focused on the rehearsal and training process, there is an understanding that interpretation and reinterpretation are integral to those processes. Furthermore, that the practice of interpretation and reinterpretation should be a collective act, similar to the Levins' characterization of actors' adjustments must be made in partnership with the other actors involved [158].

While I have pointed out Zeami's focus on the audience experience as one of the reasons for inclusion in my theoretical basis for a game interface [130], I only briefly touch on what he says about a Nō performance's need to react to an audience's energy level while staying within the necessary theatrical structure [223]. Zeami uses the example of nobles arriving late to a show as one of the ways an audience's energy can become out of tune with the production. He goes on to describe the delicate

navigation necessary to repair this disconnect and notes that there simultaneously must be a change in the performances to accommodate this while not allowing the show to slide all the way backwards into its initial state. This understanding of the audience and performers' symbiotic relationship is in contrast to the more the more commonly cited theatrical practices in computation show little concern for the audience by comparison. Zeami's discussion of altering a performance in response to the audience provides another potential avenue forward for interactive storytelling in treating players as someone to be in conversation with about details that exist within the larger game structure, fitting in with Brenda Laurel's view of the user of a computational interface as existing somewhere between an actor and a member of the audience. In fact *Hades*'s Pact of Punishment, a difficulty customization interface introduced after the first successful escape, is a vehicle for doing much of the kind of performance adjustments within a structure as discussed by Zeami, Bogart and Landau[3], and Stanislavsky.[4]

3.3.2 Time, Structure, and Distance

Both Aytemiz et al. [13] and Kleinman et al. [138] are, at a high level, concerned with analyzing how games use time as an avenue of expression. When discussing changes in systems, Aytemiz et al. are primarily concerned with understanding how games can use these changes in a way endemic to the form and aid other designers in taking inspiration from the patterns observed in their case studies. Kleinman et al. build

[3]Even if the Viewpoints approach, when compared to other theatrical methods, deemphasizes the text associated with a production.

[4]Other Supergiant games [253, 254] have had highly customizable in-game difficulty systems, but these were not foregrounded in the same way as in *Hades*.

a taxonomy of rewind mechanics and are primarily concerned with understanding the affordances for both designer and player different approaches provide. Our work in this paper is also about how games can use time as a storytelling tool, as a way of building a context for players which can then be altered to give new perspectives on the narrative or even the entire game's structure and encourage reinterpretation through recontextualizaiton.

While Ben Samuel's usage of Brecht's approach to theater mostly focuses on shared authorship, his general discussion of those ideas relative to computation [230] are relevant to our own application of theatrical ideas to videogames, particularly using the alienation effect to further explain the workings of the theater or a game. When describing the general function of a *learning-play*, as apposed to more traditional Aristotelian plays, Brecht says "the learning-play is essentially dynamic; its task is to show the world as it changes (and also how it may be changed)" [33], a sentiment my definition of theatricality shares. I can even go so far as to say that the alienation Brecht discusses is a common element of the games we reference in this paper as being theatrical. From the end credits marking *Hades*'s shift from family drama to performing for the Olympians to *Nier: Automata*'s eventual constant perspective changes to *VA-11 Hall-A*'s occasional conversations outside of the bar featuring dramatically different interfaces, each of these games recontextualizes the act of playing them multiple times.

Brecht's usage of alienation, and by extension the changes in context we are discussing in games, falls under the umbrella of changes in *aesthetic distance*, how emotionally or critically engaged a person is with a piece of media [48]. Generally

speaking, calling attention to the artifice of an experience will increase the aesthetic distance (and push a person to engage more critically) and powerful emotional moments and other methods of inviting suspension of disbelief will decrease the aesthetic distance (and push a person to engage more emotionally) [48]. I have previously described these shifts in perspective (as moving between the acting and editing levels of interaction), even within a singular character's perspective, as an inherently Brechtian move [130]. Not all changes in aesthetic distance or usage of Brechtian alienation will result in the full theatrical recontextualization I describe here. That said, the games that engage with this aspect of theatricality use extreme swings in aesthetic distance to prepare their players for the eventual recontextualization of their narratives.

Aristotle, in his definition of what makes a whole and complete play makes an important distinction between the *story* of a tragedy and its *plot* [26]. The *story* can include events taking place outside of a play, for example Zagreus's mother's departure from the underworld in *Hades* or the initial war with the Machines in *Nier: Automata*. Whereas the *plot* is only made up of the events within the play and their ordering. Also important to Aristotle is the idea that the playwright does not need to be responsible for creating new *stories* but is instead only responsible for structuring a *plot* and making the play understandable to someone who has never experienced the *story* before. It is through this view of playwriting we can further pry into what these theatrical games do. I even posit a degree of harmony between Aristotle's and Brecht's views on the purpose of theater: that these moments of recontextualization and alienation in games are frequently facilitated by careful placement of exposition about *story* events. In other

words, playwriting. For example the revelation about why Zagreus's mother is hiding from the Olympians (*story* information) drives the performative context *Hades* takes on after its credits roll.

Ultimately, the interpretive and contextual work done in theater must be done, and, as Chemers says, "the question is merely how it gets done, and by whom" [48]. Performance entails its own brand of this as Stanislavsky, Bogart and Landau, and Zeami describe. I argue that games can be a tool to bring players into this artistic conversation. Some even have. *Hades* takes this one step further and allows players to participate in some of the interpretive adjustments in much the same way performers do.

3.4 Theatricality in *Hades*

"There is no escape", *Hades* promises the player when they first start the game—and this promise holds long after the credits roll. This is not to say, however, that nothing changes in the story or the world. In fact, there are two key turning points at which the game significantly recontextualizes its initial promise: upon Zagreus's first successful escape and upon his tenth. When we first meet Zagreus, his anger is directed at his father and the fact that he is not allowed outside of the underworld. Finally defeating his father and meeting his mother, Persephone, for the first time reveals that Hades (both the game and the character) was right: Zagreus cannot leave, not because his father has forbidden him from departing, but because a still greater force binds

him permanently to the underworld. Here, Zagreus acquires a new motivation for his escape attempts: each successful attempt gives him an opportunity to talk to his father, who only seems willing to say anything after being bested in combat, and his mother, who can only reveal so much in their brief time together. After besting his father for the tenth time, Zagreus is able to convince Persephone to return to the underworld, but introduces a new problem in the process. Through their brief conversations, it is revealed that the gods of Olympus do not know where Persephone is—and should they learn of her life in the underworld, they would exercise their wrath on the House of Hades. There is, however, a solution for this problem. To keep the Olympians, who have been assisting Zagreus in his escape attempts, from realizing anything is wrong, Zagreus must continue to attempt escape as he always has and put on a *show* for his audience. *Hades* keeps its promise: the player is stuck in the same loop they have been in since starting the game, only now, rather than attempting to escape for his own self-interest, Zagreus is doing so to allow the process of his family's healing to continue.

3.4.1 Starting a Save

Upon starting a new save file, *Hades* immediately drops the player into their first escape attempt. The player quickly surmises that Zagreus is in the underworld and attempting to leave, but no further context is given. On this first "table read" of *Hades*'s core experience, Zagreus and the player muddle through as well as they can, trying to learn the controls and contend with increasingly dangerous enemies until they succumb to the cumulative effect of many small failures. Jumping into group dynamics and feeling

around for something to latch onto is very much a part of the Viewpoints training [29], and the cold opening of *Hades* implies a similar attitude towards experimentation and failure: what is ultimately important is growing an understanding of how everyone involved in a production works together.

It is only after this initial, inevitable failure that the player is allowed into the House of Hades—and to talk to other members of the cast outside the context of an escape attempt. Many of the game's characters reside within the house, and most can be engaged in conversation to learn more about the game's world and backstory, develop relationships, advance specific subplots, and acquire additional perspectives on the game's major themes. These characters' remarks tend to complicate the player's understanding of the game world. Additionally, through growing entanglement with these characters, the player can acquire additional short-term narrative goals [41] (such as learning more about a particular character, or deepening a particular relationship) that can help to shape the motivation of each escape attempt.

This is also where the player is allowed to start making longer term investments into their character. *Hades* offers two major outlets for metaprogression. First is the Mirror of Night, a character upgrade interface where the darkness resource (a room-clearing reward) can be spent making Zagreus slightly more powerful, provided by Nyx (Zagreus's supposed mother). And second is the house contractor, who can both make the House itself more pleasant (with various cosmetic upgrades) and alter the different areas in the underworld in small ways to make each area occasionally more forgiving. Spending darkness on Zagreus to make him more powerful may be a fairly standard

videogame progression system in the abstract, but its placement within the house allows it to become a tool for theatricality as well.

Due to *Hades*'s nature as a roguelike, these more permanent upgrades can only be made outside of a run—yet the resources used to perform these upgrades are collected during an escape attempt. Every escape attempt allows the player to become slightly more familiar with the areas of the underworld and their denizens, and this knowledge can be carried forward into the next run, much the same way as rehearsal allows members of a production to familiarize themselves with group dynamics and collectively make interpretive decisions [29, 48, 158, 223]. We can then view the collection and spending of darkness in *Hades* as the game systems embodying and solidifying this element of theatrical production, in a gesture to make every escape attempt carry some amount of growth.

The courtyard Zagreus passes through to begin an escape attempt functions much like a prop table, being home to the weapons and keepsakes that can aid Zagreus in his fight through the underworld. Keepsakes are equippable items that grant small bonuses, ranging from extra health or damage to a guarantee that a specific god's boon will be the first one found in an area. They are obtained by giving gifts of nectar (another chamber reward) to any of the characters who can be spoken to. Only one may be used at a time, though they can be swapped out between areas if the contractor has been paid to install the necessary infrastructure. These keepsakes can then be used to pull an escape attempt in certain directions. One of the simpler strategic uses of keepsakes available early on in the game is to select keepsakes belonging to the

Olympians whose boons the player enjoys using, in order to make these boons more reliably available. Keepsakes are one of *Hades*'s ways of allowing the player to make adjustments to their escape attempt in the same way Stanislavsky describes [158], and importantly, the possibility of choosing different keepsakes reinforces the idea that the game and player share the responsibility of deciding how a given escape attempt should feel. Even the simplest effects—like that of Cerberus's collar, which gives up to an extra 50 health[5]—can dramatically change the flow of a run by allowing a player to prioritize rewards other than health upgrades at certain points, potentially enabling the player to gather a much more powerful set of boons earlier in the run.

The six weapons (sword, spear, shield, bow, gauntlets, and machine gun) in *Hades* have some of the most obvious and profound effects on the way an escape attempt will feel. Much of the difference between the weapons lies in their move sets and by extension their effective ranges, with the gauntlets having the shortest range, and fastest attacks, and the bow the longest range, and slowest attacks. Though the player is making this initial choice, the game will offer its own alterations to the way weapons feel through the two Daedalus hammers available per escape attempt. Each hammer has up to three upgrade options which can be simple upgrades to the damage of certain moves to switching a move out entirely. For example the gauntlets have an uppercut by default but this can be turned into a medium range projectile. Again, these upgrades put the player and game in a conversation about how this particular escape feels, especially early in a playthrough when exactly what each upgrade does is a mystery and, like

[5]Keepsakes can be upgraded twice, with upgrades gated by how many rooms have been cleared with the keepsake equipped.

making a major change in the middle of a performance [223], carries with it a degree of risk.

All of these elements within the house give it a similar feeling to the green room and backstage of a theater, where the cast can celebrate or lament how the last performance went and then prepare to do it all over again.

3.4.2 A Successful Escape

The first successful escape attempt is much like the first off-book[6] run-through of a production. The play as a whole remains a work in progress, but the core structure is now solidified enough to permit a freer form of experimentation with its presentation. In *Hades*'s case, reaching this milestone requires the player to successively defeat each area's boss: Megaera the Fury (or one of her understudies) in Tartarus, the bone hydra in Asphodel, Theseus and Asterius in Elysium, and finally Hades on the surface. The first successful escape is also when the structure of the game is fully revealed to the player. The three main underworld areas (Tartarus, Asphodel, and Elysium) are generally between ten and twelve chambers long, with a required mini-boss fight appearing toward the middle of each area's progression and one of the aforementioned boss fights (followed by a brief respite) at the end. Unlike the previous three areas, which are linear successions of chambers, Styx is a hub branching off into five sequences of small rooms; one of these sequences offers an item that enables the player to proceed to the surface once acquired, while the others culminate in a mini-boss fight. However, the contents

[6]When actors no longer need to reference the script during rehearsal.

of each sequence are semi-randomized, making it unclear to the player which sequence holds the item needed to continue. Additionally, the player is returned to the hub after clearing each sequence of rooms—enabling them to delve back into the other sequences even after they obtain the item necessary to reach the surface, if they so choose. At some point, however, the player will inevitably either find the item or exhaust all the rooms, at which point they have no choice left but to proceed to the surface—where the grueling fight with Hades waits.

Upon his defeat of his father, Zagreus is greeted with a beautiful sunset before managing to find Persephone, his mother, tending to her garden. Their reunion is brief, as it is revealed that Zagreus cannot survive on the surface for long, and he is dragged back down to the House of Hades—but not before Persephone asks him to come and visit her again. When he walks out of the pool of Styx this time, Zagreus is given what passes for a congratulatory welcome in the House. What accompanies this victory is a sizeable shift in Zagreus's motivations: rather than wanting to run away from home, he is now primarily motivated by the desire to understand what happened between Hades and Persephone that resulted in their separation.

Also accompanying the victory is the Pact of Punishment placed over the door through which Zagreus passes at the beginning of each escape attempt. This rather intricate difficulty adjustment system serves many purposes. The first is simply further emphasizing the magnitude of Zagreus's triumph over his father with a large, angry scroll. Second, the Pact allows the acquisition of more rare upgrade materials: titan blood for upgrading and altering weapons, diamonds for the more impactful and narra-

tively important contractor projects, and ambrosia for further deepening relationships with characters, rather than the sizeable darkness reward for repeatedly defeating a boss. The catch is that, to continue earning these awards, the player must continuously increase the difficulty of the game after each successful escape with a weapon. With this system *Hades* encourages players to gradually experiment with the difficulty to make escape attempts novel each time, while also making each successful escape at a new difficulty level carry the same player and in-game progression that darkness was used for early on. The Pact's gradual, conversational attitude towards difficulty is, again, similar to members of a theater production encouraging each other to experiment and try new approaches within the structure of the show being rehearsed [29]. Finally, the Pact's introduction marks the point in *Hades* where all the pieces for the player and game to converse about the nature of an escape attempt have been introduced, making the subsequent escape attempts (until the game's credits roll) analogous to a production finally getting to make use of the actual performance space rather than the facsimile from early in the process.

By the first successful escape, Zagreus and the player have likely been able to build some amount of rapport with the other named characters through occasional gifts of nectar, mirroring the camaraderie built through the rehearsal process [29], though these relationships are far from settled. Megaera and Thanatos (if he's appeared) are still resentful of Zagreus's attempts to leave, and even the more helpful characters (such as Nyx and Achilles) are only as open about their thoughts as they need to be. But these difficulties are relatively minor, and can generally be solved with time and more

conversations over nectar. The real problem, and why this portion of the game still feels like a rehearsal (in the context of an endlessly repeatable experience), is that any further conversation with Hades and Persephone requires Hades, and by extension Megaera, the bone hydra, and Theseus and Asterius, to be killed. Even then, the progress made during each conversation with Zagreus's parents is rather slow, and these conversations can be few and far between depending on how well the repeated escape attempts are going.

This is also the part of the game where the performative nature of each run lessens a little. For now, there is a pressing problem to deal with that both the player and Zagreus want to see resolved—and consequently, this is where the aesthetic distance between the player and Zagreus is at its smallest. The player is comfortable enough in the role they've been given that they are no longer just practicing the motions—and, as a result, they can now be caught up in the family drama. At this stage of the game, each escape attempt is motivated primarily by the goal to learn more about the story, rather than a desparate desire to be free of the underworld. As we now know, escape is impossible.

3.4.3 Reaching the Credits and the Performance

Persephone agreeing to return to the House of Hades is the sign for the game's credits to roll. While there are still tensions between the family members, and especially between Hades and Zagreus, they have now come to something resembling an understanding. There is another reason Persephone agrees to return and it is connected to the

question of who each section of *Hades* views as its audience. The Olympian gods who have been helping Zagreus from a distance all this time do not know where Persephone is and when the player and Zagreus learn the details of how she came to be in a relationship[7] with Hades, this becomes a secret in dire need of protecting. The solution: to convince the Olympians that nothing has changed, Zagreus must indefinitely continue to attempt to escape the underworld just as before. In this way, the Olympians become the audience of this (now very well-rehearsed) theatrical production.

Who, then, was the audience for the other, pre-credits, version of the narrative? The player. I characterized the family drama as decreasing the aesthetic distance of the player and the game because at this point, even with its highly cyclical nature, *Hades* is still telling a semi-traditional game story with an ending of some kind being foreshadowed. The player attempts to escape and is rewarded with more insight into Zagreus's family drama or more conversations with the other members of the House of Hades, often consoling him in his failure. The player as the audience for this story in fact mirrors the way that the audience during rehearsals, and even to an extent during the actual performance, are the other members of a production [29, 223].

There is also the matter of motivation for the performance beyond the credit roll: after all, the lure of further family drama no longer exists. The Olympians, for all their help, do not dwell on the details, and as a result, the reason for Zagreus's very first escape attempt can once again become the basis for a show. The post-credits performance for the Olympians' benefit can then be characterized as highly polished,

[7] In *Hades* Persephone already wanted to leave Olympus. Zeus, in an attempt to help his niece and show some gratitude towards Hades, sent Persephone to be Hades bride in secret.

well-rehearsed versions of that initial clumsy attempt to simply run away from home to spite his father.

Everything that has happened up to this point—the family drama, the Pact of Punishment, growing closer to the cast—was all in preparation for this play for the gods. By now Zagreus and the player are comfortable, even happy to be in their roles, and adding new challenges becomes part of the reason to keep performing. Increasing the game's difficulty is no longer something in the way of a potential new revelation, it is just a part of maintaining the player's interest and growth in the performance, much like the care of the flower to which Zeami likens the art of acting [223].

Returning to the weapons and their relationship to the Pact of Punishment, there are three unlockable versions of each weapon. Two are only tied to the amount of titan blood required, which can be significant when upgrading the final levels of each. The final, secret version is only unlockable after certain named characters tell Zagreus a specific phrase and also costs a significant amount of titan blood to unlock and upgrade, requiring time investment and skill to accumulate the necessary resources. Importantly, the game also requires Zagreus to have working relationships with other cast members to fully open up the possibilities and variation present in each weapon, in a way reminiscent of the Viewpoints conception of collective growth during rehearsal and performance [29]. When discussing performance goals for an actor, Zeami is quite adamant that an actor should seek to master all roles available and can only do so through practice and willingness to continuously grow [223], in much the same way that *Hades* hides the weapon versions behind an increasing number of runs all at increasing

levels of difficulty.

3.5 Conclusion

Past discussions of theatricality in games have tended to emphasize the *performance* dimension of theater practice, with a particular focus on the liveness of performance. Though some aspects of *Hades* can certainly be understood through this lens, I find that what makes *Hades* stand out as a theatrical game instead lies in its presentation of a process of continual reflection through gameplay. The game scaffolds this reflection for the player so they may be drawn in to more deeply examining the core themes of the plot. All of these elements come together to create an experience akin to the theater *production process*. The first arc of *Hades* can be viewed as a facsimile of the rehearsal process, providing players with a window into the gradual development of a shared understanding between the cast and crew (i.e., the various characters) of what the play is about. Once the player successfully escapes from Hades for the first time, the focus shifts to an investigation of the central mystery in the game's backstory: what exactly happened between Hades and Persephone? Finally, once this uncertainty is resolved, the focus shifts again to the gradual revision of the play throughout its run for its new audience: the Olympians, who must be simultaneously appeased and gradually brought to a new understanding of past events through theater.

All throughout, the cast dynamics and other behind-the-scenes aspects of the play continually shift and evolve. The player is encouraged to take on a slightly different

gameplay goal for each run: sometimes to advance the main narrative, sometimes to advance a particular subplot, sometimes to complete a prophecy, sometimes to acquire more resources, sometimes to overcome a self-imposed Pact of Punishment challenge, and so on. A particularly ambitious, grueling and flawed run may be followed by a relatively safe "breather" run at the player's discretion, allowing players to deliberately negotiate how hard they want to push themselves on each performance. Yet despite the shifting context of each run, every run ultimately exhibits a relatively fixed overall *shape* as the player journeys through the various regions of the underworld in an unchanging sequence.

Notably, the narrative of *Hades* feels dynamic and responsive despite the lack of any branching structure. Dynamism is instead achieved through the recombination of fixed elements within a fixed high-level structure, exposing a slightly different selection of dramatic elements to the player on each run. As in theatrical rehearsal, the player comes to understand the story volume of the production as a whole via repeated sampling of individual performances and reflection on how the different elements of the performance all hang together—or, in some cases, fail to do so.

Through its identification of the play process with the process of theatrical production, *Hades* demonstrates an intriguing potential future direction for deliberate theatricality in games. In the context of a conventional theatrical production, the process of reflective reinterpretation through which a production's members continually revise their collective understanding of a play's meaning and purpose can span many weeks, months, or even years. Moreover, this process almost inevitably involves dozens

or hundreds of participants in a wide range of roles. As such, participation in this process is demanding and unapproachable. Obviously, a videogame cannot recreate this process in its entirety—but, through game and narrative design strategies like those used in *Hades*, games *can* provide players with a distilled form of some of the play-pleasures of reflective reinterpretation.

For Zagreus, as for the cast and crew of a long-running theatrical production, there is no escape from the process of continual remotivation: no final resting point that brings all the elements of the production into total, timeless harmony. Every run of *Hades* brings a slightly different context, a slightly different motivation, and slightly different obstacles that must be overcome. But for Zagreus, as in theater, there is an eventual pleasure and harmony to be found in the cyclic process of revision itself. The process, far from being an obstacle to the realization of a final, "perfect" performance, is the beating heart of theater as a medium: the reason both we-the-audience and we-the-performers find it so persistently compelling. There is no escape—yet in the end, one must imagine Zagreus happy.

My hope is that theatricality as I posit it here is freeing. Theater's understanding of storytelling is applicable to far more than just the stage. Narrative design is more than structure and plot. I say this as a playwright, in some ways the ultimate arbiter of the plot. But why do I do this work? Why do I have this view of theatricality? Because I love seeing what people do with my plots and characters.

Interactivity and liveness are larger concepts than they are often given credit for. Interpreting and reinterpreting a story is interactive, enacting a game's story and

processes is a form of liveness. The next two chapters tug at these concepts and, hopefully, convince you that systems are optional when creating interactive narratives and that part of the fun of theatricality is in collective experience.

Chapter 4

Theatricality and Story Volume Poetics

This chapter is the result of a collaborative effort between myself, Isaac Karth, and Max Kreminski to use *Umineko When they Cry* [1] to expand the understanding of interactive storytelling and story volumes [134]. Unsurprisingly my main contribution here is related to the concepts of theatricality and interpretation I discuss in the previous chapter. The reason we picked *Umineko* as the follow up to the *Hades* paper, other than it being our fixation at the time, was that it is almost diametrically opposed to *Hades* in structure and system design while still making reinterpretation the core aesthetic.

The contrast between *Umineko* and *Hades* strengthens the concept of theatricality I posit in my work by untying it from the presence of computational systems. Theatricality is an aspect of narrative design that focuses on the aesthetics of the play experience. By using *Umineko* as an example in this chapter, I want to highlight this fact in particular. Systems can support our goals of drawing players in and asking them to deeply engage with a narrative, but they aren't strictly necessary. This may sound

strange coming from a system designer whose dissertation is heavily focused on building new systems to support this aesthetic, but my work is really focused on using the right tool for the job.

I created the systems I describe in this dissertation because they were a necessary part of creating the full play experience I wanted. *Umineko* proves that theatricality is bigger than just building systems to facilitate particular experience. It shows us that we can ask our players to model entire games in their heads and reward them for rising to the challenge without the need to build bespoke systems. Now, I love my bespoke systems and the experiences they create, but they are one way of approaching theatricality. Again, choosing the right tool for the job is important and one of the main takeaways I want from my dissertation.

Computational systems aren't a hard requirement of this form of theatricality, as *Umineko* and many others [22, 25, 62, 146, 240] demonstrate. What you require, however, is a desire to have your players or audience actively reinterpreting your work as they experience it. You must be willing to support them in this endeavour, through the lessons found in both computation and theater, through narrative design and dramaturgy respectively, and show them the pleasures and joy of seeing the themes and narrative unfold in beautiful and complex ways. .

4.1 Introduction to Theatricality and Story Volumes

Many interactive digital narrative (IDN) systems are capable of producing numerous distinct *storylines*, which share some common properties but differ from one another in ways that appear contradictory when attempting to treat them as co-canonical. One theory of IDN poetics therefore positions an IDN system as defining a *story volume*: a generating function that produces storylines, and whose meaning lies not just in the storylines themselves but also in the relationships *between* these storylines.

The theory of story volumes was originally proposed by Jason Grinblat in the context of interactive emergent narrative. For Grinblat, a story volume is "a family of emergent stories, all of which are begotten by a set of carefully curated system parameters" [93]. However, despite its introduction in the context of emergent storytelling specifically, the theory of story volumes also has a broader applicability to IDN in general, providing a common terminology for discussing issues related to the experience of difference and repetition across multiple stories generated by a single IDN system. It would therefore be useful to disentangle the discussion of story volumes from the specifics of their use to understand interactive emergent narrative in particular.

To advance a more general understanding of the poetics of story volumes, I examine *Umineko no Naku Koro ni*—a heavily metafictional visual novel series with no emergent or systemic narrative components and almost no player choice, but in which the characters explicitly understand themselves as existing within a story volume—as a case study of how story volumes can be used to create narrative meaning. *Umineko*

is unusually explicit in its discussion of its own structure, and therefore presents an unusually strong exemplar of story volume poetics and theatricality in a wholly linear context.

4.2 Related Works

The definition of *story volume* used here is taken from Grinblat's "Emergent Narratives and Story Volumes" [93], which in turn builds on the sense of the term introduced in a Project Horseshoe group report [66]. The story volumes framework is elaborated further by Grinblat et al. [94], who contrast story volumes with the *protostory* concept from Koenitz's *System, Process, Product* model of IDN [141]: unlike a protostory, a story volume puts its "emphasis on the shape of the Product stories and de-emphasis on any narrative cohesion prescribed by the System" [94]. The rest of this chapter is devoted to a close read of *Umineko* using the lenses of story volumes and theatricality to highlight its approach to narrative design and lessons we as designers can take from the way it invites play.

Umineko involves the repeated diegetic "replaying" of very similar events. There have been many discussions of the role that replaying and rewinding have in games. For instance, Kleinman et al. present a framework for discussing rewind mechanics [139]. In that framework, from a player's perspective, *Umineko* has designer controlled rewind; has either scope 0 or global scope; uses UI and narrative elements in its rewind presentation; has a linear structure at its meta-level; and acknowledges the

existence of the rewinding to an unusually metafictional degree, even when compared to many other games that have extra-diegetic acknowledgement of rewinding.[1]

In "Reading Again for the First Time: A Model of Rereading in Interactive Stories," Mitchell and McGee "propose a model of rereading in interactive stories in which readers are initially rereading to reach some form of closure" [194]. Separately, Mitchell and Kway discuss replay and rereading in storygames with a rewind mechanic, particularly with the distinction of playing for completeness and playing for closure in the context of *Elsinore* [189]. While *Umineko* continually rewinds to the beginning, the more linear metafictional frame story emphasises reading "in a new way" [189] and presents another form of repeat experience to include in future analyses.

4.3 What is *Umineko no Naku Koro ni*?

Umineko no Naku Koro ni (lit. *When the Seagulls Cry*) is an independently-developed episodic visual novel by the doujin circle 07th Expansion [1, 2]. It consists of eight episodes, beginning with four Question Arcs (each of which depicts an alternative version of the same events) and concluding with four Answer Arcs (which present additional backstory and metafictional discussion between the characters regarding what happened in the question arcs).

Umineko's first episode begins as a realistically-grounded, orthodox, whodunit murder mystery on an isolated island, portraying itself as a Golden Age detective puzzle

[1] From the *characters'* metafictional perspective, the "game board" (as discussed in Sec. 4.4.1) has designer dominant control, dynamic scope, and an ontologically complicated narrative justification.

novel: the kind of mystery that the reader has a fair chance to solve through examining the clues. In-text it explicitly compares itself to the structure of Agatha Christie's *And Then There Were None* [51].

The initial plot is straightforward. Following years of estrangement from his family, Ushiromiya Battler pays a visit to his outrageously wealthy and occult-obsessed grandfather's private, isolated island of Rokkenjima[2] during a family conference in October 1986. The family's arguments about their future inheritance are abruptly interrupted by ritual murder. While the more superstitious people present attribute the murders to the unseen magical witch Beatrice, most of the family is initially skeptical, unwilling to pass the blame to a presumably nonexistent witch who never appears on-screen. They gradually become less skeptical as the ritual continues, with characters being killed in seemingly impossible locked-room murders. The first episode ends with all of the characters presumed dead and the mystery of the impossible murders unsolved.

So far this mirrors the structure of Agatha Christie's novel, with an extra dash of slasher horror. The story swerves at this point: in an extradiegetic epilogue the characters conclude that the unsolved murders mean that they were in a fantasy novel, with magical murders as the only logical explanation. Battler disagrees, vehemently. At this point the previously unseen Beatrice walks through the fourth wall and appears on-stage, setting up the metafictional debate that drives the remaining episodes: Were the murders somehow committed by a human culprit or was it a witch's magic? Is Beatrice real? Is this mystery or fantasy?

[2]Which can be read as "Six House Island" in a nod to Ayatsuji Yukito's title conventions (e.g. *The Decagon House Murders* [11]).

As befitting a game about the ontological status of a particular murder mystery, the mystery genre itself is a major theme and discussed in detail, particularly the sub-genres that focus on solvable puzzles. Authors such as Agatha Christie, Shimada Souji, and (anachronistically) Ayatsuji Yukito are mentioned by name, as well as the personification of Knox's "Ten Commandments" for detective fiction [101], tying it to both the Western Golden Age [102] detective fiction tradition as well as the parallel Japanese *honkaku* and *shin honkaku* sub-genres [61, 241].

By borrowing elements of detective fiction, *Umineko* sets up a parallel between the game of solving a mystery and the meta-game of understanding the story volume and the game's highly theatrical nature. The characters and the player are both trying to understand the same thing: what are the rules that govern this space we find ourselves in? A very similar question to what theatrical productions must answer and push against when working with a script [29, 48].

4.4 How does *Umineko* Illuminate Story Volume Poetics?

Umineko's first four episodes—the Question Arcs—depict four alternative storylines drawn from a single story volume: different versions of the Rokkenjima Incident, in which the vast majority of those present on the island are killed in some way. The final four episodes—the Answer Arcs—shift to presenting backstory and conflict between the characters as to which interpretation of the story volume as a whole is most true or valid. The mystery that the characters (and the audience) seek to solve thus gradu-

ally shifts from centering on the question of what happened in *one particular storyline* to the question of what common circumstances are consistent with *all* of the depicted storylines: in other words, what parameters define the story volume. These shifts in attention and priority of the characters in the story mirror the shifts in context that take place in *Hades*, complete with changing the distance between the player and characters' motivations for continuing with the game.

In a metafictional move, in *Umineko* discussing the shape of the story volume is a central part of the plot. This move first becomes apparent in the epilogue of episode 1, which initially appears to be a non-diegetic, non-canonical, out-of-character *omake* in the form of a "wrap party," with the characters speculating about how they could do better next time (Fig. 4.1). The shift here is arguably even harder than any of the changes in context in *Hades* as the latter never fully enters a metafictional space with its framing. But when Battler objects to blithely accepting the magical nature of the murders, the witch Beatrice breaks the fourth wall's fourth wall, walking onto the stage and turning the backstage conversation into part of a larger diegesis.

This meta-diegesis sets the stage for the rest of the games: in a tea room in Purgatory, Battler and Beatrice argue about whether the story is fantasy (and the culprit is a witch) or mystery (and a human somehow committed the impossible murders).

Structurally (in the Carstensdottir et al. sense of the graph structure [42]), *Umineko* consists of a linear sequence of eight episodes. With one major exception, the episodes are linear visual novels, though the player has access to a hypertextual "Tips" menu that enables them to revisit key information about the characters and events of

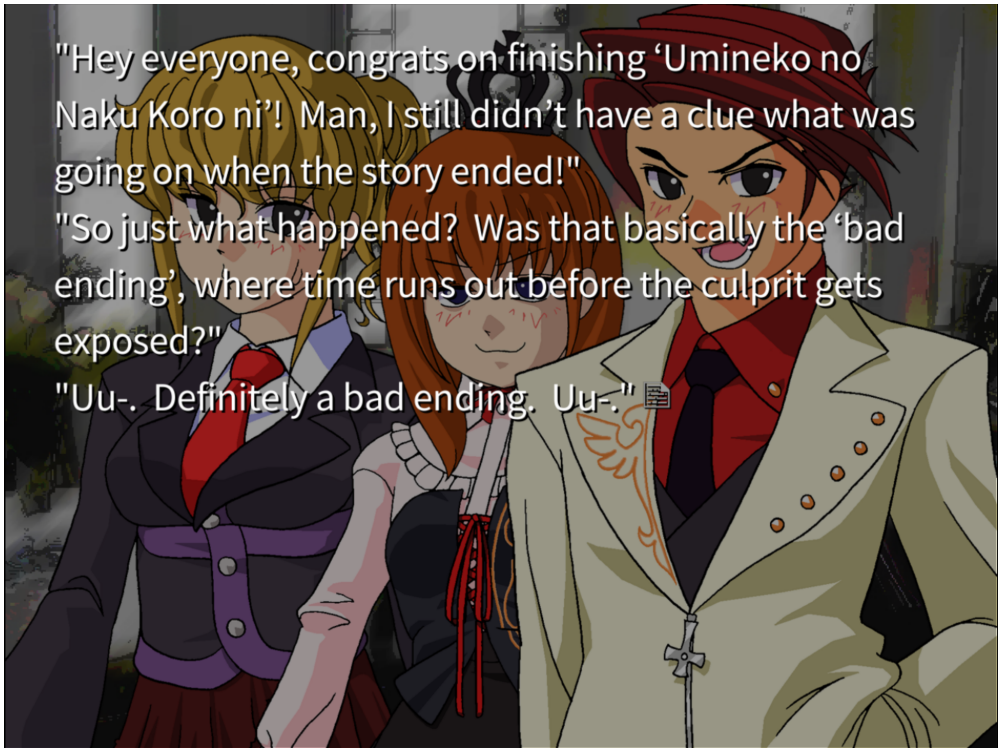

Figure 4.1: Characters in the first episode epilogue, discussing the the ending of the episode in branching narrative terms.

Figure 4.2: The game is mostly linear but includes hypertextual information about the characters and clues.

this episode so far (Fig. 4.2). The final episode contains several opportunities for player choice leading up to a conclusive interactive branching point, which puts the question to the player: is it fantasy magic, or a mystery trick?

This mostly-linear structure that nevertheless is intimately concerned with examining story volumes gives us a unique opportunity to consider story volumes in the absence of emergent and choice-based narrative considerations.

4.4.1 A Diegetic Story Volume

The degree to which *Umineko* discusses its own structure is somewhat unusual, even within the context of metafictional story volumes.[3] The shape of the story volume is an explicit topic of discussion as the characters try to figure out the rules of the mystery.

Battler and Beatrice refer to the storylines that play out in front of them as games on a game board. The characters on the island are referred to as game pieces, first as a metaphor and then, in a pataphoric move, as the literal pieces in the game Battler and Beatrice are playing (as they grow into the diegetic roles of detective-story reader and author respectively).

The game board that the characters are using generates, in our terms, a story volume. The game board is a set of rules that express what stories are possible within that story volume. The first four episodes of *Umineko*, therefore, depict a series of

[3]The character roster eventually includes personifications of the author, the reader, fanfic writers, internet commentators, love, marriage, the roles of certainty and miracles in narrative, detective fiction genre conventions, guns, and duct tape.

storylines drawn from the same story volume, presented as both a repeated in-world tragedy on the game board and a discussion topic in the metafictional debate.

The reason *Umineko* gives for foregrounding the story volume is to get the reader to start thinking about the differences and similarities between storylines. It deploys the metafictional discussion as part of a poetic strategy to encourage the player to look beyond the immediate situation of a single storyline. The games encourage the player to triangulate the mystery from multiple angles, with an explicit goal of inducing the audience to reason out the solutions and their implications while avoiding explicitly stating them within the text.[4]

The first step of story volume poetics is to convey to the player that there *is* a story volume. Making the metafiction diegetic is a particularly blunt way to do it. *Hades*' approach to theatricality is a more, if not overly gentle, approach to exposing its story volume. Many other games have experimented with ways to signal the existence of a story volume: "Clementine Will Remember This" in *The Walking Dead* [262] implies a story volume extradiagetically, the diegetic time travel [97] in *Majora's Mask* [201] and *Ocarina of Time* [200] uses time loops and contrasts between timelines, *Nine Hours, Nine Persons, Nine Doors* [52] includes a flowchart mapping out the story volume, and repeated resurrections in *Planescape: Torment* [27] make the common videogame dying/respawning cycle diegetic.

[4]In several ways this is reminiscent of the novels described in Borges' "Examen de la obra de Herbert Quain" [32]—Like *The God of the Labyrinth* the reader is left to discover for themselves that a stated solution is wrong, like *April March* the text branches backwards in time, like *The Secret Mirror*, the characters find themselves in stories written by others, and like *Statements* several of the stories are deliberately calculated to disappoint the reader.

4.4.2 The Cat Box and the Game Board

The game board encompasses the events that happened during two days on Rokkenjima. As the episodes continue, it is revealed that people outside the island do not know the solution to the mystery either. In-game this unsolved mystery is described as a *cat box*, in reference to Schrödinger's cat: just as the cat is at once both alive and dead until the box is opened, without the knowledge of the true ending, any of the storylines might be true.

This is an explicit statement of the poetics of a branching or emergent narrative game, making some of the possible procedural rhetorics [30] more visible. For example, *Umineko* frequently makes use of the multiple storylines to give us multiple perspectives on the characters, in a way that is difficult to do without a story volume. Changing the mutually-exclusive situations that the characters are subjected to expands our insight into their relationships: we can see both how George reacts to the death of his mother Eva, and how Eva reacts to the death of her son.

Rather than being limited to a single version of the character, we can examine all possible versions. This is a useful device for a writer: since we aren't locked in to a single version of events, we can try out different combinations. As the characters become more fleshed out, they transition into actors playing new roles.

Part of the pleasure of playing the game comes from being able to build up a mental model of the characters through seeing them react to multiple variants of the same situation. This anticipation of behavior and the contrast between characters is

accelerated in the context of a story volume.

By means of two mutually-exclusive storylines, the first two episodes establish a parallel between Natsuhi and Rosa: how does each character behave when thrust into the role of being the sole responsible adult, trying to protect her daughter as their family is murdered around them?

We see how they both define themselves through their role as a mother, their sense of failure in living up to the expectations the patriarchal environment has placed on them, and the tragedy this engenders. Comparing their roles across storylines, we can see themes that might otherwise be obscured, or even impossible to see from within a single storyline.

On the other hand, the opportunity to show the characters from multiple perspectives shows how the story volume can be used to enrich the characters. Rosa's relationship with her daughter Maria is explored across multiple storylines. A pivotal event for them that occurs early in most storylines is when Maria is caught in the hurricane's torrential rain: we see different ways that it could happen and a range of possible outcomes. Was Rosa neglectful, forgetful, abusive, or uninformed? The different configurations of their relationship are reiterated across multiple storylines, allowing the intertwined horror and love to assume new configurations in each. The player's role as they are exposed to different readings and portrayals of recurring events can be likened to that of the dramaturg in theatrical production, who is tasked with compiling the production history of a play to better understand the decisions made and themes emphasized in previous iterations and interpretations of the story [48].

The general trend in interactive fiction that is centered around discovery is toward what Sedgwick termed *paranoid reading*, focusing on teasing out the "true" meaning from a text [233]. *Umineko*, however, refuses to build up to a singular "true" ending, instead repeatedly emphasizing that every storyline being presented is potentially true, though not equally plausible. Thus, the player is invited to actively engage in *reparative play*, taking parts that "arrive in disrepair" and assembling them into a coherent whole [94]. With the cat box left deliberately unopened, it is ultimately up to the reader to assemble their own preferred interpretation of the events on Rokkenjima, drawing on fragments of potential truth that were presented across many mutually contradictory storylines.

4.4.3 Probability and the Shape of the Story Volume

One way that *Umineko* encourages reparative play is through its depiction of probability and plausibility. Though the events of each storyline can be seen as true, some events take place in a majority or plurality of possible storylines, while some are rare, taking place only in storylines where multiple unusual circumstances intersect.

As an unrolled[5] story volume, *Umineko* can induce a felt sense of probability in the player by literally controlling how frequently a certain event takes place across the storylines that are presented. Because all of the presented storylines conform to the rules of the story volume, as players we begin to develop a sense of what is and is not probable as we untangle the overdetermined tragedy. This can be used to establish some

[5] *Unrolled* in the sense of an unrolled loop in computer science: code that would have been repeated in a loop is instead written sequentially (e.g. [142]).

recurring scenes as highly likely, if not inevitable (e.g., Maria being caught out in the rain, which occurs in all of the Question Arcs) and even to create a sense of dramatic irony: while the characters speculate that Jessica would have been less headstrong and independent if she had been raised in other circumstances, we players know from observing her across multiple storylines that this aspect of her personality is a constant.

Simultaneously, through its metafictional elements, *Umineko* is able to *invoke* probability by having characters with greater metafictional awareness discuss it directly: for instance, when Bernkastel (the story volume-traversing Witch of Miracles) states that a particular depicted event is so rare that "there's actually a $2,578,916/2,578,917$ chance" against it happening. Because the player has (by this late stage in the game) directly witnessed the fact that some events are more plausible than others across the story volume, an explicit statement about probability from a metafictionally privileged character can thus be used to create the felt sense of a miracle—just as explicit procedural die rolls or even the stochastic nature of simulation can be used in games like *Disco Elysium* [280] and *Hades* to make the player feel that they have succeeded or failed against all odds.

From a storyteller's perspective, manipulating the weight of probability as felt by the reader is difficult in the absence of a story volume but highly effective in its presence. In *Umineko*, this becomes even more apparent when we move from considering the vanishing probability of a miracle to the certainty of tragedy.

4.4.4 Logical Quantifiers

In order to explicitly lay out the rules for fair play in the mystery, *Umineko* introduces a mechanic for stating absolute truth: red text. When a witch makes a statement with red text, it is axiomatically true. This allows the mystery narrative to continue without excessive haggling about descriptive details in the clues, but it also sets up a set of logical constraints on the shape of the story volume: something stated in red text is true of all storylines. If it is stated in red text that the murders were never done with small bombs, then that is true across the entire story volume, closing off previously-perceived possibilities. This applies to *previously encountered* storylines as much as future ones: while we readers might encounter storylines in a particular order, they are notionally parallel, not sequential.

Red text has utility to both the characters, who are debating the rules of the story volume, and the players, who are trying to learn the rules of the story volume. Its appearance primes players to actively participate in the mystery-solving (by giving them footholds on which to base their theories) while also actively directing them to think of the mystery in terms of the story volume and its possible shapes. In this way, the red text functions much like a script in theater, establishing what must happen to be considered a version of a particular play.

As a device, absolute statements can also be deployed for emotional effect: the pacing of a reveal can be timed to crush a character's dreams with the realization that such a thing is categorically impossible. Further, discovering absolutes in the story

volume can imbue previously insignificant-seeming details with new narrative weight, provoking players to reinterpret the implications of previously depicted events.

Notably, none of this requires resolving the ambiguities in the text. *Umineko* wants its players to grasp the ambiguity, but not necessarily to resolve it. The text explicitly encourages engagement with what *could have* happened, perhaps even more than it encourages players to solve the mystery of what *did* happen—and most often, red text is deployed to rule out a single hypothesized explanation for past events without suggesting any obvious alternative. Thus, absolute statements are used to keep the player in a state of uncertainty, even as the player attempts to uncover the story volume's ultimate truth.

4.4.5 Storygameness and Story Volumes

Mitchell's work on *storygameness* discusses how the player's understanding of a game as a storygame affects their focus: "as players experience a storygame, they shift focus between the narrative and the playable system" [191]. *Umineko* explicitly engages with the question of whether the story or the system should be the focus of player understanding at several points.

In *Umineko*'s fifth episode, the role of the detective is taken over by Furudo Erika, and her approach is explicitly contrasted with Battler's quest. By this point, Battler is more interested in the message that he believes the rules are intended to communicate. Erika, in contrast, wants to learn the rules in order to reduce the mystery to a logic puzzle and never have to think about it again. In a deliberate construction,

neither is right: Battler is unable to comprehend the message without understanding the rules, and Erika is unable to solve the mystery without understanding the message.

This theme continues, and it is extended to the audience: when answers to the mysteries are presented in later episodes, the reveals are structured so that the logic is incomplete without understanding the characters and their motivation in the narrative. Ultimately, *Umineko* is structured so that the narrative is incomprehensible without understanding the rules of the game board, but the rules are incomplete without also understanding the narrative.

The interdependence of system and story in the mystery as a whole is emphasized in other episodes: for example, when in-universe authors appear on stage, writing what is—from their point of view—real-person fan fiction [74, 263] about the murders, strictly following the rules of the game board. Or the extended discussion of the role of the author and the reader in the meta-frame story for the meta-world of the game-board in the fifth episode.

In each case, the rules and the narrative coexist: while an author could, in theory, write anything, there is a limit to a reader's suspension of disbelief, particularly when the story needs to conform to the expectations of the mystery genre. Getting the reader to accept a miracle is difficult. This is part of why *Umineko* focuses on probability: as with the die rolls in *Disco Elysium*, the reader needs some mechanism to accept improbable results, even—or especially—good results.

As it turns out, much like how a theatrical production putting on a play must perform *that script*, the author *can't* write just anything: they can only write what fits

within the story volume.

4.4.6 PCG Poetics

By treating a story volume as a generating function or ruleset for many different storylines, we also gain the ability to analyze story volume poetics using tools that were originally developed to understand the subjective experience of procedural content generation (PCG).

For example, Karth's category of *repetition* in PCG poetics is a useful lens to use for analyzing story volumes, since the parallel story threads have an aesthetic effect via *repetition* [133]: the similarities help the differences stand out, while also making it more obvious what each storyline has in common. Viewed through this lens, the aesthetic effect of the storylines is linked to the ritual repetition of events. *Umineko* makes this explicit with a literal occult murder ritual, but this repetition aesthetic can be seen more broadly as other elements repeat. When viewed together, parallel instances of the ritual have the feeling of call-and-response, as we anticipate the next seemingly inevitable murder.

In a generative system, an artifact that breaks the pattern draws attention. In a story volume narrative, much like a play we see a new production of, we read to discover what is different this time: a difference signals that there is more story to be found. We use these landmarks to orient ourselves.

The relationship between the individual instance and the distribution of possibility can communicate something that neither could on their own. We need the story

volume to be able to notice differences, and the individual storyline to appreciate why those differences matter. We can only understand the heart of the story when we have a sense of both.

We can think of both generative systems and story volumes as possibility spaces, with each individual storyline a point in this space. This implies that tools for analyzing generative spaces—such as Expressive Range Analysis [144, 244]—can be applied to interactive narratives, and tools for working with interactive narratives can be applied to generative systems.

This is similar to how tabletop roleplaying games and procedural content generators are systems of rules that describe a volume of possible content. As Guzdial et al. argue, viewing tabletop roleplaying games under a PCG lens gives us a way to describe how roleplaying game systems can produce widely varied content that is nevertheless bounded by its possibility space [96]. In contrast, viewed through the lens of a game between the player and the author, the process of "solving" a mystery story revolves around narrowing a space of possible storylines down to a plausible and satisfying explanation.

Umineko is a mystery story, but it is also a commentary on the puzzle-mystery genre. The concept of the "fruitful void" [15] from tabletop roleplaying theory—the unsystematized central theme at the heart of every tabletop roleplaying game—is particularly interesting in this light. In *Umineko*, the solution to the mystery is never explicitly stated.[6] Instead, the reader-player is encouraged to discover the thematic heart of the

[6]Indeed, the writer has stated in interviews that one desire was that the solution would not be presented in such a way as to able to be captured in a screenshot [136].

story by circling but never fully resolving the ambiguity. The multiple endings leave it up to the reader to determine: was it mystery, or was it magic?

4.4.7 Theatricality

This cycle of replaying the same storyline with minor variations conforms to the definition of *theatricality* I posit as: "a property of creative works that repeatedly reinterpret and recontextualize a partially fixed performance over a period of time, in such a way that this continual recontextualization is at least partially exposed to the audience" [129].

Similar to how *Hades* cycles players between the *stage* (a single attempt to escape the underworld) and the *diegetic backstage* (the House of Hades) [129], *Umineko* implements internal theatricality: it has both a stage (the "game board", on which a particular version of the Rokkenjima Incident plays out) and a diegetic backstage (the purgatorial tea room). In both cases, the characters return to the backstage to debrief, where we see them as actors (or playing pieces) rather than characters. In the backstage our collective understanding of the story-volume / play-script increases: diegetically, the actors come to understand their roles better as we players grow our understanding in parallel. Additionally, in both *Umineko* and *Hades*, this performance/reflection loop calls attention to subtle differences between performances, leading the player to pay attention to details that they would have overlooked otherwise.

For me, theatricality also enables the player to experience the production process, from early rehearsals to opening and finally closing nights. In episode 8, *Umineko*

implements this diegetically, presenting a final command performance, with the characters commenting on the skill in staging the mystery gameboard, while a character tries to come to terms with the meaning of the story volume as a whole.

Theatricality, in this sense, is directed to getting you to try new things. The central mechanic of the mystery genre is the player coming up with a theory to explain the mystery. Because *Umineko* incorporates this cycle of theatricality, it can diegetically demolish the theories that you formed on previous runs. But the cycle continues, inviting you to form new and better theories. The player's improving skill is invested in acquiring a deeper understanding of the rules of the mystery and the themes of the story volume.

Theater and theatricality can be viewed as a search for novelty and "making the old look new" [223], as described by Zeami Motokiyo in his treatises on Nō theater. Our understanding of the characters and their situations is fueled by this cycle of new looks at old things. Characters who are unsympathetic from one perspective are more understandable from another angle.

For one example: We get some familiarity with Maria's interests in the occult in the first episode, a deeper look at her troubled relationship with her mother in the second episode, and much deeper insight how both of them connect later, all of which is necessary for understanding how the conclusion of the fourth episode could be possible. Each cycle gives us a new perspective on old events. Understanding can build up through successive episodic cycles, triangulating insights about the story volume that are never explicitly stated in the text.

4.5 Summing It All Up: The Heart of the Story Volume

Viewing a storyline in the context of its story volume tells us more than either would on their own. The possibility space of a story volume defines an *expressive range* [244] of storylines that meet the story volume's constraints. The meaning of an individual storyline is partially dependent on its positioning within the overall story volume, much as Kreminski et al. [144] argue that individual artifacts from a generator's expressive range derive their meaning in part from where they fall within this expressive range (cf. Sec. 4.4.6). After observing multiple storylines, the player develops a sense for what is probable, possible, or impossible in the story volume as a whole.

Episode 7 takes advantage of *Umineko*'s metafictional nature to explicitly address this by combining multiple storylines, looking across the entire story volume. This leads to a central theme of *Umineko* as a whole: knowing the facts isn't as important as understanding the heart of the story volume.

The theme of knowing the heart is most explicitly put forward by a character in the seventh episode, Will,[7] an about-to-retire detective who is driven by his care for the people affected by mysteries: "If you want to play the detective, don't neglect the heart," in his terminology, referencing the importance of understanding the motivation of the culprit when solving the mystery.

While a distant read of the shape of the story volume can give us valuable information, we also need the individual storylines to understand the heart of the story.

[7]Willard H. Wright (ウィラード・H・ライト), who is also an oblique reference to the American mystery writer S. S. Van Dine.

The shape can't tell us what motivates the characters to make the decisions that create that shape. The statistics of the story volume don't tell us about the emotional impact. We need to experience the storyline for ourselves to feel the emotional impact of Maria being abandoned in the rain.

The importance of understanding the story's heart is perhaps made clearest by the introduction of a new game mechanic in episode 5. Like the red text that is used to make absolute assertions from episode 2 onward, *Umineko*'s Answer Arcs also feature occasional instances of gold text—the exact meaning of which is never made explicit. However, it can be inferred that gold text statements represent conclusions that can only be drawn by someone who understands the story completely, particularly the character motivations that shape the story volume. We can therefore think of *Umineko* as a ritual guide or theater script—leading the player to a place from which they, too, can make conclusive statements about the shape of the story volume, speaking from the heart.

4.6 Conclusion

By examining *Umineko no Naku Koro ni* from the perspective of story volume poetics, this chapter sets out an initial theory of how story volumes can create narrative meaning, independent of player choice and emergence. In particular, distinctive forms of narrative meaning can arise in works of story volume narrative from the presentation of mutually contradictory storylines as equally true.

This type of meaning can emerge from the player's felt sense of *probability* (i.e.,

what happens more or less frequently in storylines drawn from the same story volume); and from the existence of *logical quantifiers* (i.e., absolute truths about what is always required to happen, or never capable of happening, in any storyline drawn from the story volume). Additionally there are strong connections between story volume poetics and the conceptual frameworks that have previously been used to make sense of *storygameness* (i.e., the extent to which players understand a game as a storygame); *PCG poetics* (i.e., the aesthetic effects of procedural content generation in games); and *internal theatricality* (i.e., the aesthetic effects of a system-facilitated performance/reflection loop in IDN).

Broadly speaking, this analysis of story volume poetics paves the way for a clearer understanding of any IDN work that encourages replay-with-variation; positions its characters as having awareness of the multiple storylines that might emerge from the system; or makes use of strategic re-presentation of events to achieve aesthetic effects on the player. From a technical perspective, this analysis also begins to suggest a novel, story volume-aware approach to story *generation*: by generating stories that are intended to be experienced as part of a story volume, we can take advantage of story volume poetics to achieve aesthetic effects that conventional forms of narrative might not be able to leverage as easily.

With the end of this close read, I hope to both expand the applicability of theatricality beyond what might at first seem like a narrow scope when exclusively looking at *Hades* and to better position the concept itself in conversation with other work in interactive storytelling. I see theatricality and story volume poetics as mutually

reinforcing lenses which are useful when applied to works of both the computational and theatrical variety. A play script functions as the constraints of a story volume when creating a new storyline through the theatrical production process. Looping narrative designs can effectively draw out the theatricality of an interactive narrative by forcing players to revisit themes and moments with fresh understanding.

My hope here is that I am giving tools to both computation and theater to deepen their respective understandings of what it means to tell a story. This entire dissertation is devoted to exploring the parallels between these two forms and the close reads in this chapter are my expression of that in pure narrative design.

Something else I think is worth highlighting here before further expanding the lens of theatricality is this: that working with unfinished stories can be powerful. As both a playwright and narrative designer, anything I create is necessarily unfinished. It *must* be played with to be complete. Even something as linear as a traditional play script or *Umineko* must be played with to become complete. This aspect of theatricality isn't something I so explicitly call out in the papers that make up this chapter, Chapter 3, or Chapter 5, but it's there. The understanding that by inviting reinterpretation of the stories we present, we are both required to keep our stories unfinished and to trust our players to in fact play with the story to find its meaning.

Not one single meaning mind you, for that is the nature of unfinished work and a large component of the last piece of theatricality I discuss. So far I have discussed singleplayer experiences with respect to theatricality but theater itself is highly communal and collaborative. It's best not to forget that.

Chapter 5

Theatricality and Communal Ritual Play

This chapter on theatricality is a condensation of a collaboration with Bjarke Larsen [151] focused on describing narrative design patterns found in multiplayer games with communities devoted to interpreting their stories. I feel it's important, as a theater scholar and practitioner in addition to being a computational researcher, to draw parallels between theater's communal nature and the ways existing games create communal interpretations of an experience.

Now much of what I have said in chapters 3 and 4 still holds true, as one would hope in a story like this. The main through line of these three chapters and their understanding of theatricality is the necessity for a story to be unfinished. If I give you a story in a box that you only have to put the barest effort in to experience, why should you want to dig deep into its themes or meaning? Stories and games are colored by the

past experiences of players [91, 191] necessarily. When a work is unfinished and must be played, it not only creates space for a dialogue with the person engaging with the work, but makes revisiting the work fruitful. Contexts change, more experiences are had, and they will continue to color a player's relationship to a story. We can choose to embrace it and trust our players to find and re-find meaning in our work. We can design work in such a way that players find community through shared interest in searching for meaning.

5.1 Introduction to Theatricality with Multiple Players

Storytelling is frequently and historically a highly communal activity. Sitting around a fire, gathering in an amphitheater, gossip networks, copypastas, group reads of *Subcutanean* [217], we as humans love sharing stories together. Even the way my co-authors and I played both *Hades* and *Umineko* falls under this umbrella. Each of us had our own playthroughs progressing through *Hades* and we would compare notes about story beats as we encountered them while with *Umineko*, we were all playing the game and spinning theories together, something the game hugely encourages. Yet in neither of those papers or sections did this factor into our readings of these games.

When we as narrative designers do acknowledge the community's place in storytelling, we mostly focus on the collective *creation* of stories [118,160,161,276]. The communal reading or playing of extant stories, in games or otherwise, is under discussed in computation compared to the individual experience, with the play experience of

Terminal Time [168] being one of the notable exceptions. In this chapter I highlight how theater and theatricality [1] can help us as designers encourage communal engagement with our designs.

Communal play and rituals can teach us how to design narratives around the repetition that is exceedingly common in games as well as find ways to encourage players of multiplayer focused games to engage with their narratives. Current games with less traditional single player narrative structures and plots (such as *Destiny 2* or *Elden Ring*) utilize design patterns that encourage communal storytelling instead. These patterns can be extrapolated to create more experiences where the storytelling approach strengthens a community and narrative together, creating more powerful stories that connect people to each other as well as to the work of the narrative.

The single player narrative experience is well researched in interactive storytelling and games [3, 12, 13, 71, 131, 140, 153, 167, 197, 228, 273]. One ongoing strand from this direction of research focuses on better understanding the ways players' relationships to games change over repeated playthroughs [129, 138, 192, 193], showing the power of narrative understanding through repetition. The work in this area has largely focused on singleplayer experiences, in both the choices of games to analyze as well as the emphasis placed on each player's distinct, personal reinterpretation [129, 138, 192, 193]. Interpretation, however, is not exclusively an individual act and entire communities can engage in the practice while playing games, collectively creating new meaning out of

[1] The original paper is an intertwined discussion of ritual play using mythology and theatricality as a lens but I am only discussing the theater aspect here because it is the most relevant to my own body of work

(re)experiencing the same story. Not all games are equally suited for this kind of communal retelling and interpretation and certain design patterns lend themselves better to it than others.

By creating the lens of *communal ritual play* and focusing on its connections to theatricality, I hope to better understand the narrative design patterns that encourage collective interpretation of games. Such a lens enables the expansion of the work of studying the ways narrative designs create modes of storytelling that encourage players to engage in interpretation into the way these designs support *collective* sharing and interpretation of in-game experiences. Furthermore, this lens for a deeper understanding of the way multiplayer narrative design draws players into engaging with these games' stories which are often multimodal and feature hidden or otherwise non-obvious elements, frequently requiring a collective effort to discover in the first place. When narrative elements are used in these multiplayer games they are often told through fractions of a story when experienced a single time, requiring repeated playthroughs or engagement with a community, or sometimes both, to begin understanding the full story being told.

In a way, this chapter's focus on *ritual* as the site where myth and theater intertwine is a callback to the origins of theater as religious ceremonies [267]. Rituals, like theater and many multiplayer games, are built for the express purpose of repeated enaction of events. The narrative of these games rarely exists as a single linear story. As a result, the design patterns found in such games help to foster a community's interpretation of a narrative.

The lens of communal ritual play is built from theatrical performance practices and the literature around ritual and myth. From these, we can form an understanding of what is compelling about revisiting stories which can then be applied to existing multiplayer games, which have communities built around their interpretation. Theatrical performance practices work within the constraints of the play script, which provides a framework for telling a specific story, while performers form their own interpretations of the roles to find novelty and bring the narrative to life [29, 158, 223]. Likewise, storytelling traditions from mythology are strongly tied to ritual and retellings [71, 148], as communities would retell and re-enact myths continously, to both expand their understanding and convey their meaning to the society [86, 145, 220, 234]. By drawing from ritual, myth, and performance practices, we can expand our understanding of how people collectively interpret stories from a small group of specialists to much wider communities of varying expertise levels engaging in this form of novelty seeking and interpretation.

The rest of this chapter summarizes the narrative design patterns found using the lens of communal ritual play. Theses design patterns are: *Incomplete Information*, *Not Helping the Player*, and *Building for Player Expression*. In keeping with the overall theme of this chapter, these patterns all share the common element that they require some amount of the game to be incomplete, and, even beyond that, require more to be filled in than a single player can provide.

Synthesizing these patterns involved first identifying games with active communities discussing their play and storytelling experiences. Next, searching through

public discussions to find repeated topics about game elements with *high player friction*, defined as requiring discussion with at least one other player to be understood. Finally, constructing categories by grouping these discussions based on commonalities in subjectmatter and play experience. *Destiny 2* [35], *Final Fantasy XIV* [246], and *Elden Ring* [82] provide the subjects of the case studies in this chapter. By defining and illustrating these patterns via examples, I aim to not only once again expand the applicability of theatricality to narrative design, but also remind computation that there are multiple ways of putting people in community with each other.

5.2 Related Work

Reinterpretation through repetition is a phenomenon that has been studied in relation to digital games [129, 138, 192, 193], discussed as part of performance practices [29, 158, 223], and ritual and mythological studies [86, 145, 150, 220, 234]. Common to these is communal rituals as a storytelling practice. Communal storytelling is also seen in games, to understand how the creation of stories can be a communal activity [118, 160, 161, 276], something also well known in table-top roleplaying design [60]. Yet, this focus on collaborative *creation* is not the only aspect of communal ritual play, as through a synthesis of ideas from theatrical performance and ritual and mythological studies, a different lens emerges, of considering a collective interpretation of authored stories.

5.2.1 Replay and Rewind

The work focusing on replaying stories in games has primarily focused on the way individual players interpret singleplayer games and how those games facilitate reinterpretation of their stories [129, 138, 192, 193]. Mitchell and McGee describe the phenomenon of players of interactive stories transitioning between "reading again to reach closure" and "reareading after closure" where they move from looking for catharsis in the story in the former to reflecting on their relationship with the story in the latter [193]. When investigating the experiences of players revisiting storygames, Mitchell et al. expand on the idea of reading for closure to encompass both the experience of closure with the narrative and with the game systems, to describe ways players lose interest in replaying a game once the balance between both types of closure is lost [192].

Unlike Mitchell et al.'s focus on player closure, Kleinman et al.'s taxonomy of rewind mechanics describes the way designers facilitate the replaying and recontextualization of the story for the purpose of evaluating games which recontextualize themselves through their mechanics [138]. In a similar vein, Junius et al. propose the concept of theatricality present in games with looping structures specifically designed to recontextualize a game's narrative and invite a player to reinterpret their relationship to story elements within the game [129]. Expanding upon this definition of theatricality, Karth et al. add the notion of story volumes to explain the ways game narratives can be restructured while keeping the same *type* of story and guide players into interpreting the rules governing the storylines [134]. Where communal ritual play diverges from these

existing approaches is in its emphasis on the act of reinterpretation as *collective*, even if it still shares the recontextualization focus of rewind mechanics and theatricality.

5.2.2 Performance

Communal ritual play draws from performance practices to build its lens as theater, from early in its history, has a focus of recontextualizing and retelling stories for different audiences [26]. As performance often operates within the constraints provided by a script, these practices focus on the need to find novel and unique ways of portraying and interpreting a character, lest the performer(s) or audience become bored [158, 223]. Nō theater uses the metaphor of a flower as the central aesthetic of performance, something that continuously creates novelty and requires active care to maintain [223]. The novelty seeking described in Nō is an interpretive act, and importantly, a task for each actor to consider on their own [223], similar to the approaches to analyzing replay in games discussed in section 5.2.1, while also making its practice distinct from more recently developed theatrical practices.

Performance's relationship to the texts it uses to tell stories has undergone numerous re-examinations as new approaches have developed, with some practices viewing the text as dictating every aspect of a production [158, 223] and others viewing it as simply another element to be incorporated [19, 29, 123, 155]. When the text of a script is de-emphasized, it invites experimentation in performance and interpretation, finding views of roles, themes, etc. that may have otherwise been missed [29]. This freedom of interpretation informs communal ritual play as, like the work in section 5.2.1 describes,

repeated replaying of sections of multiplayer games will change players' relationship to the game's narrative components and move them beyond simply trying to experience a singular version of the story.

The popular western acting practices of the Method of Physical Actions [158] and the Viewpoints [29] heavily emphasize that performance, novelty seeking, and interpretation must be understood as collective acts. The Method of Physical Actions maintains that every action performed on the stage must in some way motivate a reaction from another character, even if the actor is alone [158]. Additionally, while the rehearsal process fixes some elements of the performance, it is the entire cast's job to explore the spaces between those fixed elements [158], sharing similarities to the way story volumes have been described [134], with the addition that exploring the possibility space is a collective responsibility. The Viewpoints shifts the focus of performance even further towards the collective, not only de-emphasizing the text but also the role of the director, to find an interpretation of a text and performance completely unique to each production [29]. More than anything the Viewpoints is intended to foster collaboration between actors to explore how they can discover actions on stage by indulging in possibilities rather than being restricted under some central authority, including the text of a play [29]. This is still framed within the rather small bounds of a theatrical production and though theatrical performance's understanding of finding novel interpretations within constraints helps to explain the creativity to be found in repetition, by itself it is insufficient to construct the communal part of communal ritual play.

5.2.3 Myth and Ritual

Early sociologists and myth scholars such as William Smith, E.B. Tylor and Frazer [4, 79, 234] discussed the connection between ritual and traditional mythology. In what has been named the "myth-ritualist theory" the function of myth was seen to make sense of ritual, as a way for societies to explain their customs [234]. Another view is seen in Campbell: "ritual is simply myth enacted" [36, 227]. As Segal [234] notes, the various interpretations differ on whether they consider myth or ritual primary and the other secondary, yet, ritual has long been closely tied to religion and mythological storytelling. See Nagy for a modern interpretation of the myth-ritualist position [199].

Ritual, furthermore has been directly connected to play. Huizinga directly discusses play as a kind of ritual [111], also seen in Wagner [272]. Rettberg [220] discusses how there is a ritual quality to the repetition of play, comparing it to rituals of life such as weddings and funerals. This echoes Krzywinska [145], Geraci [86] and Larsen and Carstensdottir [149, 150], who discuss the ritual and mythological quality of MMOs by treating the communal experience of playing a game as a kind of myth. Asimos [10] and Rusch [227] have a more expanded view on how contemporary games can be seen as mythology. Harrington [98] and Cragoe [60] furthermore discuss the connections between mythology and games, and how we play and talk about games can be viewed as a socially constructed and part of larger contexts. Drawing on myth, as a communal storytelling practice that relies on ritual, would be a valuable resource to understand ritual play in games and how it connects to storytelling.

As discussed by structural sociologists, myths can *"teach us a great deal about the societies from which they originate"* [157]. Scholars like Dorson [64], Levi-Strauss [156, 157], Bronislaw [162], Campbell [37] or Jung and Eliade [86, 234] have all contributed to the modern understanding of myth as useful to building and interpreting communities. And while this has later been challenged and reinterpreted in postmodern scholarship [17, 106, 137, 234] (also seen in game studies [77, 78, 98]), the connection between mythology and ritual and play are still valuable insights for games because of their repetitive, ritual nature.

Through this lens, we will see how mythological, communal storytelling experiences arise through ritual repetition of play. Thus, by looking at how rituals are formed in games, and how the communities understand, interpret, and maintain these rituals (for themselves and others), we can understand something about how the community interprets the game they are playing. Our view on ritual is informed by Junius et al. [129] and Rettberg [220] as a recurring, repetitive activity that is given contextual, social, or cultural significance, turning mundane activities into more meaningful ways to relate to the world. Rituals in games are informed by the design of the game, its affordances and design constraints, but also in subversion to them, as communities discover optimal strategies and new ways to play unintended by the developers, and thus seeing how the player rituals are formed in relationship with the design of the game is crucial to understanding how design impacts the ritual play of community.

5.3 Narrative Design Patterns for Communal Ritual Play

The following three narrative design patterns showcase communal ritual play through examples from the following games: *Destiny 2* [35], *Final Fantasy XIV* [246], and *Elden Ring* [82]. Each of these patterns focuses on a different way designs encourage players to form communities around narrative experiences created through ritual play, either through providing players with *incomplete information* (5.3.1), *avoiding helping them* (5.3.2), or *letting them express themselves* (5.3.3).

5.3.1 Incomplete Information

The design pattern of incomplete information focuses on distributing components of the narrative in such a way that a single player cannot comprehensively understand the game with only a single playthrough. Rather, the complete experience is obtained through repetition, wherein players experience new pieces of content, different problem solving approaches, or hidden objectives. All these elements combine to make each repetition a somewhat new experience each time. Additionally, players are given material reason to re-experience the same content over and over, such as getting new rewards or a new bread crumb of narrative. As a result, each repetition has set goals and expectations with the potential to uncover a new perspective on the content.

This relationship between player and game parallels the way new productions of existing play scripts, including historical and ancient texts, function in theater [28, 48, 223]. Combined with the potential for a revelation, this design creates the ritual

experience, yet within that there is room for flexibility and play because the performance of players collaborating is in focus. Each player, much like each member of a theater company, is involved in both understanding and enacting the ritual of the narrative through play. And again much like the way a play script is necessarily incomplete, by designing narratives for repetition via incomplete information, we as designers encourage players to come together to fill in the blanks and engage in communal ritual play.

One of the most common implementations of the design pattern of incomplete information found in *Destiny 2* revolves around breaking up pieces of the narrative across time, space, or both. Seasonal activities in *Destiny 2* are designed to be repetitive but also progress the narrative via voice chatter, broken up across playthrough in a way similar to *Hades* [129]. Additionally *Destiny 2* has run large community scale puzzles like "Corridors of Time [84] which required players to assemble randomly rewarded puzzle pieces that no single player could control all of. Both of these implementations mirror elements of the theatrical production process where it is impossible to fully understand a work from a single read through or single, isolated production.

Both *Elden Ring* and *Destiny 2* very purposefully obscure information about how to progress certain events in their narratives. *Elden Ring* in particular primarily relies on vague hints that are then passed around the community by the game's asynchronous in-game message system or external guides, explaining the requirements and consequences of advancing storylines like the "Frenzied Flame" quest or the locations of illusory walls [2]. *Destiny 2* hides Public Event objectives and entire missions behind

[2]Another infamous example of FromSoftware's propensity for hiding large portions of their games is how *Dark Souls* [81] hides a secret area behind two consecutive illusory walls.

opaque, or even no, instructions beyond something strange spawning in the world at certain times [151]. By hiding the objectives and requirements of optional content, both of these games encourage their players to come together to solve problems and build resources together. Again this mirrors the theatrical production process as it requires the entire company's understanding of a script to make storytelling decisions which build towards collective thematic and artistic goals.

Dungeons and raids in *Destiny 2* and *Final Fantasy XIV* require a combination of trial and error and teamwork, especially when newly released. These difficult activities are often design around mechanics that neither explain themselves to players nor can easily be completed by a single player. Instead it requires coordination across the whole group participating in the activity. As a result, it has become common practice in these games for experienced players to guide new players through these activities, helping them learn, and eventually give them the ability to pass the knowledge on. This mode of interaction parallels the way members of a theatrical production support each other and their creative decisions surrounding a script to discover the choices most resonant with the production's artistic goals.

This is thus the ritual of these activities, as each players' role becomes clear through play and communication, and through the repetition the roles are strengthened, enforced and passed on to new community members. Across all parts of this pattern is the fact that the information gathered by a single playthrough of an experience does not encompass the whole, and one must rely on continual engagement either through repetitive sequential actions or in parallel (by other players) to understand the full

breadth of what the game's experience has to offer.

5.3.2 Not Helping the Player

Another way games can encourage players to be in community with each other is to intentionally inconvenience or trouble them, making it harder to accomplish their goals by themselves. This design pattern focuses on the obfuscation of necessary information, creating friction around basic actions or narrative material. By *not giving guidance to optimal paths* or *distributing the backstory across many fractured instances* it makes it more difficult for an individual player to understand what to do and what is going on. This is not necessarily because the information is secret, such as in the previous design pattern (5.3.1), but rather because the game provides inadequate guidance on how to understand its systems or narrative. By presenting narrative elements to players in pieces and out of order, overwhelming the player with too many disparate pieces of information, or *forcing players through shared difficulties*, one player is not expected to understand the entire experience by themselves, akin to the nature of a play script requiring multiple perspectives and interpretive specialities to collectively transform a script into a finished theatrical work [28, 29, 48, 223]. This is where a player begins relying on the community to map out and collectively understand the connections between each disparate element, as well as optimise strategies for how to play through the game, communally making sense of the mythology of the game.

Elden Ring purposefully does not guide players in how to approach its world or story beyond some very basic explanations for mechanics and highlighting where

to go to progress the story and open up new areas. At the game's launch, numerous players went directly to the game's first major dungeon and hit a wall in the first main boss of the game, having ignored the area in the opposite direction to the dungeon. This common experience led to the wisdom of "go south" [92] being passed around the community both in and out of game as the region at the southmost end of the map was designed to help players level up and get better tools to tackle the dungeon with. Combine this experience with how many items in the game, and in *Destiny 2* and *Final Fantasy XIV*, require lengthy questlines to acquire and players are forced to coordinate with each other to better understand the bounds of the game and where pitfalls might be. One could view the nature of this implementation as paralleling the production histories found in theater which serve as compilations of prior creative decisions and can be the basis of making informed production and performance decisions about a script [48].

Both *Destiny 2* and *Elden Ring* fragment and scatter their narratives to encourage discussion within the community and ultimately foster countless interpretations of their respective stories. *Elden Ring*, as a much more self contained experience does this mostly by leaving extremely large holes in the history, mostly told through short item descriptions and cryptic dialogue, it presents to players, rarely telling its story in a way that leads to any clear cut answers. *Destiny 2* shares much of *Elden Ring*'s narrative approach with its lore books, but has the added wrinkle of parts of the game no longer existing. The original "Red War" story and expansion packs from the game's launch are completely inaccessible save for recordings on YouTube. Fractured narratives

like these, but especially *Destiny 2*'s removal of the first part of its story draw theatrical parallels to both our incomplete understanding of ancient and historical drama (as many plays are lost and presumed destroyed) as well as the concern with mediated theater overtaking live theater I discussed at the start of this chapter. These parallels should remind us that games, sharing theater's nature as an incomplete form of media, are exceedingly fragile in certain ways and the concerns of theater preservation today are very much shared.

5.3.3 Building for Player Expression

Player expression is a powerful tool to let players experiment and make their own narratives or fun, such as seen in The Sims [174] or Minecraft [195]. In a communal setting, however, player expression can take on a new life and grow beyond an individual player's choices. How players act and dress and customize their characters can become a part of the legend and ritual of playing within the community. This design pattern focuses on letting players' deviant play [59] be used for narrative purpose. By letting players express themselves, use suboptimal strategies or equipment, perform unoptimally or breaking the rules, they can shape their own identity through their play and define their relationship to the game's world and narrative by making concrete decisions about who they are. Expression is a core element of theatrical performance and can highlight otherwise non-obvious thematic elements of or bring an entirely new perspective to a story [29, 48, 158, 223]

Many multiplayer games have their own legendary community members, who

become well known for one thing or another, either in of fame or infamy. Let Me Solo Her is one of *Elden Ring*'s most famous player characters for his obvious cockiness thanks to his name, costume consisting of a loincloth and a pot, and singular focus on defeating the game's hardest boss in co-op [54, 235]. But these characters and members of the community are not limited to impressive mechanical feats. Players who devote themselves to sifting through a game's narrative fragments like *Destiny 2*'s My Name is Byf [116] and Myelin Games [198] or extended acts of comedy like RubberNinja's egg eating marathon in *Final Fantasy XIV* [53] can achieve notoriety. They arrive at their place in the community not through singular acts but by consistently playing a role. These figures parallel the way theater has its famous practitioners cementing their place through continuous engagement with the art form, including many of the ones I discuss in this dissertation [29, 33, 158, 175, 223], and whose shadows touch theatrical performance to this day.

Finally, one would be remiss to not mention the many ways player expression can be shaped through player-made activities. *Final Fantasy XIV* is the greatest example of this. Players can own and customize their own houses, and some sub-communities spend great effort findings ways to bend the rather limited interior decorating toolset to their will. This housing system is also a playground for a variety of player-run scenes, such as sprawling night club, theatre, or fashion show communities [221, 269], the performance of play becoming literal theatrical performance or roleplay, inviting reinterpretation on what playing the game can be. These often do exist on a ritual basis, as night clubs and theatres might only be open on certain times. An even grander

example of this is "Lunarcon" an player-run convention that exists entirely inside the game [236]. This repurposing of space is highly remensicent of the way theatrical productions shape their environment to fit their goals [29, 34], including UC Santa Cruz's very own Barn Theater which was converted into a stage in 1968 and turned into an entirely student run theater in 2004 [8, 9].

5.4 Conclusion

With this chapter I hope to again further expand my own definition of theatricality and not lose sight of theater's extremely communal nature, even as my work tends towards singleplayer focused experiences. Additionally, what I wrote here, different from the original Communal Ritual Play paper as it may be [151], should serve as further affirmation that the concept of theatricality I posit in this dissertation is a flexible lens to understand narrative design found in a wide variety of experiences. Furthermore, that the theatrical production process itself can help provide tools to better understand what we as designers ask of players as well as what affordances we give them.

The design patterns here may be focused on multiplayer games, but both *Hades* and *Umineko* exhibit large amounts of incomplete information and not helping the player, with *Hades* also including some degree of player expression. Similar to the games discussed in this chapter, both *Hades* and *Umineko* form rituals around their play. For *Hades* it is the preparation and enactment of a run through the underworld.

For *Umineko* it is each iteration of the murder mystery game.

This observation brings me to a broader point about this chapter covering both why I brought ritual into the discussion of theatricality and what it means for designers. What qualities lead us to call a work theatrical? Theatrical work often embraces its own artifice and frequently deals in the expressive and symbolic, but these qualities are not what I identify as necessary for theatricality. Theater is an art form which grew out of ritual and incorporates many small rituals into its practice still [267]. So what sets theater apart from pure ritual? It is play. Theater plays with its rituals' enaction to avoid their ossification and, eventually, to find new interpretations and meaning in the rituals and stories themselves.

If we view games and other interactive computational systems from this lens of theatricality, we can understand their affordances and constraints as part of the creation of a ritual. Interacting with these games and systems is the enaction of their associated rituals. Like with theater, this enaction can be playful, finding new interpretation and possibilities through interaction. Again, paralleling theater, finding new interpretations, meaning, and even approaches to interaction requires designing these artifacts to facilitate playing with their rituals. This principle is even more important for highly repetitive multiplayer games, such as those I discuss in this chapter, because part of these games' design involves sparking conversations between players amongst each other and between players and the game, and by extension the designers, itself. These conversations can only be sustained through playing with the rituals and narratives the game establishes. In turn, these conversations breath new life into the repetitive elements of

these experiences and the cycle can begin anew.

When we choose to create interactive works, we should strive to engage with our players in both the computational and theatrical senses. Through computation, we can support our players' exploration of our work and facilitate deep, personal conversations with them through play. Through the theater, we can further open our works to playful interaction and change the relationship our players have to our works in the process. After all, why make this work interactive?

It's not my place to provide you with *the* answer to that question here, nor is it my goal. That is a question for you to answer as you work. It doesn't necessarily have to be a deep answer to begin with. When I started working with theater and computation and developing Puppitor, my answer was maybe a little shallow, something along the lines of "performance is interactive! Why don't we make more experiences using it?". Of course, this is still the foundation of my entire dissertation, but it has grown more complicated, as that is the nature of deeply researching and designing something. My ultimate answer to that question is this dissertation, alongside my MFA thesis [128] and my MS thesis [127]. We should be designing more with performance in mind! It is interactive! It also asks a lot from players. It requires you to find the fun in the play. I think that's what we should be striving for as designers, to facilitate play. Yes we can tell stories but we can choose to tell them in a playful way that invites rather than drags players through the experience.

The following chapter details my exploits designing and building an story-telling experience intended to facilitate performance. I do want to note somewhere for

posterity's sake that the chapter order here and even some of the publication order of papers isn't reflective of the order of discovery. My dissertation is here to tell a good story and tell a cleaner version of the truth than the way research and art tend to happen. Mostly I want to acknowledge that the work in the three theatricality chapters very much grew out of my work on systems rather than neatly inspiring the system that would become Puppitor itself. I have an entire chapter dedicated to that piece of the story for a reason after all.

Chapter 6

The Design of an Acting Game

How does one *make* a computational experience centered on performing a character? This chapter is part of my answer to this question. The work I discuss here has spanned my MS, MFA, and now my PhD, with the concepts for the Puppitor library and *Tracks in Snow* visual novel dating back to 2019 [127, 130]. I have written before about the narrative design of *Tracks in Snow* [128] and design of Puppitor [126]. This chapter provides an explanation of the design choices that underpins both Puppitor and *Tracks* by highlight the design insights I have gathered over the five years and three degrees worth of work developing this system and game. In particular, the focus here is on the way designing for performance pulls narrative design to its extremes. On one end, the minutiae of actions and expressions suddenly become incredibly important to facilitating a conversation with the player. On the other, the high-level structure and pacing of an experience greatly influences where players' attention is drawn as well as the ways they can engage with a story at all.

Even if the reality of research is that the previous chapters in my dissertation largely grew out of my time designing and building Puppitor and *Tracks in Snow*, this is one of the chapters where I put my money where my mouth is. You can in fact build a system to play with dialogue in a theatrical way. You can base an entire experience on that aesthetic. It can work too well[1]. I'll show you how.

I do want to note that Puppitor, *Tracks*, and this dissertation all have their origins in what may seem like a simple question: "What if we made the blocking of a scene playable instead of what is said?" As I've mentioned before, I'm a playwright. I love seeing how a production chooses to interpret my scripts and what new insights they bring while still telling my story. That seemingly simple "what if?" question grew out of all my experiences on theatrical productions. Specifically, seeing how much creativity and play a script offered even without being dynamic.

That constrained yet still surprisingly open aesthetic is what I have been chasing with Puppitor and *Tracks in Snow*. Not only did I want to see what even more people would glean and interpret in my stories, I wanted to make my experience of theatrical production just a little more accessible to a wider group of people. My solution to this was to create an interface and AI library and an entire game to prove out my philosophy behind the design[2].

This chapter is split into three main sections. The first discusses how my

[1] This was one of the conclusions in my MFA thesis, that *Tracks* was too successful at being an acting game because of how difficult an experience I'd created. By wanting to create the "fighting game of acting", I'd imported much of the same need for practice and reflection from fighting games and added the difficulty of needing to interpret a story at the same time. It's a huge ask.

[2] And I'll admit get away with developing a yuri visual novel in an engineering department.

design philosophy connects to prior interactive narrative work and what sets it apart. The second focuses on synthesizing my sources of inspiration—acting, puppetry, and fighting games—into an approach to designing interactive narratives. The final section delves into the high level considerations of this style of narrative design, namely the effect of asking a player to be both performer and audience along with insights about using structure and pacing as support.

6.1 An Introduction to Design Considerations

A longstanding thread of research surrounding interactive narrative concerns itself with creating dynamic interaction between human and AI controlled characters. Dynamism in this case often focuses on the outcomes of these interactions altering the narrative's plot in some form [152, 163, 166, 275]. The centrality of outcome in turn de-emphasizes the importance of *performing* the interaction itself. These assumptions are present in many commercial games' dialogue systems [65, 209, 214, 216] as well. But is this the only way for a narrative to be interactive or interactions to be playful?

Commercial games commonly tightly couple the staging, physicality, and line reads of dialogue scenes to the explicit choice of what line of dialogue will be spoken. Everything flows from this single choice and as a result, there is a set number of ways any given scene can be performed. A line will always carry one particular meaning. All of this results in scenes with little room for the creative expression by a player.

With this in mind, if we as designers want to create more dynamic, expressive,

and playful scenes between characters, we should not be satisfied with the power a single decision has over a scene. In other words, we should be decoupling the blocking, physicality, and reading of lines from the text of the line itself. By doing so, we open up dialogue focused scenes as a site of player creativity and expression that doesn't solely rely on the outcome to be made meaningful to the narrative, creating the potential for numerous new kinds of storytelling experiences.

Theater provides a many perspectives on the relationship between performance and text. Practitioners frequently see the interactions between characters as a highly necessary place for play and creativity [19, 29, 123, 158, 223]. An important element of this view of performance's relationship to storytelling is that the text cannot and should not dictate everything [19,29,155]. There *must* be ways for performers to make creative choices about characters, scenes, and ultimately the story itself or the performance is a failure [158, 223]. In other words, performers must be able to play with the text of a script. This is the foundation of my approach to designing experiences around the aesthetics and affordances of performance.

In this section I describe *narrative frame data* as a design approach which decouples the performance of a script from the text of the script and thus create experiences rooted in the aesthetics of performance. Narrative frame data is a method of systematizing interaction in a story to facilitate playing with characters and text in a manner that encourages the replay and reinterpretation of the experience. This approach draws from the development process of the *Tracks in Snow* dramatic visual novel [124, 126, 128] and the design of the Puppitor acting system [126, 130], as well as

performance practices and the narrative design of existing games. I first discuss the influence of acting, puppetry, and fighting games on the formulation of the design of narrative frame data. Next, I describe both how to and why we should decouple the playing of a story from the script of a story. Finally, I discuss the broader effects of narrative frame data on design—including lessons from Brecht, fighting games, and vehicular combat games—and how to keep the player's experience in mind when designing using the approach.

6.2 Related Work

Narrative frame data overlaps many existing concepts in the space of interactive narrative. I begin positioning this approach in relation to interactive drama and its evolving points of reference. Next I discuss narrative frame data in relation to work in transformative play and its goal of pushing back against the dominance of the neo-Aristotelian perspective in interactive stories. I then describe the way work on replay and rewind mechanics in games feeds into the conceptualization of narrative frame data. Finally, I discuss the relationship between computational works of collaborative performance and narrative frame data.

6.2.1 The Roots of Interactive Drama

Interactive drama traces its roots back to Brenda Laurel's introduction of Aristotelian poetics to computing [152, 154], the Oz Project's experiments [165], and the work in the 2000s [44, 238, 259] typified by *Façade* [172]. The neo-Aristotelian

poetics that characterize interactive drama focuses on giving the player agency through their ability to change outcomes within the experience, especially when they alter the structure of the narrative [167]. This focus on outcome makes sense as a derivation of Aristotle's writing, but crucially ignores much of the way theater operates and the way practitioners of the art understand it. Much of the writing and research supporting narrative frame data draws from alternative sources—such as dramatic acting, puppetry, and Brecht—to re-examine the prioritization of changing narrative outcomes through structure.

Even as interactive drama has sought to incorporate the knowledge of theater practitioners, the assumptions around changing narrative structure remain. The reactive lighting system found in *Mirage* itself does not alter narrative outcomes and focuses on highlighting the effect of actions on tone, even drawing from some of the practices discussed in this section [69]. In many ways the system is an example of the approach I advocate for with narrative frame data. But its attachment to a work which does ultimately prioritize agency as understood by interactive drama [68] makes it a murkier than ideal example.

As interactive drama has found new sources of inspiration to incorporate into its design philosophy, it still assumes the foundation of Aristotelian poetics as necessary for interaction. Both *Comme il Faut* [180,182] and Versu [72] explicitly move away from the top-down drama manager found in earlier interactive drama work and drawing from sociology and philosophy respectively for their social models. However, even with these layers on top of interactive drama, there is still the assumption that for a player's choice

or action to matter, they must be able to change the outcome of the story [72, 176]. This continual re-importing of the foundational assumptions of interactive drama is why I advocate for the approach of narrative frame data as an alternative design for interactive storytelling. Even if I root it heavily in theater, what I choose to draw from is distinct from the Aristotelian origins of interactive drama and, importantly, my aesthetic priorities are completely different than those of the neo-Aristotelian approach.

6.2.2 Transformative Play

Transformative play seeks to provide an alternative aesthetic of interaction to the focus on outcomes commonly found in interactive drama and its derivatives. This approach to interaction focuses on the pleasure of enacting a story and playing a role [260]. Transformative play draws on elements of Stanislavsky and Nō as foundations for enabling players to become the role they take on, possibly even transforming a player's sense of self [261]. While transformative play draws from similar sources as I do with the approach of narrative frame data, there is a huge distinction in conclusions. The goal of transformative play is to immerse players in a new identity for them to inhabit and bring them as close to the character as possible. Conversely with narrative frame data, as much as I emphasize the connection between player and character, my interests are not solely in identity and identification with a character. Narrative frame data seeks to give players both the ability to make motivated choices about how they perform a character and the support to constantly re-examine their relationship to their character as well as the story itself.

6.2.3 Replay and Reflection

There is an extensive body of work focusing on the ways players revisit and replay games in addition to the ways games themselves encourage this behavior. When looking to understand how players reread interactive stories, Mitchell and McGee found that players first search for some form of closure and do not consider themselves to be rereading until they find that closure [193]. This form of closure can occur when an understanding of the story or system is reached, and when one is reached before the other, the experience of alternating attention between the story and systemic context can be broken and ultimately prevent players from continuing to replay games [192]. Exploring this idea of closure is highly relevant to narrative frame data as one of the approach's primary goals is to support players replaying and ultimately reinterpreting games as a central aesthetic of interaction.

Rewind mechanics can recontextualize players' relationship to a game's narrative as Kleinman et al. describe [138]. They frequently call attention to the artifice of the games they are present in, much like Brecht's use of stage craft to create similar effects in his audience [33]. When investigating the rewind mechanics of *Elsinore* [89], Mitchell and Kway found that its central rewind mechanic caused the game to actively resist the search for closure previously described in relation to replay, and with subsequent playthroughs in fact feeling aesthetically similar to the initial playthrough [190]. This observation reflects Brecht's, and my own, distaste for Aristotelian drama, and why a lack of catharsis is important for critically engaging with a story [33] and theatrical

performance's need to continuously find something new [29, 158, 223].

Designing for reflection and reinterpretation can take many forms. Junius et al. propose the lens of internal theatricality to describe the way narrative games will recontextualize their narrative via looping structures mirror the theatrical production process and encourage players to continuously re-examine their relationship to a game's story [129]. This notion of theatricality does not require traditionally interactive mechanics [134] and is also present in multiplayer games [151]. This form of theatricality provides part of the foundation for the aesthetics related to narrative frame data discussed in this section. Miller et al. propose a design framework focused on describing design patterns, broadly categorized as disruptions, slowdowns, questioning, revisiting, and enhancers, which support a broad spectrum of aesthetics of reflection [183]. In comparison, narrative frame data, as described here, requires a combination of expressiveness and scaffolding for reflection as design elements, as it is an approach to facilitating performance within stories.

6.2.4 Collaborative Performance

Experiences centered on embodied, collaborative performance naturally draw heavily on theater, either explicitly or implicitly. These styles of experiences have drawn from both improv theater [232, 247] and choreography [118, 266] and generally require players to physically participate in some fashion with minimal props and costuming. As a result these experiences are not mediated performances, though still provide scaffolding for players so they can effectively perform their roles. This scaffolding can take the

form of narrative framing and scripts in the case of *Coffee: A Misunderstanding* [247] and *Bad News* [120] or a digital performance partner in the case of the Viewpoints AI [118] and LuminAI [266]. In contrast to this approach to developing a performance focused experienced, narrative frame data requires the mediated performance of a digital character to achieve its design goals.

6.3 Performance and Narrative Frame Data

To design an experience using narrative frame data, we must first understand how to decouple the way scenes are enacted from the core dialogue and text of a script. As a result, the first part of this section is devoted to understanding the perspective of performance practitioners on how they find creativity within even highly rigid constraints. The final part of this section is focused on using fighting games to further develop the approach of narrative frame data and discuss an example of its usage in Puppitor and *Tracks in Snow*.

6.3.1 Acting

To build an experience which decouples the playing of a scene from the text of a script, finding somewhere to start from is important. Theatrical acting provides solid grounding in the concerns performers have for both their own craft and the audience's experience [223]. With this style of experience, as with interactive drama [152, 167], we as designers are asking players to be both a performer and the audience for their own performance, hence the importance of both sides of an actor's responsibilities.

Three lineages of acting practice provide the foundation for this discussion: Zeami's Nō treatises [223], Stanislavsky's Method of Physical Actions [158], and Bogart and Landau's Viewpoints [29]. These schools of practice have been discussed in relation to interaction design [130] and served as the foundation of a fully realized game system [126], making them ideal for the purpose of understanding what it means to design for the aesthetic of performance.

But this begs the question, what is the aesthetic of performance? Luckily these three performance practices do have a common answer, even as their concerns and styles differ wildly. That of portraying characters requiring novelty and the ability to find the new in the old.

Zeami's writing on Nō theater makes heavy use of the central metaphor of the flower to discuss all aspects of theater [223]. Like a flower, acting must be continuously maintained while understanding that its beauty is both ever-changing and ultimately fleeting [223]. Maintenance for acting of course involves practice and study, but importantly, it must find novelty to remain alive and engaging for both audience and performer [223]. Zeami even explicitly points out that an audience witnessing the exact same performance two days in a row will become uninterested [223]. After all, without the actors' believing in what they are doing, what hope does the audience have to be engaged? Similarly, games which have tight coupling between the text and performance of dialogue make finding new life in their scenes upon revisit difficult because of their inability to support novelty seeking beyond the outcomes of those scenes.

One of the most important elements of Stanislavsky's Method of Physical Ac-

tions is the concept of an *actor's adjustments*, or the exact arrangement of actions and movement on stage [158]. While these adjustments are ultimately collectively decided by the production's rehearsal process and necessarily repeated with each performance of a play, Stanislavsky warns against these becoming fixed [158]. In fact, these adjustments should be impossible for anyone to fully predict [158]. Again, if an actor becomes bored with their performance, so will the audience, even if Stanislavsky is generally unconcerned with the audience's experience compared to Zeami. And again, when games don't support players making adjustments in their interactions, the ability to play with scenes and focus on something other than the scene's ultimate outcome is lost.

Bogart and Landau's Viewpoints seeks to create a common language around composing, staging, and performing scenes and, importantly, making performance a collective process where actors can make creative decisions [29]. These goals create an approach to performance which emphasize the playing with space and time as the core of the creative process, and from which interpretations of scenes and stories emerge [29]. This emergent approach to working with a text is designed to open up the creative choices available to actors and give them the freedom to compose scenes in unique ways, rather than letting a single person dictate what they want [29]. Again, we return to the notion that performance must be dynamic to be satisfying, with the addition that it can stand on equal footing to the text of a script. When the enactment of an interaction or scene is entirely subservient to the choice of a line of dialogue, we as designers make our intent the supreme authority on what a scene means. We prevent the player from bringing their own ideas into conversation with us.

Performance is built on the ability of juxtaposition to influence the meaning of a story even if it cannot change the outcome of any given moment. All three of these performance practices view the relationship between the physicality of a performance and the text as an important site of creative choice. If we as designers tightly couple the physicality of characters to what is being said, we lose the ability to play with juxtaposition. We cut our players off from an avenue for creative expression. We let the flower whither. We let the adjustments become fixed. We dictate what it is we want.

6.3.2 Puppetry

By designing interactive narrative experiences around performance, we as designers are asking our players to become performers. We cannot assume they have any training or experience in this art, and accordingly, we must support them in understanding the character(s) we are asking them to portray. For this reason, puppetry is an important part of formulating an understanding of how to decouple the enactment of a character from the text of a script. As tools of performance, puppets offer concrete affordances and suggestions to performers that parallel many of the concerns of game designers, especially when creating interactive narrative experiences.

There is almost always some level of distance, the level of separation and contact between the performer and the object being manipulated [132], separating a person from a role. Puppetry increases this distance by making the role a completely separate, physical object for the performer to manipulate [132]. This distance in turn shifts the focus of the performance from the puppeteer to the puppet itself [132]. The same

distance between player and character exists within games and interactive narrative experiences. A digital character in a game is an embodiment of a role entirely separate from a player. The affordances of such a character: moving about the environment and interacting with characters and props, controlled via a keyboard or gamepad strongly parallel the way marrionette strings and rods allow puppeteers to perform their own characters.

By having such a strongly embodied character, the character itself can make suggestions about how to perform them [132]. This is especially important for us to keep in mind as designers because, if we want to ask our players to express themselves and their creativity, we cannot simply hand them a script and tell them to act. Making an interactive narrative using the aesthetics of performance, in essence, requires us to become puppet makers. It is our responsibility to make characters which feel capable of life, much like the responsibility of traditional puppet designers [123]. A puppet, in this sense, is a way to avoid the blank page and give players something immediate they can *play with* to understand the character and role we create for them.

A puppet, analog or digital, also calls attention to itself as part of a performance, sometimes to the point of drawing focus away from the dialogue and script [123]. This phenomenon is why we put on plays in the first place. It doesn't matter that we know the outcome of a scene if getting to the outcome is more engaging than the outcome alone. If we want to support players through performance, we need to be designing our characters after puppets. Our characters should suggest their personality and physicality rather than enforce them as is the current standard. Certain kinds of actions

a character can perform may of course be easier or more difficult as flows from their personality, and we can design characters with this relationship in mind. But, if we want players to be able to play with our stories, we need to separate the performance from the script. We need to make characters which our players help bring to life and are a window for human artistry [265].

While we as designers give up control over the exact portrayal of a character, scene, or story when we decouple physicality and staging from the text of a scene, we stand to gain so much more. The interactive component of an interactive narrative no longer has to solely rely on outcomes to be meaningful to a player. They can bring in their own creativity and ideas into conversation with what we designers build. They can breath their own life into characters. Now that we know what the aesthetic of performance we are aiming for, we can understand how to build it.

6.3.3 Fighting Game Mechanics

Performance is concerned with the minutiae of physicality and the construction of many kinds of puppets reflects this priority. A puppeteer often has control over individual limbs, extremities, even fingers [132]. This is in contrast to most kinds of game characters. Players are allowed to control where their character moves, often including how quickly they move. They can perform full body gestures with mechanical purposes, such as rolling or reloading a weapon, or expressive purpose like emotes in *Dark Souls* [81], *Destiny 2* [35], and many other games. However, the interface for each of these actions is often abstracted out to single button presses. While players still find

creative and expressive ways to use these actions, they do not allow for much nuance, either they are performing the action or they are not.

There is an exception in games. Fighting games. In the likes of *Street Fighter* [38], *Under Night In-Birth* [80], *Tekken* [16], *Guilty Gear* [6] and many others, players are not only able to string their character's actions together into combos, many characters have multiple variants of their moves or even can modify particular moves into something new with distinct properties. This aspect of fighting games enables players to develop strategies and creativity about how to play their character [185] in much the same manner as actors and puppeteers describe their craft.

Fighting games generally attempt to model a character's physicality and expressiveness via martial arts [184], often replicating real life styles for their characters. They create this model both through character designs and animations themselves, and, even more importantly, through frame data [121]. Frame data is essentially a markup language which defines the stages of an action computationally. The three stages of every fighting game move are: startup, active, and recovery. During the startup phase, a move will not do damage, but may be invulnerable, allow the character to take a hit or two, or be cancelable into a different move. During the active phase, a move is capable of dealing damage to an opponent. During the recovery phase, the character will be stuck in an animation, unable to deal damage or move.

Frame data provides the base of the deep, systemic character interaction fighting games build their replayability and expressiveness from. Frame data is not limited to describing a move in a vacuum. It also determines the relationship between two

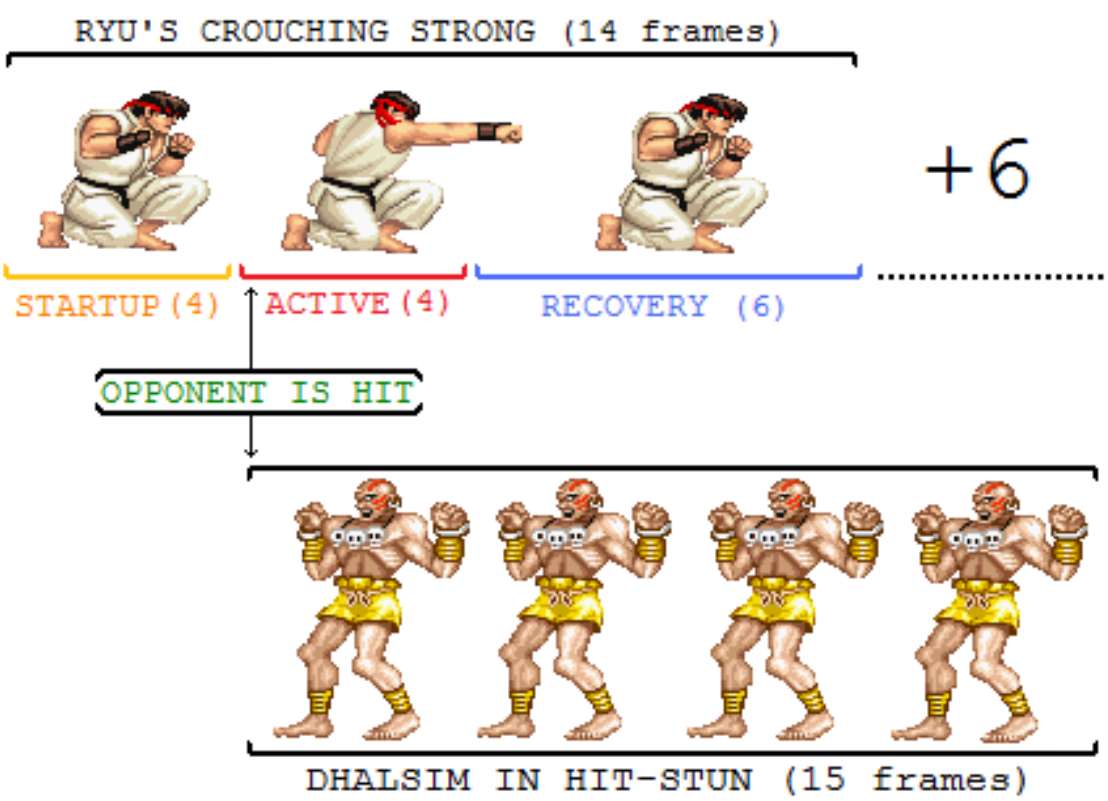

Figure 6.1: Frame data describes the physical interaction between two characters in a fighting game. The startup, active, and recovery frames of Ryu's crouching strong punch determine when Ryu can do other moves, when he is vulnerable, and when his moves are dangers, that is the active frames. When the punch is active and hits a character, the frame data determines how long that character cannot take any other action. In this example Dhalsim is stunned for 15 frames while Ryu only has to recover for 6 frames. Thus, Ryu can follow up his punch with another move, including the crouching strong punch, and hit Dhalsim again before he can react. Sourced from [121].

characters depending on if a move hits, whiffs, or is blocked [121], described in figure 6.1. On a successful hit, moves will generally carry some degree of advantage for the attacker, either via putting their opponent into hit-stun or becoming cancelable into other moves. When a move is blocked, the ability and safety of followup attacks can be drastically different and allow opponents openings to start their own combos or at least avoid more damaging followups.

This system is why the execution of combos themselves can be forms of expression similar to what actors and puppeteers describe with their respective crafts. A character's combos can require varying degrees of precision to execute in addition to the number of moves required and the amount of damage they do, all thanks to frame data. Choosing difficult combos that require precision down to single frames, even for only minor additional damage, is a way for players to express confidence in the way they perform their characters, and, importantly, is a choice about how to use a character's affordances. In this way the frame data, in addition to the animation, visual design, and input style of a character suggests ways of playing them [186] much the way a puppet does [123,132]. This design allows players to have their own distinctive styles of playing and portraying the same character and even express their frustrations to each other through playing their characters [3].

What this discussion of fighting games and frame data ultimately gets us as narrative designers is an existing model of physicality and expression. A model of martial arts is still a model of performance, given the intertwined history of martial arts

[3]During the commentary for Evo Moment #37, Daigo's impossible comeback against Justin Wong at Evo 2004, one of the casters comments about Daigo being angry at Justin's playstyle [73]

with acting and dance [117,249]. While frame data in fighting games exists to model the relationship between characters physically striking each other, it is ultimately a way of creating a simple, deterministic, and expressive way of describing the effects of actions on characters. We can then take this more general description to create *narrative frame data* as an approach to decoupling performance from text in interactive narratives.

6.3.4 Designing with Narrative Frame Data

The concept of narrative frame data underpins the design of the Puppitor system and the *Tracks in Snow* visual novel. While this term hasn't been used to describe the design philosophy [126,128] before, both the system and experience exemplify what it means to create an experience around performance using narrative frame data. Puppitor explicitly takes inspiration from Zeami, Stanislavsky, and Bogart and Landau as well as fighting game inputs to build its paradigm of actions expressing emotional affect [126]. *Tracks in Snow* uses the system to create characters who feel physically distinct to play thanks to the underlying differences in how their actions convey particular emotional affects [126, 128]. These descriptions of each character are codified in rule files linking actions to exact changes in a character's expressive state which are then broadcast to and interpreted by other characters in a scene [126]. In other words, the rule files themselves are an instance of narrative frame data. Through Puppitor and *Tracks in Snow*'s design around narrative frame data, players can create tone and pacing of scenes completely unique to their interpretation of the characters without the need for a highly branching narrative structure, demonstrated in figures 6.2 and 6.3.

Figure 6.2: A line of dialogue from the first scene of *Tracks in Snow*. Rika, the character on the right is sad while Chiara, the character on the left is angry. The typography of Rika's line of dialogue uses a light font weight to reflect her state, along with her facial expression and the game's background music.

Figure 6.3: The same part of the scene shown in figure 6.2, but this time Rika is angry and Chiara is happy. The typography of Rika's line uses heavy font weight to again reflect her state, along with her facial expression and background music.

When characters suggest rather than impose affordances, they invite play. Frame data may be mathematical and fixed in certain ways, but like sheet music or a play script, its combinatorial nature allows for depth when playing a character that few other approaches can match. If we as designers want to create more playful ways of interacting with characters and decouple the performance of a character from the script we use, we should take more cues from fighting games. Our ultimate goal should be to enable players to learn the affordances of their character and support them in expressing their version of a character through play. With this in mind, we should seek to create designs around narrative frame data. We should look for new ways for actions to convey meaning which can then be interpreted by other characters to drive their own action and performance. We should design more experiences around the aesthetics of performance.

6.4 Brecht and Playing with Aesthetic Distance

As storytellers and designers, we must keep our players in mind when developing our experiences. Much the way theater failing to engage with its audience is nonsensical [33], an interactive storytelling experience which fails to consider its players will fail. The discussion of narrative frame data and dramatic performance so far provide us with guides to creating playful interactive stories, but their concern generally lies with how to make play and performance happen, not with the audience's experience. As much as designing with narrative frame data in mind can turn players into actors, a

player is not just an actor. They are also the audience.

The rest of this section focuses on this aspect of the player's role. Our designs and stories must still engage someone who may never have thought about performance in any way and may have no experience as a storyteller. We are still telling our stories to players even when we ask them to perform them for us. Acting, puppetry, and playing fighting games all rely heavily on their players' ability to reflect on performances [19,184, 223]. When we design with narrative frame data and for the aesthetic of performance, we must design space for this reflection into our experiences. Without moments to rest and reflect, players' performances will become exhausted, slide into meaninglessness, and ultimately unenjoyable as both a performer and audience member [33].

How do we know where to put these moments of rest and reflection or how frequent they should be? I begin this section exploring how Brecht, fighting games, and vehicular combat games like *FreeSpace 2* [271], *Armored Core: For Answer* [115], and *Chromehounds* [114] each provide answers to this question. The thread connecting these three seemingly disparate sources is that of dynamism. Specifically, dynamism through engaging a player or an audience through both emotion and intellect [33]. This dynamism is known as *aesthetic distance*, the intensity of emotional or critical engagement with a moment in a story [48]. Low aesthetic distance is often characterized by an audience's deep emotional connection with an event while high aesthetic distance frequently involves an audience critically engaging with the story's developments [48].

When we have players perform a character, we are reducing their aesthetic distance of the experience and directly entangling them in the emotions of the work.

Conversely, when we have players passively experience elements of a story, we are increasing the aesthetic distance of the piece and providing space for critical engagement with themes. This isn't to say there cannot be moments of high aesthetic distance when a player is performing in a scene or moments of low aesthetic distance when passively experiencing a story. These moments are a natural component of storytelling, interactive or otherwise. But for the purpose of designing around narrative frame data, it is useful to know which direction active performance tends to pull the aesthetic distance.

I end this section discussing how these elements are exemplified in the structure of *Tracks in Snow* and its use of scene transitions. *Tracks* features many of the same constraints intensely low aesthetic distance places on experience design, particularly because it so closely parallels fighting game design [128]

6.4.1 Brecht

Central to Brecht's concept of dynamism in theater is the removal of the audience's ability to experience catharsis [33]. By withholding this moment of catharsis, an audience is brought into a critical mindset even as they watch a play [33]. Important to Brecht's approach to removing catharsis is that a play must be able to both induce emotions and provide space to investigate those same emotions [33]. If a play is all reason or all emotion, it becomes static and disengages both the performers and the audience [33]. For this reason, when designing using narrative frame data, we must be able to change the aesthetic distance of our stories. Without it, we cannot ask players to reflect on their experiences or engage deeply with our work. Without reflection, there

is no room for players to grow and learn and develop a relationship to the stories we tell. To accomplish the goal of narrative frame data and, more broadly, the decoupling of performance and text, to create dynamic stories for players to explore and express their creativity within, we must help players reflect on their performances.

6.4.2 Fighting Game Structure

To return to fighting games, when looking for how to build space for reflection, we can look at these games' commonly used match structure. A typical match in *Street Fighter*, *Under Night*, or *Guilty Gear* begins with each player selecting a character and ends when either player wins a total of two rounds. A round generally has a 99 second time limit and ends when one player knocks the other out by dealing enough damage or when the limit is reached. Between each round is a short break, which can be made shorter by skipping victory animations, as characters are moved back into place and the timer resets. At the end of a match, players are usually given the option to play a rematch, return to the character select screen, or quit to the main menu. While fighting games are first and foremost modeled after martial arts tournaments [184], including their moments of respite for players to recover their breath and recenter themselves.

There is no time limit for staying on the post-match screen and this importantly builds space directly into the game for players to reflect on how they performed. Playing the individual rounds of a fighting game is an intense and often emotional experience [187], or minimal aesthetic distance. Conversely, the end of match screen allows the hightening of aesthetic distance. This is the dynamism necessary for a performance

oriented game, one built using narrative frame data or otherwise. Without the end of match screen, playing a fighting game would become more exhausting than it already can be and there would be no space for players to reflect on their relationship to their character and the game.

6.4.3 Vehicular Combat Structure

The vehicular combat games most relevant to this discussion of dynamism in narrative design are mission based, usually singleplayer experiences such as *FreeSpace 2*, *Armored Core: For Answer*, and *Chromehounds*. These games structure themselves around a briefing, mission, report formula [4]. The mission sections of these games are the only place exposing combat mechanics. In contrast, the briefing and report sections are almost entirely devoted to characters setting up the expectations of combat and discussing the consequences of the combat respectively. Like fighting games, the combat mechanics and speed of these vehicular combat games are intense, requiring a high degree of focus to successfully complete their objectives. Also like fighting games, the continuous time spent playing their combat is relatively short, rarely lasting more than 10 minutes [5].

As much, if not more time, is spent in the briefing and report sections of each of these games. The briefing sections in particular can last just as long as missions while consisting entirely of reading and listening to characters contextualize the actions

[4] This structure bears some resemblance to the *jo*, *ha*, *kyū* structure Zeami describes, where actions begin in an easy manner, develop dramatically, and finish rapidly [223]

[5] "Exodus" in *FreeSpace 1* [270] is the only mission across both retail *FreeSpace* campaigns which lasts 20 minutes.

the player is expected to perform. This slow lead up to the eventual dramatic action allows players the opportunity to consider their relationship to the narrative before engaging with it. Additionally these games enable the player to mechanically prepare for performing their tasks in a manner similar to *Hades* [255] run structure [129]. The report section is generally much shorter than the briefing but importantly provides a way for the game to directly acknowledge actions the player took.

The structure of briefing, mission, report sandwiches the intensity of the low aesthetic distance of combat between the calmness of the high aesthetic distance of narrative context, giving players ample space to reflect on their approaches to playing these sorts of games. Again we have the dynamism Brecht discusses. Again we give players the ability to be both entangled with emotional experience of play and reflect critically on their interactions with the game and story.

6.4.4 Designing with Dynamic Aesthetic Distance

Tracks in Snow follows the design structure of both fighting games and vehicular combat games. Each acting scene begins with a lead in and ends with a moment free of performance. These breaks exist as either an animated introduction or a short transitional scene described entirely through on-screen text. These short respites function similarly to the structure of fighting game matches or vehicular combat missions. They provide places for players to rest, prepare, and reflect on their performance as they progress through *Tracks in Snow*'s story. They are moments of high aesthetic distance, in contrast with the low aesthetic distance of performing a character in a scene. As with

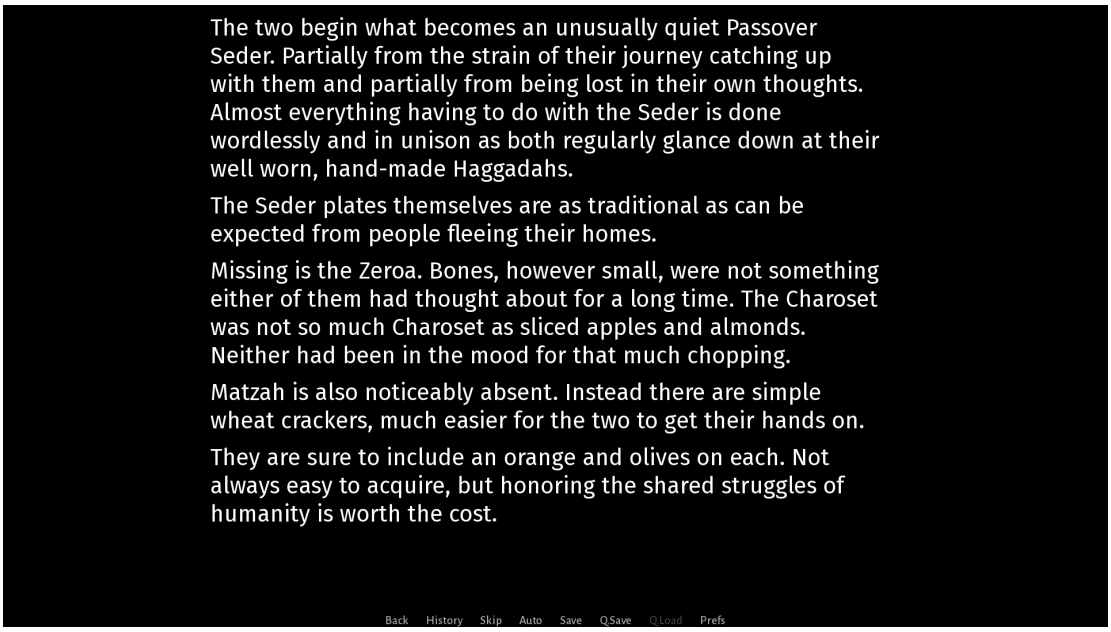

Figure 6.4: The first text screen from the scene without dialogue in *Tracks in Snow*. This scene serves to focus players on the story rather than having to balance between paying attention to the plot and performing a character.

fighting games and vehicular combat games, without these transitions the experience would become exhausting and eventually static as Brecht describes.

In addition to putting these short breaks between scenes, *Tracks in Snow* features a scene composed entirely of on-screen text half way through its script. This intermission of sorts serves both to create a scene that would be impractical using Puppitor and to provide important narrative momentum for the second half of the story. Asking players to perform such a scene would shift their focus away from what the scene means for them in future parts of the story. Lessened aesthetic distance to the script combined with the characters themselves overwhelming the text [19, 123] would make this pivotal scene lose its greater purpose: that of recontextualizing the characters'

situation within the story.

Aesthetic distance is more than just a tool for managing players' interest in the stories we tell. We can use aesthetic distance to scaffold the relationship players develop with our stories. When combined with the approach of narrative frame data, the usage of dynamic aesthetic distance allows us to fully embrace the aesthetic of performance and design with both action and reflection in mind. In doing so, we not only give players the ability to return to stories with a fresh perspective, we give them the space to further deepen that perspective and, ultimately, their relationship to the narrative.

6.5 Conclusion

In this section I argue for the necessity of decoupling the portrayal of a story from the written text of the story to support players in developing deep, playful relationships to interactive narrative experiences. I posit the concept of narrative frame data as one approach to decoupling these two aspects of storytelling and define the term as: a method of systematizing interaction in a story to facilitate playing with characters and text in a manner that encourages the replay and reinterpretation of the experience.

This section explores the origin of narrative frame data as well as the goals in its usage by unpacking the elements used in its synthesis as well as providing an example of the kind of experience the approach can create. Theatrical acting's constant search for novelty and meaning provides one of the aesthetic goals to design for with narrative frame data. The mediated performance of puppetry serves both as an example

of the power in creating characters which can be physically played with as well as an understanding of the way these characters call attention to themselves. Fighting games bridge traditional performance and the world of computation by providing a model of martial arts performance along with their control schemes paralleling many aspects of puppetry.

While not an aspect of narrative frame data itself, it is important to understand the relationship between structure, pacing, and player when building work using this approach. The concept of aesthetic distance is especially useful to designing games around the aesthetic of performance, as performance requires players to both become emotionally and critically engaged with a given work. Brecht views dynamic aesthetic distance as a necessity for creating plays which resonate with an audience. Without it, the play will become static and die in the same way other theater practitioners describe. When it comes to constructing an experience using narrative frame data, fighting games and vehicular combat games provide a guide for developing structures to support players' long term engagement with a work. Fighting games and vehicular combat games are both cognitively and physically demanding experiences. As a result these genres are built around short windows of interacting with their respective combat systems and periods of down time for players to decompress, rest, and reflect on their performances. Games designed using narrative frame data are similarly intense experiences [128] and should be paced with these constraints in mind.

I use Puppitor and *Tracks in Snow* as examples of work designed and developed using narrative frame data. This discussion includes the mechanical design built

around modeling character physicality and expressiveness as well as the high level narrative structure and pacing of the experience. An experience designed around narrative frame data is emotionally and intellectually demanding by design. In essence, creating performance focused work is asking a player, who likely has no formal training, to simultaneously enact parts of a story while constantly re-evaluating their relationship to that story. For this reason, it is important to structure experiences around dynamic aesthetic distance and moments of rest, as *Tracks in Snow* does, to avoid players burning out or the experience itself becoming monotonous.

By calling to decouple the portrayal of a story from the written text of a story here, I hope to inspire my fellow designers and researchers to create more experiences built around playful interaction with characters. My description of narrative frame data provides one potential approach to designing this kind of novel storytelling and even hopefully more sources of inspiration to draw from when creating systems and interactive narrative experiences. While narrative frame data is heavily focused on the interplay of enaction and the text of a story, its emphasis on character expression through physicality has potential relevance to social modeling and simulation.

Chapter 7

Developing an Acting System

Here is where I fully combine the two main strands of my research—theater and computation—into the Puppitor library. It is a codebase which balances the slippery, ethereal nature of performance and the rigid, structured world of computation. It is a system which translates the lessons of acting into a form playable on a computer which can then be used to tell stories and facilitate characterful performance. This is the chapter where I lay out exactly what Puppitor is, beyond the spectre that has haunted me for almost six years, and is the what would fall out of this document if you cut it.

This chapter is edited from my original paper describing the Puppitor system [126]. In it, I first discuss the way the system grew from interactive drama and my motivations for making the system. Second, I describe the process I used to synthesize a number of theatrical process into a computational model of character acting. Next I detail Puppitor's core architecture and functionality. Finally, I describe its integration with *Tracks in Snow* and how it acts as both a game interface and domain for AI

systems.

7.1 An Introduction to Puppitor

Interactive drama has sought to tackle the question of how to allow the audience to become a player in the narrative by drawing heavily from theatrical practices and theory. For the systems that explicitly describe themselves using this term, the ability for a player to change the plot structure and direction through their choices (plot action) is a central component of turning an audience member into a player. However, when looking at the way theater itself is created, particularly practices that surround acting and its relationship to a script, this preoccupation with changing plot structure bears little resemblance to the art of creating drama, described as embodying and giving life to a literary character through actions on stage (stage action) [158]. Plot action is concerned with the events of a story, what happens, who is involved, what order do the events happen in. Stage action is concerned with the details of the way those events are enacted, how does a character say something, what is the reaction to a movement, how does a character convey a feeling. Interactive drama asks *what* characters should do in a story. For theater, *how* characters act in a story is equally important.

Therefore, the question of how characters express themselves and how interaction with characters is conducted by the player, is significant to any study and development of AI characters in interactive drama systems. Character expression and reaction is central to the feedback players use to determine how to react and reason

about AI characters and shapes their expectations for the interaction. Further, the interface and/or control scheme the players use to communicate their intentions back to an AI character shapes their understanding and experience [229].

Many existing interactive drama systems' approaches rest on the assumption that constructing or altering plot structure based on actions and events taking place during a simulation is the way to include a player in a drama. For example, Façade enforces a degree of plot structure using its underlying reactive planner written in ABL [169] to create a coherent and focused narrative while giving the player a wide array of choice in how they act. That is not to say that these systems lack expressivity for their characters. Versu [72] takes a more distributed approach to storytelling by allowing characters', whether AI or player controlled, pursuit of their own goals to drive conflict in the plot rather than enforcing a more structured experience and uses character's subtle expressions as a core part of interaction.

While these systems support expressivity for their characters, their emphasis and focus on altering the plot tends to turn this expressivity into another means for impacting the plot progression rather than facilitating the exploration of a story through dramatic play. Dramatic play allows the player to embody the character through actions and in doing so focus on how the characters express their emotional reaction rather than how or even if that impacts the overarching sequence of events. While both approaches have merit, the exploration of player expressivity and play with AI characters in the context of dramatic play has been comparatively under-explored.

In this chapter, I present *Puppitor*, a system for character expression of emotion

for interactive storytelling. Puppitor centers expressivity and dramatic play to allow players to focus on interaction with AI characters rather than focusing on plot action. Puppitor is a computational caricature [243] of physical theatrical acting practice, that allows players to control the characters' physicality in a narrative space similarly to how they might control fighting game characters in a combat context. Built on existing acting practice [29, 158, 223], Puppitor provides an embedded domain specific language for specifying character rules for how performing actions convey emotion. In this chapter, I describe Puppitor in detail, and showcase how it enables varied dramatic character expression through a case study of the visual novel *Tracks in Snow* [124].

7.2 Related Work

7.2.1 Interactive Drama

Interactive drama traces its roots back to Brenda Laurel's dissertation [154] and the development of interactive Aristotelian dramatic theory through the Oz project [165] and *Façade* [167, 169, 173]. Foundational to interactive drama is the idea that plot must react to a player's actions. In pursuit of more simulation oriented approaches, psychology and sociology were leveraged to develop and augment interactive drama systems [72, 180, 181, 238] such as *Comme il Faut* which draws from Goffman's social dramaturgy to guide its creation of modular social exchanges [180]. Other systems are not as steeped in the Aristotelian approach and draw from a variety of sources [45,68,259] such as the story engine from *Madame Bovary on the Holodeck* explicitly using Gustav

Flaubert's inventory of feelings to direct character goals [45]. None of these systems, however, refute the core assumption of the Aristotelian approach: that changing the plot is core to interactive storytelling.

These interactive drama systems commonly have some sort of emotional model as part of their character definitions, from *Madame Bovary*'s inventory of feelings [45] to Versu's emotional states based on Ekman's typology [72] to *Mirage*'s usage of *Hap* supported by an emotion model similar to FLAME [68, 70]. When compared to a system specifically designed as an emotional model, such as GAMYGDALA [210], the distinction between an emotional model and performance model, such as Puppitor, becomes clear. First, GAMYGDALA uses psychology as its theoretical foundation while Puppitor uses theater. GAMYGDALA implements a specific model of emotion, OCC [204], and using the system requires use of that particular emotional model whereas Puppitor is designed for use with multiple different theories within its framework beyond the Stanislavsky, Nō, and Viewpoints domain described in this chapter.

7.2.2 Acting Practices

The primary influences for Puppitor' s design are Stanislavsky' s Method of Physical Actions [158], Viewpoints [29], and Nō theater [223]. Each provides a unique perspective on how to express emotions and character through physicality. For a more detailed discussion for how these practices relate to characters in digital games see [130]. I summarize acting practice influences for Puppitor' s design specifically below.

Over the course of his career, Stanislavsky iterated his acting philosophy nu-

merous times, the most well known of which is the Method with its emphasis on analyzing character objectives [158]. The Method of Physical Actions, a later iteration, emphasizes the connection through physical action between actors on stage [158]. Puppitor draws from the latter to model the way gesture conveys feeling. Central to the Method of Physical Actions is the idea that actions on stage are all performed with some amount of energy flowing from somewhere within an actor towards someone else on stage. Stanislavsky calls these actions and their changes over the course of a sequence of performances an actor's adjustments [158].

Like the Method of Physical Actions, the Viewpoints are a reaction to the internal focus of Stanislavsky's original Method, stemming from innovations in choreography in the 1960s and 70s [29]. Anne Bogart and Tina Landau define the Viewpoints as a set of names given to certain principles of movement through time and space, constituting a language for talking about what happens on stage [29]. They use nine Physical Viewpoints separated into the categories of Time and Space. Puppitor draws from the viewpoints to define the way gestures can be held and repeated by players.

In his writings on Nō theater, Zeami focuses heavily on the audience's experience of a performance and how an actor's physicality can emphasize elements of a script but the script importantly provides context for that physicality [223]. He describes this as "communicat[ing] first by hearing, then by sight" and describes how actions preceding their context will have diminished readability and weight [223]. Rather, he suggests that a person's intentions give way to their behavior and that the context the words of a script provide ground those actions [223]. Importantly, the way an actor speaks

and gestures should align both with each other and with the text. Puppitor uses this inspiration to guide the creation of its set of actions to be expressive and fit a wide range of contexts.

7.2.3 Applications of Theater in Computational Media

The aforementioned acting practices have been used in conjunction with prior computational media work. Stanislavsky's volitional objectives have been used as inspiration for the character behavior system in *Mirage* [68] and his active analysis technique (an evolution of volitional objectives) has been used as part of a framework for crowd sourced social simulation training [75]. The Viewpoints have been used as part of allowing AI systems to interpret human gestures [118]. While both Puppitor and the Viewpoints AI use Bogart and Landau's Viewpoints as their foundation, each system uses the acting practice for different ends. The Viewpoints are rules for interpreting live human gestures in the Viewpoints AI [118]. In contrast Puppitor uses the Viewpoints as part of the design of the interface on a more standard game controller (the keyboard). The use of masks in Nō theater as well as Stanislavsky's Method have been noted as inspirational for work in transformative play [260, 261]. While all of this work in computation does draw from similar acting practices as the ones underpinning Puppitor's design, their primary goals are not character expressivity and they thus draw from different aspects of each practice.

7.2.4 Fighting Game Characters

Theater practice does not provide a sufficient guide for developing the computational underpinnings of a system for character expression. There is, however, an existing model of computational physicality that shares some of the same values as Puppitor: Fighting games.

Fighting games are an abstraction of martial arts [187]. At their core, fighting games rely on tracking a character' s state over time for each move they do using frame data [121]: the information determining how long a move takes to be able to damage an opponent, how long it can damage an opponent, and how vulnerable it makes a character when they miss, hit, or are blocked. This abstraction matches the level of detail to what was desired for Puppitor' s relationship to theatrical acting as fighting games tie a single action to a button press while allowing for those actions to be modified, such as pressing a button making a character punch but adding a specific motion allowing them to throw a fireball instead.

Fighting game frame data usually specifies a sequence of states: startup, active, recovery, roughly simulating the physics of throwing a punch or kick. Frame data became the basis for translating the relatively coarse actions and modifiers in Puppitor into more nuanced expressions of emotion and going beyond each action being tied to a single emotion. Puppitor allows for fluidly updating all of its emotional affects with each performed action, roughly simulating Stanislavsky' s description of energy flowing out of internal movements.

Figure 7.1: An example of a character's move list in *Tekken 7*. Each button corresponds to the characters limb and the exact details of the way the character uses each limb changes based on the direction of movement, previous buttons pressed, and if multiple buttons are held simultaneously.

Puppitor does not seek to replicate the exact technical complexity of inputs found in fighting games like *Street Fighter IV* [39], *Tekken 7* [16], or *Under Night In-Birth* [80] though it still takes direct inspiration for aspects of its interface design. *Tekken* maps each of its four buttons onto a specific body part and action combination: left punch, right punch, left kick, and right kick, then lets each of those buttons be modified based on the direction of movement, shown in Figure 7.1. For example standing left punch is a completely different kind of punch than forward left punch even if they both rely on the character's left arm. In contrast, *Under Night* simply labels its buttons A, B, and C broadly mapped onto increasing reach and damage, with A being the shortest, fastest moves and C being the longest, slowest ones. What exactly A, B, and C correspond to in *Under Night* is entirely character specific, one character's A might be a short kick, a punch with a sword hilt for another character. Additionally, many moves in *Under Night* require more complex individual inputs, when compared to *Tekken*, combined with a button press to say, throw a projectile. Puppitor's interface uses the broader connection between button and action from *Under Night*, mapping a button to a general category of action, such as mapping N to the *open flow* action. The modifier keys act more closely to directions in *Tekken* where combining *open flow* with *tempo up* is an action with some different characteristics than the neutral version.

7.3 Translating from Theater to Computation

Puppitor is the synthesis and translation of the three aforementioned acting practices, the Method of Physical Actions, the Viewpoints, and Nō, into a computational system built around the categories of *actions* and *modifiers*. Puppitor's design is also intertwined with the development of *Tracks in Snow* and some of the systems features grew out of the parallel work. Actions are major changes in a character's physicality which directly carry emotional weight. Modifiers are small changes applied to actions which alter the relationship of actions to the expression of emotions. Puppitor's architectural decisions and connection between actions and expression mostly draw from the Method of Physical actions and Viewpoints while the way the system relates to narrative and its approach to using text to contextualize its library of actions draws heavily from Nō.

Puppitor's interaction paradigm, the pressing and holding of keys on the keyboard corresponding to actions the character on screen performs, are directly taken from the Viewpoints of Behavioral and Expressive Gesture, Duration, and Repetition. The on screen actions mapped to keys on a keyboard as defined in the Action Key Map (see Figure 7.2 and Section 7.4.1) represent Behavioral Gestures as described by Bogart and Landau [29]. A behavioral gesture is a physical action which represents the every day reality. Additionally, Puppitor's mapping of on screen actions to physical keys can be understood as Expressive Gestures [29]. Expressive gestures are symbolic and stylized with the goal of evoking particular emotions or concepts. In *Tracks in*

Theater Practice	Purpose	Computational Component
Viewpoints of Behavioral and Expressive Gesture, Duration, Repetition	Give players control over character's physical expressiveness	Direct player control over Puppitor actions and modifiers
Viewpoint of Tempo	Visual feedback	Labels and visuals for Puppitor modifiers: Tempo Up, Tempo Down, Neutral
Stanislavsky's concept of energy flow	Names for actions which allow character specificity from other sources	Labels and themes for Puppitor actions: Open Flow, Closed Flow, Projected Energy, Resting
Stanislavsky's concept of action extending from one actor to another	Physical expression of one character has a direct, readable effect on other characters in a scene	Puppitor actions and modifiers broadcasting an emotional affect
Zeami's discussion of role play	Create a set of actions which allow any performance to be in character	Small set of widely applicable gestures as a palette for players in *Tracks in Snow*
Zeami's communicate first by hearing, then by sight	Enable a small number of character poses to fit a wide variety of scenes and tones	Using text in *Tracks in Snow* to add specificity to broad, stylized gestures
Zeami's flower, Stanislavsky's adjustments, Bogart and Landau's relationship to a play script	Enable each action a player takes to create new and unique moments within scenes	The realtime simulation of *Tracks in Snow* and the existence of Puppitor's modifiers

Table 7.1: A mapping of ideas and inspirations from theater practices onto their corresponding computational embodiments found in either Puppitor or *Tracks in Snow* as well as the purpose behind the design choice.

Snow, the gestures lean more expressive as they are built around expressing particular emotions but the exact poses as rendered in the character sprites are closer in design to behavioral gestures. The Viewpoints of Duration and Repetition are thus a property of placing gestures under player control through pressing keys on a keyboard. By pressing and holding a key mapped to a Puppitor action, a player chooses the duration of a particular gesture. Similarly, the frequency a player chooses to return to a gesture or sequence of key presses, and by extension gestures, they are deciding the amount of repetition present in their performance. I chose these Viewpoints specifically as the basis for giving players control over their character's physical expressiveness because a gesture is a relatively contained aspect of physicality, making it easy to add additional simulation relevant information when they are performed. Gestures are also a common way for players to systemically interact with other characters, usually through physical violence, such as the punches and kicks of fighting games or the sword swings of real time action games like *Dark Souls* [81] and The Witcher 3 [216].

The connection of action to the expression of affect in Puppitor's Affecter (see Figure 7.2 and Section 7.4.2) is heavily inspired by Stanislavksy's concept of energy. For Stanislavsky, every action on stage must be performed with a particular flow of energy in mind with the goal of garnering some kind of response from a fellow actor [158]. I took Stanislavsky's concept and split it into two parts: the general flow of energy between characters and what those flows conveyed. Puppitor's actions are how characters collectively define the relationship they have at any given moment. In *Tracks in Snow*, the labels for these actions are taken directly from Stanislavsky: open flow of energy,

closed flow of energy, and projection of energy, with the addition of a resting state to allow players to re-center their characters. In Puppitor, as with Stanislavsky's Method, actions are a conversation. Just because one character wants to be open to another does not mean the other will reciprocate. To add extra specificity to these energy states, and ultimately make *character definitions*, I connected the valence of specific emotional affects to Puppitor actions. I use valences rather than a hard mapping of one action expressing one emotional affect to both allow the previous gestures and expressions to have a bearing on the current performance and enable more compositionality at design time as well as during play. The result is that the what actions, and ultimately gestures, characters use to convey particular emotional affects is specific to them both systemically and artistically at a sprite level, in the case of *Tracks in Snow*.

Puppitor's usage in *Tracks in Snow* is heavily influenced by Zeami's Nō theater, namely his discussion of roleplay as well as the relationship between physical action and dialogue. Zeami describes the roleplay element of acting as not just imitation through action but also through costuming, especially when playing women, as dress conveys a significant amount of character [223]. The character sprites of *Tracks in Snow* are, in this sense, costumes[1] for players to wear and aid them in their performances. A less obvious, but no less important, part of that costume are the character rules themselves. Like costumes and character sprites, they suggest details about the character and, in *Tracks*, they offer a specificity to support players in fitting comfortably into the role as defined by the script. These concrete elements of a role are important as I want

[1] As well as puppets.

characters to feel distinct from each other. Not only visually in how they perform gestures but also literally how they feel in players' hands while they play while also making every available action fit the character.

The exact nature of the dynamic relationship between a character's dialogue and Puppitor actions is drawn from Zeami's notion of "communicat[ing] first by hearing, then by sight" [223]. For Zeami, a gesture can mean any number of things when left on its own, especially when the gesture is heavily stylized. The words of a script thus solidify what a gesture communicates, for example when an actor first says the word "weeping" then wipes their face with a sleeve, the audience understands the gesture explicitly as brushing away tears. Zeami's concept is hugely important to allow the small number of broad character poses of *Tracks in Snow* to fit a wide variety of situations, scenes, and tones through the juxtaposition of action and dialogue to avoid creating an exponentially growing authoring burden.

When described by Zeami, Stanislavsky, and Bogart and Landau, acting, performance, and theater are all arts which seek novelty. Every night of a show must be a little different. No two moments can ever be identical. The performer must remain interested in their role. For Zeami this is the flower [223]. For Stanislavsky these are an actor's adjustments. For Bogart and Landau it is the communal interpretation. For Puppitor and *Tracks in Snow*, these ideas are not only represented in the modifiers, keys which slightly alter the rate of transition between a characters affects, they are embodied by the realtime nature of the experience. Reproducing an exact moment with Puppitor in *Tracks* is extremely challenging if not impossible at times. The system em-

bodies these three traditions' collective view of the nature of performance through its slippery, continuous state updates and insistence that *something* is always happening, even if it is subtle. The scenes and moments Puppitor creates are unique just like every night of a play. They are personal. They are intimate. They are made possible through careful and deliberate system design.

7.4 System Architecture

Puppitor is broken up into two primary modules, the *Action Key Map* and the *Affecter*. First, the Action Key Map translates player or AI input into an action and modifier for a character to express. The Affecter then uses this information to update that character's *Affect Vector* (their emotional state) based on the *Character Affect Rules*, specified in a JSON file. The Character Affect Rules describe the way every action taken and modifier applied will change values in an Affect Vector each update cycle. The action and modifier expressed out of the Action Key Map and the values stored in the Affect Vector can then be used outside of Puppitor's core modules to drive character animations, change facial expressions, alter musical scores, and any other methods of feedback a designer may want. Puppitor is intended to be a library and used beyond the visual novel developed alongside it. Therefore, the specifics of how the system's outputs are used to alter visuals, text, and audio is necessarily project dependent.

As part of implementing feedback in the visual novel *Tracks in Snow* (see the

case study), the Python version of Puppitor includes an animation system, designed for use with standard frame by frame sprite animation. This system not only enables dynamic visuals for the system, it serves as an example of how Puppitor's output can be used as part of a storytelling experience and connected to elements beyond its own modules. Since animation pipelines are very malleable and project specific, this functionality is generally outside the scope of Puppitor' s core set of features (and outside the scope of this chapter), even if the system is intended to work in conjunction with animation systems.

Beyond being incorporated into external code bases and tools like Ren'Py [226], Puppitor is designed to be fully customizable within its action, modifier, affect framework. The fixed parts of the system are its general categories of actions and modifiers as well as the method used to update its emotional state. All the specifics of what actions and modifiers are used, the domain of the emotional state, and the relationship between actions and emotions are all completely customizable. All examples in this chapter use actions derived from Stanislavsky's Method of Physical Actions: open flow, closed flow, projected energy, and resting. Modifiers are derived from the Viewpoints: tempo up, tempo down, and neutral. The set of emotions are: joy, anger, sadness, worry, fear, and love. I want to stress that each of these sets, while heavily rooted in the acting practices inspiring Puppitor, are only one possible choice for using the system. As an embedded domain specific language, Puppitor supports arbitrary sets of actions, modifiers, and emotions.

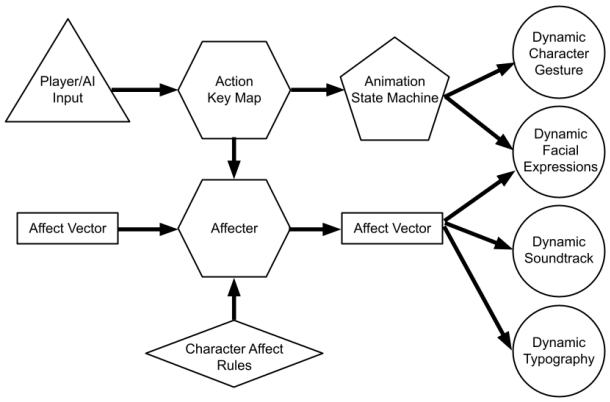

Figure 7.2: Puppitor translates inputs from a human or AI player (triangle) into expressive actions (circles) based on a specified ruleset (diamond) through its core modules (hexagons), which update the character's emotional state (rectangles) and serve as inputs to control character animations (pentagon).

7.4.1 Action Key Map

The Action Key Map module is where all of the direct input to the system (and by extension a character) is processed. Everything in the module revolves around the general categories of *actions* and *modifiers*, the exact details of which are specified as part of the Puppitor's domain specific language, an example of which can be found in the case study. *Actions* are gestures, anything that would dramatically change the way a character holds or carries themselves. *Modifiers* on the other hand are for smaller adjustments that change the feel of those larger actions. A designer specified default action and modifier gives Puppitor a way to return a character to a baseline state from any emotional state they may find themselves in. This organization by actions and modifiers is a part of each of the three elements the Action Key Map tracks: the mapping of keyboard keys to actions (key map), the tracking of key states and their associated actions (possible states), and the expressed action and modifier for use in

other Puppitor modules and outside of the library itself (prevailing states).

The key map element of this module associates Puppitor actions and modifiers with arbitrary sets of keys corresponding to inputs on a keyboard. Any number of keys may be associated with a single action or modifier with the exception of the default action and modifier, which never have inputs mapped to them as they are intended to be chosen when there is no input. As it relates to external usage, the key map is where the listeners connecting the entirety of Puppitor to the keyboard are created. The key map then allows for Puppitor's actions and modifiers to be integrated more directly into code handling player input, as the link between keyboard and meaning of the inputs is handled internally rather than requiring additional setup. For example the Stanislavsky and Viewpoints based actions and modifiers can be mapped as:

```
open flow : N
closed flow : M
projected energy : B
resting : None
tempo up : C
tempo down : Z
neutral: None
```

The possible states track the key states of the associated non-default actions and modifiers, allowing multiple keys to be held down simultaneously. To continue the example from the discussion of the key map, if both the N and M keys are pressed, open flow and closed flow will be flagged as active states.

Puppitor requires that a single action and modifier be performed at any given time which is where its final element, prevailing states, comes in. This element takes the information provided by the possible states and determines which action and modifier

to select as prevailing. Importantly, Puppitor itself does not determine the priority of each action and modifier, as the prevailing states' update behavior sets a specified action or modifier to active and makes every other action or modifier to inactive. How this update behavior is integrated into external programs is up to the developer using it. For example, when both open flow and closed flow are active according to possible states, if prevailing states is first told to set open flow to active and then is told to set closed flow to active, only closed flow will be active.

7.4.2 Affecter

The Affecter is the core of Puppitor' s emotional expression capabilities. The module is built around the idea of characters having rules that connect the actions they perform (as stored in the Action Key Map) to particular expressions of emotions. A character' s emotional state is represented by a mapping of emotional affects to their corresponding floating point values (the Affect Vector). Rulesets are made up of series of entries in a JSON file that define the update value each action applies to a character' s emotional state, the multiplier each modifier applies to those update values, the adjacency and edge weights of other emotions, and the equilibrium point for the emotion to return to when the default action is performed. The rules are then loaded into the Affecter itself and applied to the Affect Vector each update cycle based on the action and modifier specified by the Action Key Map.

For example, if the affects joy, anger, sadness, worry, fear, and love are a Puppitor domain' s set of emotions, each will have a corresponding entry in each character'

s affect vector and ruleset. The values in the Affect Vector default to the equilibrium point specified in a ruleset, so an initial Affect Vector could look like:

```
joy : 0.35
anger : 0.1
sadness : 0.54
worry : 0.77
fear : 0.54
love : 0.28
```

When updating an Affect Vector, the update value of an affect is multiplied by the modifier value, then added to the corresponding affect entry before the value in the Affect Vector is clamped between a designer specified floor and ceiling value. If no buttons are held, the default action and modifier will be performed and move affect values towards their corresponding equilibrium value rather than the floor or ceiling value. Without clamping, each affects values could grow or shrink to the point it would be nearly impossible for characters to express certain emotions. Continuing the example using closed flow as the active action, the update values in joy, anger, sadness, worry, fear, and love's affect rules corresponding to each affect's closed flow value would be multiplied by 1.0 (as no modifier is being applied) then added to the Affect Vector's values. Using the below affect rule entry fragment as an example, the joy value in the above example Affect Vector would be 0.3495.

```
"joy" : {
    "actions" : {
        "resting" : -0.0003,
        "open_flow" : 0.00004,
        "closed_flow" : -0.0005,
        "projected_energy" : 0.0005
    },
    "modifiers" : {
```

```
            "tempo_up" : 1.14,
            "tempo_down" : 0.5,
            "neutral" : 1.0
        },
        "adjacent_affects" : {
            "love" : 90,
            "worry" : 10
        },
        "equilibrium_point" : 0.35
    },
```

The adjacency lists in a ruleset specify a directed, weighted graph of a character's associations between emotional affects, where higher weights are stronger associations. These lists take the form of percentage values mapped to their corresponding emotional affect. Within Puppitor, the adjacency lists play a small role as part of the system's provided methods for picking the highest valued affect in a character's Affect Vector. As multiple affects can potentially reach the ceiling value, the currently expressed affect is prioritized. If the currently expressed affect is no longer one of the highest valued affects, one of its adjacent affects is randomly selected based on its weight value (a weight of zero is ignored, if all adjacencies are zero, then an unweighted selection is performed). If there are no adjacencies available to pick, a simple random choice of any of the highest valued affects is made. This selection process allows the emotional state represented in the affect vector to be converted to a discrete value, a character' s prevailing affect, for use outside of Puppitor, should a single value be preferable to the entire emotional state.

Where Puppitor's adjacency lists provide much more power and expressivity is when an AI system is puppeteering a character, as they can be used to allow an AI

character to interpret another character's expression of emotion (see the case study for a more in depth explanation). This interpretation can be accomplished by using Puppitor's adjacency lists to set the goal of an AI algorithm (for example greedy search) to be the AI's character's associated affect based on what a different character is expressing. Using the updated Affect Vector above as an example for a player character, worry is expressed as it has the highest value at 0.77. To decide how to respond, the AI character looks at the adjacency list in the worry entry of its ruleset and sees it can either respond by performing sadness or fear, each with a weight of zero, so a random choice will be made (if the selection follows the same logic as Puppitor's internals).

7.5 Expression with Puppitor: a Case Study of *Tracks in Snow*

Puppitor' s primary outputs are a character' s action, modifier, and prevailing affect. In *Tracks in Snow* [124, 128], a visual novel about two women fleeing their home during Passover and designed in conjunction with Puppitor, the system' s actions and modifiers are used to allow characters to dynamically change their poses and prevailing affects are used to change characters' facial expressions. Each character' s prevailing affect also changes the typography of their lines when displayed. Finally, each character has a corresponding instrument in the musical score which uses their prevailing affect to dynamically switch between tracks to further emphasize their expression of emotion.

Figure 7.3: Chiara (left) tells Rika (right) that she'll give her space if she wants. The tones of each version of the scene pictured, angry Chiara and joyful Rika above and afraid Chiara and angry Rika below, are drastically different in the moment as well as how each scene was played before to create these two moments in *Tracks in Snow*.

7.5.1 Character Through Actions and Modifiers

The two characters in *Tracks in Snow*'s cast were designed to contrast with each other narratively, visually, and most importantly systemically using Puppitor to support and emphasize the other two aspects. Rika and Chiara share some of the complexity of fighting game characters as part of their expressiveness. Fighting game characters feel different to play based on the kinds of inputs they use as well as how their animations and frame data all connect to create an expressive character [58]. Puppitor allows for a similar degree of characterization when combining character poses, facial expressions, all linked together using the system. Both Rika and Chiara use the same buttons on the keyboard to gesture when being played by a human, though which parts of the interface they emphasize is as much a part of their personalities as their hairstyles, poses, or ways of speaking.

Rika is a more flamboyant and mercurial character who can access most of her emotional range by changing her gestures without needing to rely on modifying any of those gestures' tempos. Her ruleset is mainly focused on using larger actions to quickly switch between expressing different emotional affects, as each action has a strong positive connection to two affects and a strong negative connection to at least two other affects. In this case briefly performing an intermediate action to change Rika's emotional state is the most reliable way of moving her towards expressing a desired affect. For example, if a player wants Rika to express anger after she has had her feet up performing open flow for some time (as in figure 7.3), first having her perform closed

flow will reduce her expression of joy and love, as both of these affects are positively associated with the open flow action and negatively associated with the closed flow action. A few moments of being closed can then move into having her perform the projected energy action, which increases both anger and love. Since Rika performed closed flow, love's magnitude in her Affect Vector has been reduced, allowing anger to then be expressed as desired.

Chiara on the other hand is a more subdued character with a very wide worry streak who constantly has to actively overcome that worry to express other emotions. To emphasize this anxious part of her personality using Puppitor, as well as to provide a very different tactile experience from playing Rika, Chiara almost always must modify her actions to access the rest of her emotional range. Rather than having two affects strongly associated with each action by default, every unmodified action primarily increases worry. Her modifiers then are designed to be both the only way of shoving her worry aside by having each modifier and action combination correspond to a strong expression of a single affect as well as any modified action reducing worry's overall value using a negative multiplier. For example, for Chiara to even get angry (as she is in figure 7.3), she not only needs to perform the projected energy action, she must specifically modify it with tempo down. If she does not she will simply worry more. If she instead uses the tempo up modifier, she will become joyful as joy is the affect with the highest modifier multiplier and action value using the combination of tempo up and projected energy.

When setting up either Rika or Chiara to be performed by a human player, first

their actions: *open flow*, *closed flow*, and *projected energy* are mapped to the keyboard keys of: 'N', 'M', and 'B' respectively in the key map dictionary of Action Key Map, allowing those keys to be listened for in Ren'Py's event handler. As part of the same process, their modifiers: *tempo up* and *tempo down* are mapped to the keys: 'C' and 'Z' respectively. The default action of *resting* and default modifier of *neutral* are passed into the Action Key Map. Now when any of those keys are pressed, their corresponding action in the possible states dictionary will be set to true. In the event no key is pressed, *resting* and *neutral* will be true as the respective default action and modifier. Finally, the possible states are used to determine which action and modifier are expressed in the prevailing states dictionary. The exact priority that prevailing states uses is determined as part of connecting Puppitor to other system's update functionality. In *Tracks in Snow*'s case, this priority, from highest to lowest, is: *open flow*, *closed flow*, and finally *projected energy*.

7.5.2 Interpretation Using Affect Adjacencies

The AI actor in *Tracks in Snow* uses a greedy search to evaluate each of the action and modifier pairs it could perform each frame and chooses the combination that will add the highest value to the affect it is given the goal of performing. By itself the search has very little ability to reason about what is put in front of it, which Puppitor's features can address. Each affect in a Puppitor ruleset can be given adjacencies to other affects within the domain, ultimately building a weighted, directed graph of how emotional affects relate to each other. In other words, the the associations a character

has about each affect.

In *Tracks in Snow*, the AI actor uses the adjacency lists as primary method of allowing interpretation of what the human controlled character is doing. When the human player performs any affect, the AI receives this and then chooses an affect it wants to perform based on a weighted random selection of an affect out of the adjacencies to that received affect in its ruleset. A single adjacency will make for a fairly calm performance from the AI using this approach as it only has one association. By adding more associations, the AI will commit less to trying to perform a single affect. When used in conjunction with the weight values, the feel of these changes can create different performances. To return to the example in figure 7.3, if the AI is performing as Chiara and Rika just started expressing joy, Chiara's adjacencies for joy are:

```
love : 90
worry : 10
```

Approximately ninety percent of the time the AI will try to perform to express love back to Rika. The remaining approximately ten percent of the time the AI will try to perform worry back. The time it takes for emotions to change means the AI will likely never be able to actually express worry with this weighting, but it will make for a mostly stable performance with the occasional adjustment as the quick attempts to express worry pop up. With a more balanced weighting, such as:

```
love : 60
worry : 40
```

the performance will get more fidgety even with the weighting still favoring love, worry may creep up every so often and be fully expressed.

Puppitor's adjacency lists provide a way for AI characters to interpret other characters' performances through their own associations with each emotional affect as well as provide an additional way to tune an AI's performance within the domain specific language itself. By design, most of Puppitor's language focuses on the way a character expresses themselves, but with the adjacency lists, there is a built in way for characters to make choices about how they want to respond to other characters' performances as well.

Through its implementation in *Tracks in Snow*, Puppitor enables dynamic character expression either by a human or AI player. These dynamic poses and facial expressions help shape the tone of a scene alongside other sources of feedback like a reactive musical score and typography. All of this feedback is realtime and continuous. As a result of this layered approach, shadows of previous emotions can remain as scenes progress and flashes of others can surface as characters change what they are trying to express. Being a linear narrative using Puppitor, *Tracks in Snow* has an emphasis on interpretation and reflection as described by Junius et al [129], though the loop of acting and reflecting is much more dependent on a player's desire to than built directly into the structure of the game. One significant area of improvement is enabling the AI characters to further reason about Puppitor's state to make more interesting acting choices.

7.6 Conclusion

Puppitor is a system for character expression of emotion built as a computational caricature of theater acting practices [29, 158, 223], and allows players to control the characters' physicality in a narrative space similarly to how they might control fighting game characters in combat. Specifically, Puppitor maps actions and modifiers to keyboard inputs and based on this mapping applies rules about how emotions are expressed, all specified using an embedded domain specific language. Puppitor facilitates a wide range of expression for both human and AI players to perform characters, which I demonstrate through the case study of *Tracks in Snow*. Puppitor provides means of systematically exploring dramatic play and stage action, which the field of interactive drama has left relatively under explored in favor of plot action. Through emphasizing dramatic play, Puppitor allows us to expand the study of AI character design and interaction paradigms for interactive storytelling experiences, and more broadly AI characters as they are utilized in interactive narrative and beyond. With Puppitor I hope to both inspire further investigation of systems for supporting dramatic play as well as providing a new domain for work focused on character and agent behavior.

Chapter 8

Exposing AI Decision Making with Expressive Response Curves

The highly dynamic nature of *Tracks in Snow* scenes meant that understanding why the AI chose any given action was at least partially guess work. Comparing behavior to the raw rules definitions in JSON explains some acting decisions, but is insufficient to paint a whole picture. What I describe in this chapter is as much enabling the evaluation of Puppitor rulesets as it is a tool for understanding AI search's relationship to a domain.

Deciding how to evaluate Puppitor was one of the things I wrestled with most for this dissertation. What would be helpful to me? What even makes sense for the kinds of questions I'm interested in? What even are the questions I'm interested in? I ultimately landed on what became automated playtesting for expressive characters rather than generated levels or artifacts.

The work I discuss in this chapter, edited from the original Expressive Response Curve paper [125], grew out my realization that as simple as Puppitor rules look in a JSON file, understanding the spaces and dynamics they create is extremely difficult without support[1]. Until I started working on the tools that would become the final utilities now bundled in Puppitor's public GitHub repository, the only option for iterating on a character ruleset was playing *Tracks in Snow* and looking at raw console logs. While better than nothing, there are some fairly extreme limits to what one could observe just by playing a game and looking at state printouts.

This chapter begins with a description of Expressive Response Curves and their relationship to existing work in quantitative evaluation. Next, I describe the method used to create and analyze Expressive Response Curves. I then discuss the full process of applying the methodology to Puppitor rulesets and the fruits of the analytical work.

8.1 Introduction to Expressive Response Curves

Designing expressive character behavior for AI performance is a difficult undertaking thanks to the complexity of the possibility spaces they create. The scale and complexity of these expressive possibility spaces makes iterating on the designs of models and domains for these systems challenging due to the limited insight their definitions provide about the nature of the spaces themselves. A lack of available tools and tech-

[1]The exact tool I describe here in fact grew out of my C# port of Puppitor and the sudden need for making sure there was parity across library versions. While the utility itself has had its functionality expanded and its name changed to something more general, prfeu.py and the entire tools directory of Puppitor's Python repository owe their existence to Expressive Response Curves.

niques to assist in understanding expressive possibility spaces reinforces this problem. Furthermore, this lack of support leads to a lack of easily intuitable information about the kinds of play experiences these possibility spaces produce. As a result, designing the possibility spaces themselves and relating them to resulting player experiences, human or AI, is frequently done through human playtesting and observation which can leave large areas of these spaces unexplored and underutilized.

Automated playtesting for games has largely been focused on evaluating generated levels [107,159,213] and game balance [207,208,282] and emulating human decision making [63, 109]. The difficulty of gaining insights about AI systems, behaviors, and potential play experiences through purely human testing drove us towards developing methods of automated playtesting specifically for expressive models. With our goal in mind, I view core principles of automated playtesting as applicable to expressive models, helping us expose information about the expressive possibility spaces created by these models in more intuitive ways than currently exist. Interest in pursuing more player experienced approaches to automated playtesting is increasing, as recent work in the field illustrates [18]. Through automated playtesting I can gain a more complete picture of expressive possibility spaces than traditional playtesting approaches would reveal. In turn, enabling more deliberate design iteration and better understanding the implications changes to these spaces have on the player experience.

In this paper I propose Expressive Response Curves (ERC) as the base unit of our approach to automated playtesting. An ERC represents a sequence of character actions and states, in an expressive possibility space. Individual ERCs are then combined

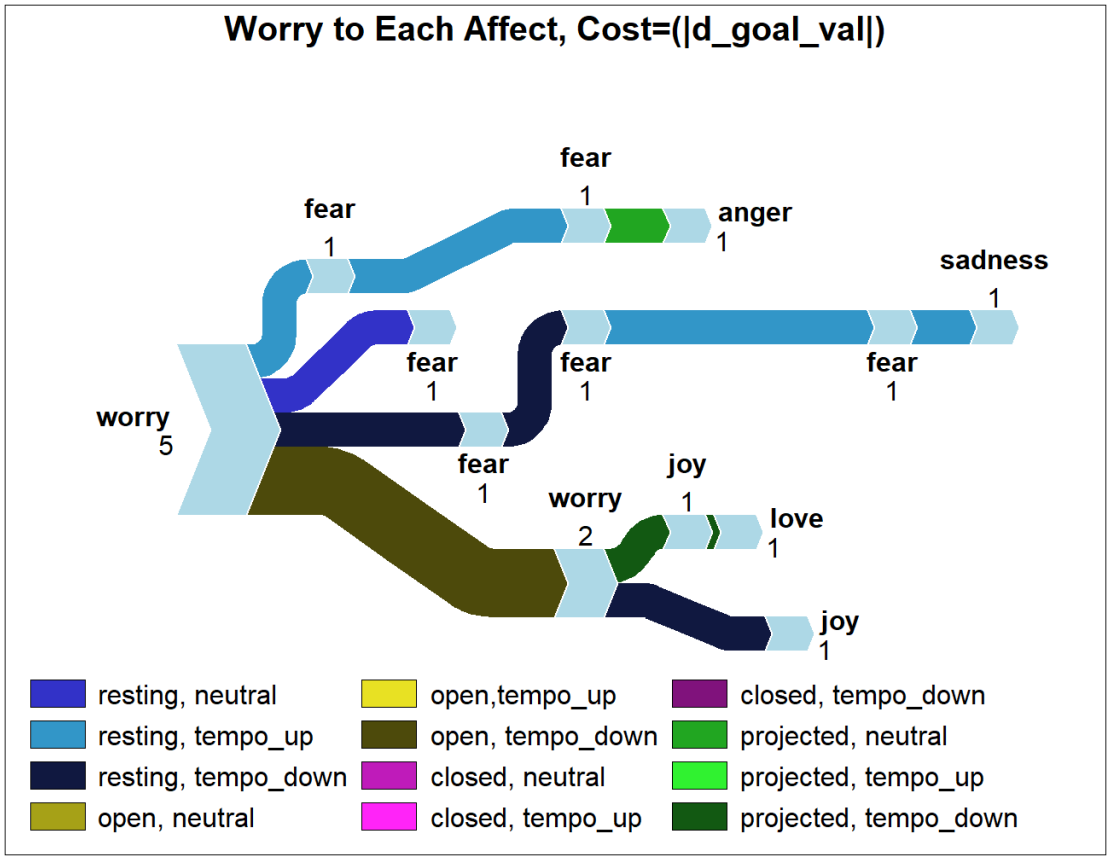

Figure 8.1: A collection of ERCs, creating an ERC map, where each flow path represents a single ERC and path from expressing worry to a different affect in the test domain of Puppitor. The same moves can be used as part of a sequence of actions to express different affects in Puppitor.

to create maps of the possibility spaces defined by AI models of expression. In doing so, *ERC maps* can surface information about how and when interaction and feedback is presented to a human player as they interact with the system or express themselves using the system. This mapping, therefore, allows us to analyze and ultimately characterize particular expressive possibility spaces with regards to the player experiences they afford. The focus of this paper is on the application of ERCs to the expressive possibility spaces created by a model of character expression, but ERCs applicable to expressive possibility spaces created from other sources like emergent narrative systems [147, 205].

Furthermore, as a sequence of potential actions and timings in relation to state, ERCs can surface information about moment-to-moment player actions and feedback. This allows us to expose how the player interacts system at the level of character responsiveness to their actions. In this sense, therefore, ERCs allow us to visualize what is the equivalent of Game Feel [258] for expressive character AI. This moves us closer to being able to directly study the connection between an expressive AI system's functionality and player experience and perception, and the dynamics between them.

I describe our approach to creating and using ERCs and present a case study of its application to Puppitor [126], an embedded domain specific language that defines the expressive qualities of a character to allow human and AI players to collaboratively perform scenes within narrative experiences. I chose Puppitor for this case study as it is a lightweight and flexible framework that suits our purpose of creating a complex expressive space in a human authorable format. Additionally Puppitor is a freely available, publicly accessible software library which has been used as part of a playable

experience, from which I extracted rulesets to conduct our case study. Finally, I discuss the results of the case study and the implications ERCs have for the development of AI characters and performance.

ERC is not a recommendation tool or an approach for providing explicit suggestions to designers about the spaces they create. Rather, it is a method to allow them to make more informed decisions about the possibility spaces they are creating and how their designs could be perceived by a player.

8.2 Related Work

Our approach to creating ERCs is rooted in using concepts from game feel, namely response envelopes [258] and juice [83] as an alternative foundation to automated playtesting. The analytical aspect of ERCs draws inspiration from expressive range analysis [245] as an existing approach to characterizing the possibility spaces created by generative systems. Finally, I chose to use A* [100] as part of the approach to creating ERCs over other popular algorithms like Monte Carlo Tree Search [47] due to its more consistent and reproducible behavior. Additionally, A* has built in path generating functionality which is important for the creation of ERCs, as they are paths through expressive possibility space, and their analysis to characterize the expressive possibility space's affordances for player experience as inspired by game feel.

Automated playtesting has grown from the desire to have rapid evaluation of generated content such as video game levels [107, 159, 213], game balance [207, 208, 282],

and skill progression [110]. When used for the evaluation of generated levels, personas capturing human playstles are frequently used [7, 159]. Monte Carlo Tree Search (MCTS), a best-first search algorithm developed for general gameplaying [46, 47], is commonly used to drive automated playtesters [107, 110, 135, 196, 282] as it requires no domain knowledge to be integrated into the algorithm [119]. Even with this benefit, MCTS does require some adaptation for the purpose of human-like playtesting. While the algorithm's high-level reasoning of trying to choose the best available move by predicting the outcomes does resemble parts of human decision-making processes [107], the commonly used selection formula, Upper Confidence Bound for Trees (UCT), frequently leads to inhuman choices for game moves [63]. Additionally, MCTS's nature as an aggregator of random actions makes its exact sequences of actions difficult to consistently reproduce, meaning for our purposes of creating ERCs, it is unsuited to our goal of finding reproducible paths through expressive possibility space.

Expressive range analysis (ERA) is a method of describing the possibility space of a generator, often a level generator, based on the artifacts it produces [245]. The existence of ERA creates the potential for analysis of other possibility spaces, including the expressive ones I describe in this paper or the results of automated playtesting [5]. ERA has become one of the foundations for qualitative exploration and understanding the range of artifacts produced by a generator [252]. The core of this approach to analysis is the selection of metrics to describe a level, each corresponding to an axis, to allow the for the visualization of the metric scores of a set of generated artifacts [245]. ERA has been incorporated into design support tools like Danesh [57] which will

automatically analyze a set of generated artifacts as part of its goals for exposing the contours of generators.

With our goal of exposing the responsiveness of expressive systems to designers through automated playtesting, I use game feel [258] as the foundation of our method's understanding of the effects granular, low-level interactions have on player experience. Game feel describes the way interacting with a game is built out of numerous pieces coming together, from the ways an individual button press manifests on screen and through speakers to the ways giving a player goals alters their relationship to individual tasks [258]. The two most relevant aspects of game feel to the discussion of physical character expression are response metrics and low-level rules. When describing the relationship between input and feedback, Swink identifies the stages of attack, sustain, and release (ASR) as an approach to understanding the way a game interprets and responds to input [258]. Swink discusses ASR almost exclusively in relation to the ways button presses correspond to the responses of an on screen character. Low-level rules in turn define the physical relationships between entities, such as why larger enemies tend to take more damage to kill or bigger, heavier moves in fighting games tend to do more damage than lighter, faster ones [258].

Additionally, juice is a subset of game feel focused on cascading audiovisual feedback in response to player actions and input [83] and our usage of ERCs can provide insight about where juice may ideally be applied to heighten the play experience. With that said, the specifics of juicy design are highly dependent on the type of experience being developed or discussed [103]. Interestingly, with as much focus as juice has received

as part of game feel and feedback mechanisms, it does not necessarily improve player performance in games and too much or ill considered incorporation of juicy elements can slightly detract from their use as feedback about performance [104]. While a discussion of juice as a component of interacting with AI controlled characters is beyond the scope of this work, ERC can help guide designers to places where incorporating such elements would be beneficial to the player experience.

I chose to use A* as the technical foundation for creating ERCs because it is a heuristic graph search and pathfinding algorithm. A* uses a breadth-first approach to visiting graph nodes and a priority queue based on cost and heuristic values to guide the search towards a specific goal [100]. Unlike breadth-first search, A* is not guaranteed to visit every node in a graph, nor is it guaranteed to find an optimal path without an admissible heuristic, a heuristic function which never over-estimates the distance to the goal [100]. The algorithm searches a given graph from a chosen starting point, towards a specified goal, using a heuristic to capture domain knowledge about the graph's domain to help prioritize promising nodes. From the starting node, each adjacent node is processed and added to a priority queue if it has not already been visited or the cost to reach the node is less than the previous visit. The node's priority is then determined by the cost to reach the node and the heuristic's estimated distance to the goal from the node. The highest priority node is then taken from the queue and the process is repeated until the goal is found. A* has been used in game related domains like path planning [105], action planning [202], and general game playing [119] thanks to its nature as a heuristic search allowing for flexibility across domains. In addition, the

paths A* finds are more easily reproducible than other algorithms commonly used in game AI like MCTS. Both of these features allow ERCs to have a common technical basis which can be adapted to domains beyond those I consider explicitly in this paper.

8.3 Methodology

The method of creating and analyzing ERCs is focused on characterizing a possibility space through repeated searches to build a map of the locations in a sequence of actions and state changes that have noticeable and human perceptible feedback. Combining individual ERCs into a map of expressive possibility space exposes the complexity and responsiveness of actions required for a player to move between different expressive states described by the possibility space. Points in these possibility spaces are visited in sequence, with the path between them connecting them temporally to each other. Any location where a human perceptible change occurs in one of these sequences is then highlighted.

The steps for building and analyzing ERCs is as follows:

- **Translating the domain to A*:** Modify the chosen expressive possibility space and turn it into a graph domain to allow A* search to be used.
- **Adapting A* to the domain:** A* is a heuristic search and therefore has its cost and heuristic functions informed by the specifics of the domain to create ERCs.
- **Running searches:** Starting points for A* searches are chosen and search goals are defined in line with the domain knowledge representation to generate paths

for the creation and analysis of ERCs.

- **Analyzing paths:** The paths found by A* are analyzed for their length, complexity, and responsiveness to player input and compared with each other to characterize the possibility space.

8.3.1 Translating the Domain to A*

To allow A* to operate over an expressive possibility space domain, its features must be translated into a graph format that the search algorithm can understand. This translation process focuses on identifying the elements of a given expressive possibility space that correspond to nodes and edges of a graph. A node defined by whichever domain elements describe a point in the possibility space. They contain a particular state representation, such as the relative values of each available expression and which expression is performed based on those values. An edge is made up of the elements of a possibility space which facilitate moving between the points it contains. These are represented as actions which have an effect on the state representation contained within a node, such as raising or decreasing the relative values of some of the expressions in a node. The exact details of this translation process are dependent on the way a given possibility space is defined. Which elements of a domain correspond to each graph element may not be obvious if concepts like state or state changes are not defined. I give a detailed example of the creation and application of ERCs in The case study sections.

8.3.2 Adapting A* to the Domain

Once a graph form of the possibility space is created, A* must be adapted to work with this new domain. This adaptation process focuses on defining the node representation, calculating edge costs, designing heuristic functions, and finally identifying abstractions necessary to the search, with the ultimate goal being the creation of an A* search which outputs useful and effective ERCs.

The representation of nodes able to produce ERCs requires that human perceivable changes to state be made primary components of a node. The translation of an expressive possibility space into a graph will not necessarily expose these state elements in an obvious way. For the purpose of creating ERCs, nodes must also highlight the components of the state which are feedback for the player. By highlighting these elements and making them primary components of the nodes, the sequences of actions found by A* search expose the perceived responsiveness of an expressive possibility space.

Edge costs are required to guide A* searches towards their goal and as a result can have noticeable effects on which sequences of actions A* chooses to perform, potentially making for more variance between individual ERCs. For this reason, costs are calculated during the search rather than being solely a fixed property of the graph definition. This decision allows the behavior of A* to be directed to prioritize different features of the domain and more effectively explore different permutations of action sequences. As a result the cost calculation must factor in the complexity and/or desir-

ability of available actions. Different cost calculations then can be used to produce a wider range of individual ERCs to describe qualitatively distinct paths through parts of a domain's expressive possibility space, allowing for more comprehensive ERC maps to be created.

Creating a heuristic for A* in the domain expressive possibility space requires an understanding of the goal's representation, as a valid goal for the creation of an ERC can be as general as a component of feedback being exposed or as specific as an exact state representation. Because ERCs are focused on the human perceivable elements of expressive state, the goal of a search should reflect this priority. In turn, the heuristic function must have its calculations based on the state components responsible for tracking player feedback. As a result, when designing a heuristic function to produce ERCs, the distance estimate must include some understanding of player visible feedback as defined in the domain.

Finally, the complexity and scale of expressive possibility spaces can lead to searches becoming bogged down taking huge numbers of repeated actions to make minute changes to the state. To ease this calculation burden, any area of the search featuring long sequences of repeated action can be abstracted into single actions with cost and state changes to reflect the distance they cover, and to allow for much faster creation of individual ERCs.

8.3.3 Running Searches

Running searches for the purpose generating ERCs prioritizes finding starting points with distinct human perceivable differences in state. The goal of an individual search is another point in the expressive possibility space with perceivable differences to the starting point. Effective searches from a singular starting point cover all possible goals, whether or not a particular goal is in fact achievable from the chosen starting point.

8.3.4 Analyzing Paths

The process of creating a map of ERCs begins with choosing a starting point and an expressive goal to be found. The sequence of actions to reach this goal is then created using the search as described previously. Next, any point in the sequence of actions where a new action is taken or shows feedback visible to a player is highlighted. These steps are repeated for the same starting point for each possible expressive goal: creating a new ERC and highlighting the points of change in the found path. The individual ERCs using the same starting point are then combined to make a map of the expressive space between that starting point and all possible expressive goals in the possibility space.

The reason for highlighting changes in state, both player and system controlled, stems from Swink's response envelopes, which expose the relationship between a human player's input and the game responding with its own feedback (movement, animation, etc.) as a function of the time it takes for the game to complete a state transition. An

ERC can then show the responsiveness of an area of a domain by exposing the length of time it takes for the state to respond to player input. This measurement of time can be as fine grained as the number of frames drawn to the screen, with the higher the frame count (or any other time measurement) between a player action and feedback from the state corresponding to lower responsiveness to the action taken.

Path length itself can be a measure of responsiveness and complexity. A lengthy path that contains long sequences of perceptually similar nodes, especially if their edges are distinct, is an indicator of an unresponsive area of the expressive possibility space, the expressive equivalent of the 1000 bowls of oatmeal problem [55, 215]. Whereas a long path with many perceptually distinct nodes with multiple different kinds of edges connecting them is an indicator of complexity when trying to reach a particular part of expressive space from a given starting point.

8.4 Case Study: Puppitor Rulesets

Creating ERCs to analyze the expressive possibility space created by Puppitor character rulesets required us to translate Puppitor into a more easily searchable domain. Our chosen rules are extracted from *Tracks in Snow* (*TiS*) [124], a visual novel designed to showcase Puppitor [126]. This process focused on identifying which elements of the domain are relevant to state representation and the actions which allow movement between expressive states, then converting them into a graph structure for use with A*. For this reason I begin this section with a summary of Puppitor as an expressive

domain, then discuss our translation choices, approach to applying domain knowledge to A*, and reasoning behind the creation of particular sets of paths through Puppitor's possibility space.

8.4.1 Puppitor

Puppitor is an embedded domain specific language that defines the expressive qualities of a character to allow human and AI players to collaboratively perform scenes within narrative experiences [126]. The system creates a framework to map a character's physical actions to expressions of emotional affect. These physical actions are defined as *actions* and *modifiers*, with only one of each respective category being allowed to be performed at any time (the *prevailing* action and modifier). These actions and modifiers can be mapped to buttons on a keyboard to facilitate human input:

```
actions:
    open_flow : N
    closed_flow : M
    projected_energy : B
    resting : None
modifiers:
    tempo_up : C
    tempo_down : Z
    neutral: None
```

In the above example, the action and modifier, taken from one of the *TiS* rule-sets, when no buttons are pressed are resting and neutral respectively. When only the B key is pressed, the action and modifier are projected energy and neutral respectively.

Puppitor uses these actions and modifiers to map actions to expressions of emotional affect. An affect vector contains the set of affects which a character can

expressed mapped to their relative values. Only one affect can be expressed at a time, corresponding to the affect with the highest magnitude, the *prevailing affect*, with logic to handle the possibility of multiple affects having the same value [126].

```
joy : 0.35
anger : 0.1
sadness : 0.54
worry : 0.77
fear : 0.54
love : 0.28
```

In the above example, the prevailing affect is worry. Each affect listed in an affect vector must have a corresponding entry in a character's rule file which defines the values to be added to the affect vector during an update. When a Puppitor update is applied to an affect vector in a game loop, the prevailing action's value is multiplied by the prevailing modifier's value [126]. For example, if we take the entry for joy pictured below and apply perform the projected energy action and neutral modifier with the above affect vector, joy will then have the value of `0.0005 * 1.0` added, resulting in a final value of 0.3505.

```
"joy" : {
    "actions" : {
        "resting" : -0.0003,
        "open_flow" : 0.00004,
        "closed_flow" : -0.0005,
        "projected_energy" : 0.0005
    },
    "modifiers" : {
        "tempo_up" : 1.14,
        "tempo_down" : 0.5,
        "neutral" : 1.0
    },
    "adjacent_affects" : {
        "love" : 90,
```

```
            "worry" : 10
        },
        "equilibrium_point" : 0.35
    },
```

8.4.2 Translating Puppitor into a Graph Domain

When looking at Puppitor's unmodified domain, I identify the affect vector and the actions and modifiers as the most important components of the system's definition of its expressive possibility space. I choose to translate affect vectors into graph nodes as each affect vector containing different values represents a unique point in Puppitor's expressive space. Actions and modifiers are what allow movement between affect vectors and more generally are what constrain the reachability of specific points in the possibility space. For my translation, I define an edge between to nodes of affect vectors as *moves*, the combination of Puppitor actions and modifiers.

With affect vectors being nodes, each node's outgoing edges correspond to every possible Puppitor move. There are four actions and three modifiers, making for a total of twelve moves and therefore twelve edges. Additionally, Puppitor allows any move to be performed even if it does not change the affect vector, so every node will have twelve outgoing edges. In turn, this means every node has at least twelve incoming edges, as nodes with minimized and maximized values in their affect vectors are reachable from huge numbers of other nodes.

With this graph representation of Puppitor, with affect vectors as nodes and moves as edges, I can begin to apply domain knowledge to A*. The need for additional

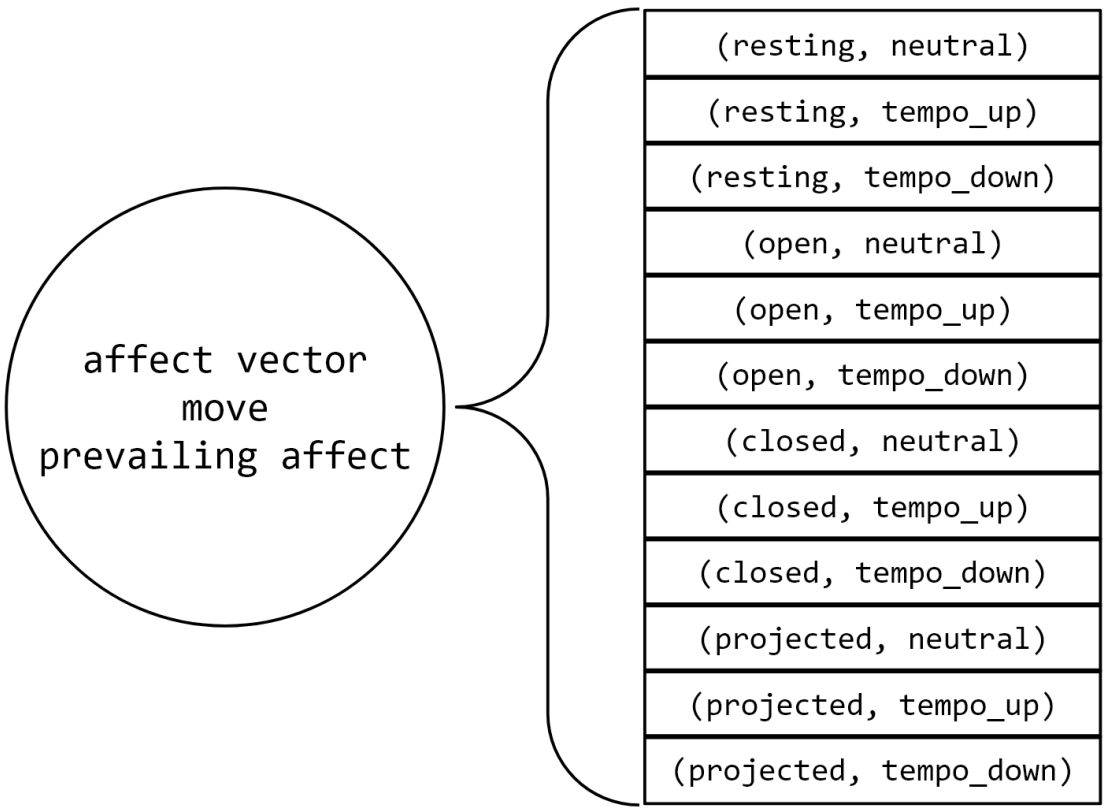

Figure 8.2: This diagram shows the contents of a single Puppitor graph node, the circle, and each outgoing edge, the rectangles.

information to guide A* through Puppitor's rule space and ultimately facilitate the analysis of paths means the graph definition of affect vectors as nodes and moves as edges is a starting point for the process of creating ERCs of the system's rulesets.

8.4.3 Adapting A* Search to Puppitor

Adapting A* search for use with the graph derived from Puppitor focuses on four areas: the information in a node, the cost function, the heuristic, and the optimization needed to traverse a large, continuous space. The goal of each step of this process is to enable A* to output paths with the information required to produce ERCs.

An individual affect vector does not contain enough information for A* to effectively search between nodes and also does not easily expose the human perceivable parts of itself on its own. To enable A* search and the creation of ERCs, I must add the additional information of: the affect vector's prevailing affect and the move which led to the particular affect vector. Without the prevailing affect stored as part of the node, affect vectors which contain multiple values of the greatest magnitude become ambiguous about which affect a player sees when they reach the node. Additionally, multiple affect vectors may share identical values but none the less be perceptually distinct due to which moves were made to reach them. As a result, I also store the chosen move in the node. An example of the full node as represented to A* search would be:

```
affect vector: {
                joy : 0.3505,
                anger : 0.1,
                sadness : 0.54,
```

```
                worry : 0.77,
                fear : 0.54,
                love : 0.28
               },
action: projected_energy,
modifier: neutral,
prevailing affect: worry
```

Due to the complexity of the possibility space created by a Puppitor ruleset, having access to multiple edge cost functions can aid our goal of mapping as much of the space as possible. The first cost function I use is: $|\Delta goal\_val|$, which calculates edge cost as the magnitude of the change in the value of the goal affect as determined by the move performed. Using the example node, if our goal is to express joy, our cost of reaching this node would be 0.0005 as the move that got us to this specific node was `(projected_energy, neutral)` which has an update value of 0.0005 according to the chosen ruleset. The costs of edges, when summed together, enable us to calculate the distance from the starting node and compare the relative difficulty of reaching a particular node. In this case, the total cost for determining the priority of the example node is 0.0005.

The first cost function used generally leads to longer paths as changes of greater magnitude cost more, so the search will tend towards the smallest steps possible towards its goal. A second cost function I developed is: $||\Delta max\_val| - |\Delta goal\_val||$. This formula for cost incentivizes A* to focus on moves that provide the greatest change in both the currently prevailing affect and the goal value.

To create a heuristic function for Puppitor's domain, the focus is on the parts

of the state preventing the goal affect from becoming the prevailing affect to produce an estimate of the distance to the search's goal. When pathfinding through expressive space, the priority is not reaching an exact affect vector, rather, our goal is to reach the nearest node where the prevailing affect is the goal affect. The following logic demonstrates this heuristic:

max_nodes = *max*(affect_vector)

max_value = *value*(max_nodes)

if goal *in* max_nodes *and* curr_affect ≠ goal **then**

heuristic_value = *big_num*

else

heuristic_value = *dist*(max_value, goal_value)

end if

return heuristic_value

The least desirable state is one where the goal affect is among the affects with the highest value but is not the prevailing affect. his is because any action taken that appears to increase the goal affect's value further will not in fact, which can cause the search to become stuck trying options which do not help it achieve its expressive goal. Therefore it is assigned an extremely low priority. Otherwise, estimate the distance to the goal as the difference between the highest valued affect and the goal affect. The heuristic value of this example node is calculated as the distance between worry (0.77) and joy (0.3505), resulting in a value of 0.4195. While a useful estimate, this heuristic is inadmissible as there is no guarantee of the value of the greatest magnitude remaining

constant as Puppitor allows rules to update every available affect value and our rules used in this case study follow this pattern. However, the optimality of a path is not the main concern here, only if one exists. It is beyond the scope of this work to create an admimssible heuristic for such a new and under-explored domain.

Our final addition to A* for use with Puppitor's graph representation is the "navmeshification" of the domain. To equate this abstraction process I use with the use of A* to pathfind across geometric spaces abstracted as navmeshes. Puppitor rules can be written with very small update values. This authoring pattern is useful at creating continuous experiences over time but when searching across expressive space, it leads to huge numbers of repeated updates and slow traversal.

When A* searches over a unabstracted Puppitor ruleset, it is the equivalent of using A* for pathfinding over the high-detail environment meshes of an area rather than the abstracted navmesh layer, i.e. moving between every tiny polygon of a cobblestone on a road. To abstract Puppitor updates, I consider each update to affect vector values a *step*. To avoid exploding the search space only to decide to perform the same action numerous steps in sequence, I can precalculate many steps in advance in the form of a multiplier.

With the example ruleset and affect vector, it would potentially take a minimum of 839 individual steps for the joy value to equal the worry value. Each step the search finds is fractionally small progress towards the goal and as a result many of the steps in the path will be performing the exact same move over and over, wasting time on calculations. If I instead pre-multiply a single step to instead be the equivalent of

90 steps, changing the update value to be 0.045 and one step from the initial value for joy to be 0.395, the minimum number of steps the search must calculate can be cut to approximately 10 steps.

8.4.4 Running Searches and Analyzing Paths through Expressive Space

For the search process, I picked six different starting affect vectors, each with a unique prevailing affect so the set would cover multiple parts of the possibility space with even a low number of trials. I then told A* to search for each affect in our domain from each of these starting points. At most five unique paths will be found, as the search simply continues to perform the same move if it is already expressing the goal affect. In addition I ran two additional trials using an alternative cost function developed during the process of this case study, the results of which can be found in figures 8.4 and 8.5.

The precalculation process requires us to reconstruct the full path and interpolate between the nodes A* discovers. Once A* finds a path between the starting point and the goal affect, nodes are added back to the node sequence to "unroll" the precalculations into the individual steps required to create a full path. The amount of reconstructed steps needed per node is based on the step multiplier originally used to precalculate the values. In our example, if the path A* creates is 10 steps, an additional 89 steps must be added between each of these nodes to account for the precalculation multiplier of 90 and interpolate between the initial value for joy of 0.35 and the next step's value of 0.395.

When reconstructing the path between steps, it is crucial to recalculate the

prevailing affect with each step added. This part of the process is necessary as Puppitor rules allow a single update to change every value in an affect vector, meaning these reconstructed steps can have different prevailing affects than either node they are interpolating between. The changes in prevailing affect contained in these interstitial steps is hugely important to the creation of ERCs because they provide a far more detailed view of the way a Puppitor ruleset responds to a move than the abstracted A* nodes alone are able to show.

Once a complete path is found, important points along the path can be highlighted, namely each step where the prevailing affect changes and each step where a new move is made. The highlighted path is now considered an ERC and can be used to characterize and analyze a Puppitor ruleset. The minute change in an affect vector's values are largely invisible until the prevailing affect changes. A new move being made changes the values being used to update the affect vector, and as a result, will alter the rates that each affect's relative strength change. Highlighting both of these steps allows us to analyze the responsiveness of a given ruleset, both for how quickly a desired affect can be expressed as well as the responsiveness of performing new moves. The shorter the number of steps between any of these changes, the more responsive the ruleset is to player input.

8.5 Case Study: ERC Map Analysis of a Puppitor Ruleset

The goal of our analysis of paths through expressive possibility space is simultaneously to characterize the human experience of specific expressive possibility spaces and to provide further insight into the ways an AI character understands and utilizes the space. The figures in this section all represent sets of individual possible paths through small corners of a single Puppitor ruleset. Each figure uses a single starting affect vector. Starting points that share a prevailing affect also share an affect vector. Each flow path shows the sequence of moves and affects to each other affect expressible from the starting point. These diagrams only represent individual paths to single possible ways of expressing affects and are not fully comprehensive captures of the entire expressive possibility space. Visualizing the full expressive possibility space of even a single Puppitor character is far too large and complicated an undertaking for the scope of this paper. Each node represents a change in prevailing affect. Each column of nodes represents a different point in time. Each flow path represents a unique move chosen. When multiple paths use the same move, the size of the corresponding flow is increased. The numerical values on each node correspond to the number of paths flowing into that unique node. Each terminal node represents the expressive goal of the A* search being reached.

For example in figure 8.1, there are three unique paths to expressing fear, each showing the expression of fear appear at a different point in time thanks to the different moves chosen. Additionally the two fear nodes sharing a time point in time are never

the less unique as they were reached using different actions. This distinction is further shown by the differing moves and expressions flowing from them. Alternatively, the same move was chosen to express worry on two different paths. From this same point of worry, different moves lead to unique paths to joy and love. Due to the incentives provided by the cost function in this example, A* does not choose the moves which will allow it to express its goal in the shortest possible time frame. This decision-making is most obvious where to express fear when fear is the goal, A* chooses the `(resting, neutral)` move which is a slower path than the `(resting, tempo_up)` move.

The nature of the precalculations used to optimize A* mean that interstitial prevailing affects are not seen unless they persist into the next node found. This is clear in the case of figure 8.1 where joy briefly prevails before love, the latter of which is the prevailing affect of the next node. Even if A* doesn't pay attention to these interstitial affects, they help give a better idea of the play experience around a particular point in the expressive possibility space.

Creating ERCs can also help find holes in the possibility space that may not be easy to intuit from rules. In the case of figure 8.3, the search did not find any path from this particular joy node to a node with the prevailing affect of fear. This particular ruleset was designed to make fear mostly only briefly prevail when moving between other affects, so its lack of presence is expected. In other cases, having ERCs available can help expose these gaps in the possibility space and allow for intentional decisions to be made about them earlier in the development process.

Using ERC to visualize the differences in searches, it becomes clear that the

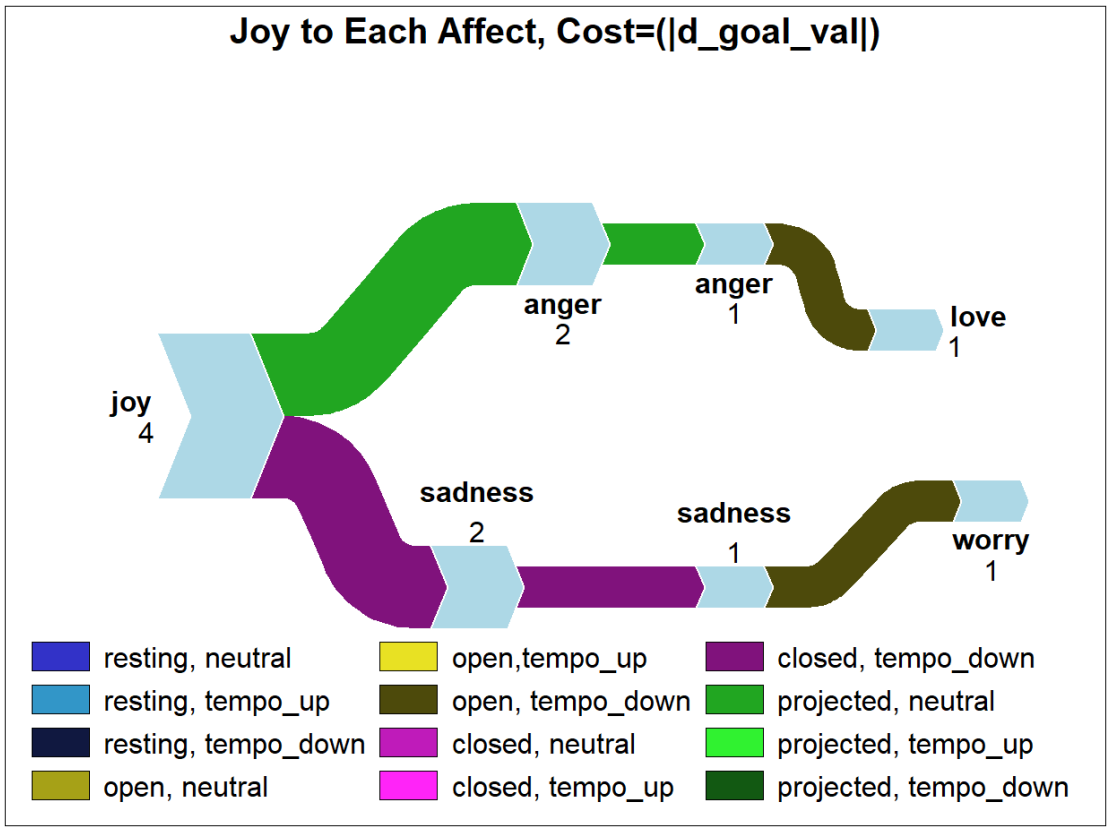

Figure 8.3: For some points in the expressive possibility space, affect values are such that moves can have very similar effects on multiple affects, hence why $(projected, tempo\_up)$ leads to both anger and love and $(closed, tempo\_down)$ leads to both sadness and worry. Also there is no path to expressing fear long enough for it to appear in a node visible to A*.

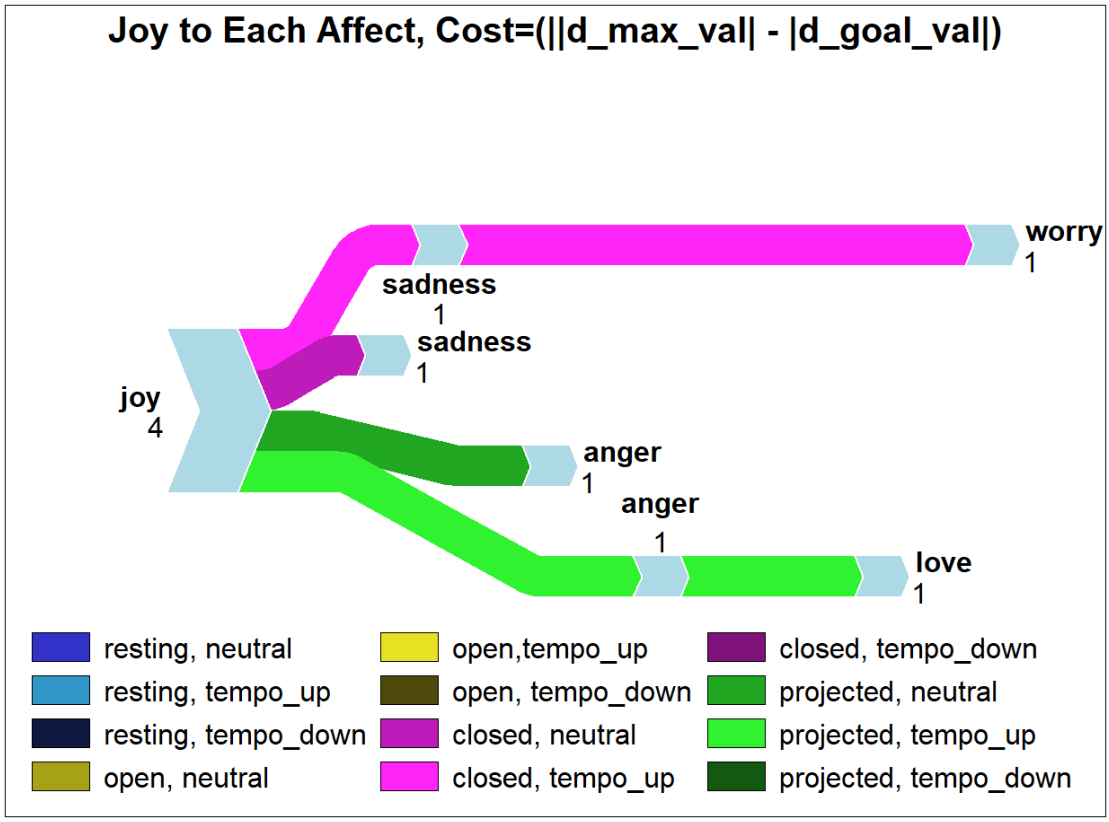

Figure 8.4: Altering the cost function leads to fully unique paths to each other expressible affect when starting at the same point of worry in the possibility space. These particular paths all exist inside of one precalculated step of A*, hence the lack of variation in moves and no shared points in time.

cost function of A* has a noticeable effect on the paths through the expressive possibility space. This change is most obvious in comparing figures 8.3, which uses the $|\Delta goal\_val|$ cost, and 8.4, which uses the $||\Delta max\_val| - |\Delta goal\_val||$ cost, as every affect able to be expressed has a unique set of moves to reach it in the latter, whereas there are only two paths in total in the former. The point of these comparisons is not to provide a recommendation for which formula produces better performances. Rather, ERC is meant to help inform the iteration and development of expressive characters by providing further insight into both the contours of expressive possibility space and the ways AI characters understand the domains they operate in.

8.6 Discussion and Future Work

By creating ERCs using Puppitor's character rulesets, I have been able to gain significantly more insight into the affordances of the system as an expressive domain for players and developers working in this area. On the player-oriented side, the use of ERCs can help identify overly homogeneous areas of a Puppitor ruleset and affects that are complex or impossible to express from certain points in a ruleset. For developers, having a fast way to view the effects that changes in domain knowledge and representation have on AI characters allows for much more deliberate iteration.

Our initial phase of analysis was focused on evaluating the responsiveness of ERCs created from a Puppitor ruleset. I define this property as the property of pathways which only require a single move to be made to see a change in the prevailing

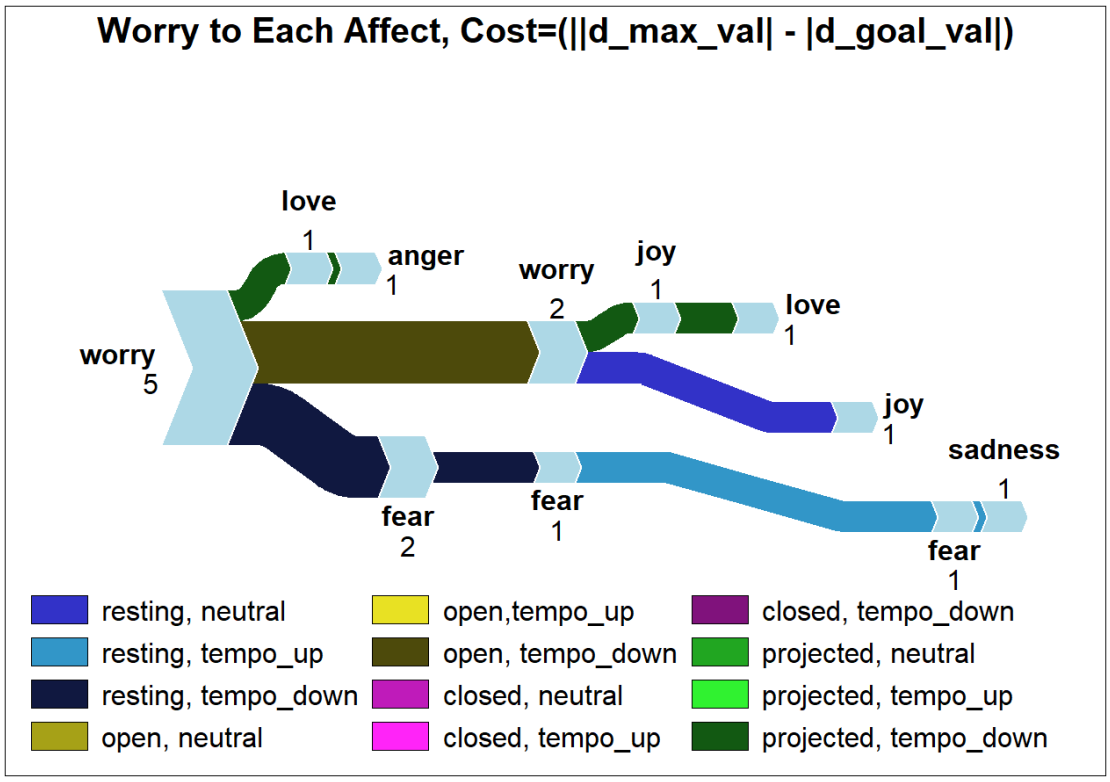

Figure 8.5: A different cost function doesn't always lead to dramatically different paths through the possibility space, but affects that can be easily expressed are much more likely to be found early in the search.

affect. The generated maps (figures 8.1, 8.3 , 8.4, and 8.5) show that the chosen ruleset often exhibited responsiveness in the individual ERCs, where individual move choices frequently led to changes in prevailing affect before a new move was considered. The starting points I chose for our figures, worry and joy, were chosen to show the effect the values inside an affect vector have on the difficulty and responsiveness of certain affects. I designed the worry starting point to make expression of a subset of affects difficult, shown in figures 8.1 and 8.5. Even then, the only affect goal which required A* to perform a single action for multiple nodes in its search was sadness. By evaluating a Puppitor ruleset by the responsiveness of the ERCs it produces, I can highlight the relative player experience of expressing affects from different locations in the possibility space.

The creation and analysis of ERCs for Puppitor enabled informed iteration of domain knowledge for A* as well as garnered insight into the way that domain knowledge changes the expressive performance of the search. I originally only used $|\Delta goal\_val|$ as the cost function to produce ERCs, but noticed that it rarely made paths containing obvious move choices when comparing its decisions with the ruleset definition. These comparisons led us to the realization that to gain more full view of even these small sections of expressive possibility space, a single set of domain knowledge would not be sufficient, hence our usage of $||\Delta max\_val| - |\Delta goal\_val||$ as an additional cost function. With the two cost functions side by side, I could then characterize the performance styles they preferred, with the first ($|\Delta goal\_val|$) preferring more subtle moves and the second ($||\Delta max\_val| - |\Delta goal\_val||$) preferring slightly more active moves.

Comprehensively visualizing the information provided by ERCs is a challenge due to the nuances in the sequences of actions and their relationship to the surfaced player feedback present even in smaller domains like Puppitor. Knowing this, I note that while the Sankey diagrams our analysis of ERCs produced prove useful in illustrating the flow between affects and their relationship to player actions, these diagrams abstract away some important information for analyzing game feel. When describing these diagrams, I stated that nodes within the same column occur at the same point in time. The Sankey diagrams do maintain some of the information about the progression of time through nodes' column positions, but they flatten the details about the exact progression over time. This is especially notable for both worry trials (shown in figures 8.1 and 8.5). The diagrams cannot accurately capture the fact that the transition from worry to joy when performing the `(projected, tempo_down)` move occurs in a single step. As a result, further work must be done to enable the creation of more comprehensive visualizations of expressive possibility spaces.

With this in mind I want to highlight the complexity of the expressive possibility spaces I discuss in this paper or, more broadly, the difficulty of visualizing playtesting data [5]. While I have been able to visualize small pieces of a Puppitor ruleset, current widely available visualization techniques limit us to a single starting point per diagram. The combinatorial explosion of even two or three starting points in a single diagram quickly renders them unusable for the purpose of analysis. Mapping every path through an expressive possibility space, and importantly presenting such paths in a user-friendly way, is an open research problem and far beyond the scope of this paper. With that

said, ideally I will eventually be able to create tools to allow the full searching and mapping of expressive possibility spaces and effectively illustrate the relationships between clusters of paths. This fits the broader trend of games being incredibly difficult artifacts to adapt to data visualization. I plan to pursue the development of more effective visualization techniques for expressive possibility spaces in the future.

Puppitor provided an ideal testing platform for the creation and analysis of ERCs. It features clearly defined states with a well defined relationship to actions a player could perform, complete with values that could easily be turned into domain information for A* search. This clarity in the original domain made for a relatively straightforward translation process, but this will not necessarily hold true for other current and future expressive domains which ERCs may be useful to. Additionally, Puppitor has a built in way of signaling changes in its state that are human perceptible, its prevailing affects. Rather than having to derive a novel method for discovering important state transitions, Puppitor's prevailing affect system make this a core feature of its interface. The system's affordance therefore lightens the translation and adaptation burden by removing the need to synthesize a new approach to identifying human perceivable state transitions. Through our experience creating ERCs for Puppitor, I recognize that not all domains which can benefit from ERCs would have these features built in. As our current work with ERCs shows, I hope to encourage more systems to expose and highlight state changes that impact player experience.

8.7 Conclusion

In this chapter I describe a method of automated playtesting which produces expressive response curves (ERCs) which can then be combined into a map to characterize the features of an expressive possibility space. Through our case study, I demonstrated that the insights provided by ERC maps allow for more informed reasoning about the domain on the part of designers, which in turn can lead to new ways of guiding A* performance within an expressive possibilty space. Combined with the more intuitive view of an expressive possibility space facilitated by ERCs, enable a far better understanding of the complexity of expressive domains and the nuances of their relationship to the development of AI characters focused on performance and expression. I hope that our work with ERCs inspires more work in the future focused on advancing our understanding of the player experience created by expressive AI models.

Chapter 9

Comparing the Aesthetics of Algorithmic Performance

Using AI is an aesthetic choice. I made Puppitor and *Tracks in Snow* to further my own aesthetic goals. I use the particular AI systems with both projects in part because they conform to my aesthetic sensibilities. Even in extremely simple scenarios, the choice of algorithms, and more broadly the technical systems, has a direct impact on the aesthetics of an experience or interaction with an AI. At some point in any project, that fact will need to be grappled with. Through the work in this chapter I hope to highlight the need to reckon with those design implications early and enable developers to make more informed choices related to the complex technical systems they create.

The evaluation work in this chapter is a confluence of small, long running components of *Tracks in Snow* and the previous chapter's automated playtesting work.

I intentionally designed Puppitor as a domain for multiple kinds of AI, and since 2022, *Tracks* has featured three different options for searches to drive the AI acting component of the game. Even before my work with Expressive Response Curves solidified into what would become chapter 8, I had noticed the sometimes subtle, sometimes dramatic differences in acting styles between the three algorithms in *Tracks*. Now, Puppitor rulesets, and the character definitions in the visual novel especially, are fairly simple domains. The searches in *Tracks* are not the most complicated or inscrutable AI systems out there. And yet, they provide different acting experiences. There are high level design reasons to use each of the algorithms.

9.1 Introduction to Aesthetic Comparison

As Artificial Intelligence (AI) systems grow increasingly complex, it is more important than ever for designers to understand the impact of their technical choices on the aesthetic experiences they create. Conventional evaluation of AI character behavior has historically focused on properties of computational efficiency and solvability. Less consideration has been paid to the way these system shape interaction with a player. This lack of focus misses that, how a character behaves has a considerable impact on a player's understanding of the system's aesthetics, the narrative, and the broader play experience. I call these implications and the relationship between the technical design, performance, and player understanding of a character's behavior the aesthetics of that AI system. From this understanding, due to the current lack of evaluation methods,

the aesthetic implications of technical decisions on character behavior at design time remain opaque to system designers.

Recent work in automated playtesting has led to the development of tools which incorporate player experience as an critical component of analyzing game elements [18, 125, 225]. These advances, however, use AI systems to provide insights about the player experience of game systems or content rather than the interactions with an AI system itself. By instead focusing on AI systems themselves, I can use automated playtesting techniques to provide insight into the aesthetic qualities of algorithms which drive character behavior in games.

In this chapter I propose the Aesthetic Comparison of Algorithms (ACA) method of evaluating AI driven character behavior. ACA combines the evaluation approaches of Expressive Response Curves (ERCs) [125] and Action Agreement Ratio (AAR) [108, 225] to understand the impact different underlying algorithms have on the details of character behavior. This evaluation approach relies on the creation of playtraces synthesized into ERCs, which provide descriptions of the visible impact of actions on character state. The ERCs generated by different algorithms are then compared using AAR to discover how similar the approaches to completing a goal are between algorithms. ACA, thus, reveals details about micro level decision making which can influence the high level experience of character behavior.

The domain of an ACA test must remain constant to allow for comparisons between algorithms. As a result, this method can highlight points of convergence and divergence in the playtraces of different algorithms in a given domain. These locations

thus show the aesthetic trade offs between technical implementations of character behavior for a given domain. Which of these trade offs make the most sense is highly subjective and project dependent. My goal with ACA is to provide a way for designers to make informed decisions about the technical underpinnings of aesthetic experiences, not a strict set of guidelines for which algorithms produce high quality results.

I describe my approach to creating a test suite and evaluating algorithms using ACA. Additionally, I provide a case study of ACA as applied to Puppitor [126], an embedded domain specific language which enables character expression through physicality, using a set of three algorithms—greedy search, A* search [100], and Monte Carlo Tree Search (MCTS) [47]. I chose Puppitor for this case study as it is a lightweight framework for character behavior capable of creating highly complex domains and has been used in a playable experience [124]. Additionally, Puppitor is freely available and publicly accessible, including pre-existing implementations of the three algorithms in my comparison. Finally, I discuss the results of ACA using the case study and the implications the method has for the development of AI driven character behavior as well as interaction between human and AI systems more generally.

9.2 Related Work

The core of the ACA methodology is built upon the combination of Action Agreement Ratios [108, 225] and Expressive Response Curves [125]. In automated playtesting terms, AAR represents how often two player personas choose the same

action [225]. For every state in a playtrace created by one persona, a second is queried for the action it would perform in the given state [225]. The final ratio is then the number of the times both personas choose the same action divided by the total number of states in the play trace [225]. I modify this approach to work in conjunction with ERCs to perform ACA on pairs of algorithms. ERCs represent sequences of actions and states as found by A* which highlight any human perceptible changes in character state [125]. Combining multiple ERCs into an ERC map then exposes the domain's level of responsiveness to possible player actions and allows for better understanding of the relationship between action and feedback [125]. Rather than running a second algorithm on a single generated playtrace for ACA, I compare the ERCs created by different algorithms using the same initial state and goal. This modification allows us to better understand the ways different algorithms approach solving the same problem to achieve a given goal.

Automated playtesting more broadly grew from the desire for rapid evaluation of generated content. Such content includes video game levels [107, 159, 213], game balance [207, 208, 282], and skill progression [110]. Personas capturing human playstyles are a common feature of automated playtesting, often guiding MCTS based agents [63,135,196,282]. These methods focus on the evaluation of artifacts rather than the testers themselves. ACA, conversely, emphasizes understanding the behavior and decision making process driving characters, such as automated playtesting agents.

Expressive range analysis (ERA) is an approach to describing generators based on descriptions of the artifacts produced [245]. ERA has been used in design sup-

port tools such as Danesh [57] and, importantly, to characterize automated playtesting data [5]. While ACA focuses more on algorithmic behavior than complete artifacts, its philosophical approach is similar to ERA. The observable aspects of a computational system provide valuable insight into their aesthetic qualities, be they chosen actions or generated artifacts, when compared in aggregate.

As a quantitative approach to describing the aesthetic components of expressive and narrative focused work, ACA parallels the motivations of both tension space analysis (TSA) [147] and progression maps (PMs) [43]. TSA takes inspiration from ERA and, similar to my goals with ACA, seeks to help authors of emergent narrative works better understand the technical and formal properties of underlying systems [147]. While TSA and ACA both are highly applicable to character behavior, TSA operates at a higher level of character motivation than the low level algorithmic goals of ACA. PMs are a graph-based model of interactive narratives focused on exposing their structure and the resulting effects on the player experience [43]. Like ACA, the goal of PMs is to analyze the impact of design on players' aesthetic experience. The two methods' main point of divergence is in exact focus and granularity, with ACA prioritizing extremely low-level algorithmic behavior while PMs prioritize high level structure. ACA thus shares a lineage with these quantitative approaches to analyzing the aesthetic impact of underlying technical implementations.

General game playing (GGP) seeks to provide test environments for intellegent agents able to learn a variety of games without human intervention [76, 90, 206, 257]. As a result, the agents created for GGP rarely can rely on domain specific knowledge

of specific games to effectively make choices, leading to the popularity of MCTS in the field [76, 257]. ACA's primary relation to GGP is through the Game Description Language (GDL) [257]. GDL is a common domain to test GGP agents, and while ACA does not mandate particular domains, its approach greatly benefits from domains authored without specific AI implementations in mind. Through GDL, GGP informed my selection of Puppitor as the exemplar domain.

9.3 The Aesthetic Comparison Approach

The goal of creating ACA is to provide designers with a method of analyzing the impact underlying technical implementations have on the aesthetics of AI behavior. ACA combines the ERC and AAR evaluation methods used in automated playtesting to understand the aesthetic differences of algorithms operating within a domain. An ERC is a sequence of actions performed and states reached by a player as they move through an expressive possibility space [125]. AAR represents how frequently two players choose the same action [108, 225]. By analyzing the ERCs produced by different algorithmic decision making processes with AAR, I can characterize the impact of technical decisions on the aesthetic experience of a character. The entire process of building a test suite and performing comparative analysis using these two methods are as follows:

- **Algorithm Selection:** Choose a set of algorithms to test, including a simple baseline, and make any modifications to allow them to output their results in the same format.

- **Domain Selection:** Choose a domain for the algorithms to use and make any modifications required for the chosen algorithms to function as expected.
- **Running Tests:** The initial states and goals are chosen based on the domain representation. Tests for each algorithm are run from each initial state to each goal.
- **Creating ERCs:** Synthesize ERCs and ERC maps from the results of the test suite.
- **Calculating AAR:** Compare pairs of ERCs with the same initial states and goals but different algorithms using AAR to calculate the similarities between the sequences of actions.

9.3.1 Algorithm Selection

Choosing a set of algorithms to test is highly project dependent, but there is one constant for all uses of ACA. A strong set of algorithms must include a simple, minimum viable baseline fitting a project's requirements. Having a simple baseline, such as a uniform random or greedy algorithm, establishes a benchmark which any more complex algorithms must at least match in aesthetic quality.

Once a suite of algorithms is chosen, their output must be normalized to enable effective comparisons between them. As the synthesis of an ERC requires a path from the initial state to the goal, the output of each algorithm must be a sequence of actions and their resulting states along with any other information pertinent to the chosen

domain. For example, greedy search is not traditionally a pathfinding algorithm but can be made to output a path useful for ERC synthesis by storing each search result in a list and running the search until it reaches the specified goal.

9.3.2 Domain Selection

A domain can be any set of rules an AI is capable of interfacing with: from expressive character behavior [126] to general game playing [85] to social simulations [72, 182] and more. To utilize the methodology described in this chapter, the chosen domain must have features corresponding to actions which update states. Actions and states may be compound entities, as is the case with Puppitor, where an action in this method's sense are broken into Puppitor actions and modifiers while the state is made up of an affect vector and the prevailing affect [126].

9.3.3 Running Tests

Choosing a set of tests is dependent on the goals of comparing algorithms in the first place. In a constrained domain, the number of test conditions may only need to be minimal to effectively characterize the way each algorithm behaves. In more complex domains, there may need to be significantly more test conditions as the convergence or divergence of algorithmic behavior is likely not uniform across state space. As a result, a useful set of initial states should cover as wide a gamut of the domain as possible. Similarly, the set of goals should cover as many potential states as possible, even if a particular goal is not obviously achievable from a subset of the initial states. A wide set

of initial states and goals allows a fuller picture of the relationship between the chosen domain and algorithms to be created. Such a set of initial states and goals also reveals the subtle differences between each algorithm's behavioral choices, enabling aesthetic goals to be in more direct conversation with technical requirements.

Stochastic algorithms, such as Monte Carlo Tree Search [47], as part of a test suite, require additional steps to create useful playtraces for analysis. Comparing between algorithms requires a seed to be specified for any stochastic algorithms used to ensure consistent behavior. Without a seed, the algorithms produce too many slightly different play traces to allow easy comparison between different algorithms.

The stochastic nature of these algorithms is important to fully characterizing their behavior with ACA. The method requires multiple trials of the same initial state and goal pair to build up an aggregate of chosen sequences of actions. Such algorithms require at least ten trials per initial state and goal combination to provide a sample of behavioral consistency. The output of each individual trial can then be used to create an ERC to then analyze the internal consistency of a stochastic algorithm's playtraces.

9.3.4 Creating ERCs

Synthesizing ERCs from sequences follows the method Junius et al. previously described [125]. Each algorithm's trial is output into a timestamped sequence of state and action information. Any human perceptible changes, such as changes in actions or state which impacts visuals, in these sequences are then highlighted and used to create an ERC.

These ERC describe the aesthetic qualities of an algorithm's behavior and how frequently changes in action lead to player-visible changes in state. As a component of ACA, ERCs provide low level details about the effects of an algorithm's choice of actions on the aesthetics of behavior. These details are primarily the timing of visible state changes and the actions leading to them. As a result, ERCs tend to emphasize state transitions more than the actions themselves.

9.3.5 Calculating AAR

AAR is calculated using pairs of ERCs and generally follows the method described by Robertson et al. [225]. The ratio itself is the number of times the two ERCs have the same action at the same time stamp divided by the total number of action changes in the ERC. A score of 0.0 represents complete divergence of behavior, while a score of 1.0 represents complete convergence of behavior. Resultingly, AAR's high level view emphasizes actions over their effects.

The method presented here primarily calculates the AAR of ERCs produced by different algorithms to compare the experiential differences in their decision making. Stochastic algorithms provide an exception as their ERCs are rarely completely consistent across all trials of the same initial state and goal pair. In this case AAR can determine how constrained a stochastic algorithm is within a particular part of the domain. Additionally, when used with a stochastic algorithm, AAR can identify which sequence, if any, are at least the plurality of trials. Sequences identified as a plurality, or majority, of playtraces can then be used to compare stochastic algorithms to other

algorithms in the test set.

9.4 Case Study: Comparing Algorithms within Puppitor Rulesets

Performing ACA requires a domain capable of being understood by multiple AI systems. The Puppitor embedded domain specific language provides an ideal testing environment focused on expression [126] rather than general game playing. Additionally, the *Tracks in Snow* visual novel [124], which uses the system, already features a selection of three algorithms to drive character behavior. The nature of Puppitor rulesets thus allows my work in this case study to focus on normalizing the output of my chosen set of algorithms rather than on domain creation.

9.4.1 Puppitor as a Domain

A Puppitor ruleset is broken into affect entries, each of which contain a set of *actions*, a set of *modifiers*, a set of *adjacent affects*, and an *equilibrium point*. This structure is exemplified by one of the affect entries in the Chiara character ruleset, extracted from *Tracks in Snow*:

```
"love" : {
    "actions" : {
        "resting" : -0.00082,
        "open_flow" : 0.002,
        "closed_flow" : -0.00088,
        "projected_energy" : -0.0006
    },
    "modifiers" : {
```

```
        "tempo_up" : 0.8,
        "tempo_down" : 1.27,
        "neutral" : 1.0
    },
    "adjacent_affects" : {
        "joy" : 90,
        "worry" : 10
    },
    "equilibrium_point" : 0.28
}
```

The values associated with actions and modifiers are used to update a character's state, known as the *affect vector*. When an action and modifier are specified, their values are multiplied together then added to the affect vector entry corresponding to the affect entry. In the above case all entries are relevant to the "love" value of a given affect vector.

The other element of a Puppitor ruleset important for ACA is the *equilibrium point*. The equilibrium point defines the value an affect vector entry will move towards when the *default action* is chosen, regardless of the sign of the value of that action. All tests I describe in this chapter use "resting" as the default action.

9.4.2 Algorithm Selection and Modification

I chose greedy search, A* search, and MCTS as my test set for this chapter due to their varying approaches to problem solving, common usage in commercial games, and speed of integration with domains. Greedy search provides a baseline in both algorithmic complexity and behavior as its decision making process is extremely simple and transparent while being more suited to the Puppitor domain's usage than purely

Algorithm 1 Pathfinding Wrapper

Input: Domain and goal information
Parameter: *initial_state*, *domain*, *goal*
Output: Path to goal

```
1: Let final_path = [(initial_state)]
2: Let curr_node = final_path[0]
3: while curr_node[state] ≠ goal do
4:     search_result ← search(state_cp, domain, goal)
5:     final_path ← (search_result)
6:     curr_node ← final_path[−1]
7: end while
8: return final_path
```

random choice. A* search has already been used as part of understanding Puppitor rules [125] and has provided the foundation of AI systems driving character behavior [202,203] or level evaluation [108,119]. MCTS is an algorithm commonly used in general game playing [76, 256, 257] and automated playtesting contexts [110, 135, 196, 282].

The primary challenge of performing ACA with this set of algorithms is that only A* produces full paths suitable for creating and evaluating ERCs. Greedy search and MCTS traditionally only output single moves and must be run every time a new move is desired. To resolve this difference in output, I created wrappers around both searches to allow them to string their outputs together into complete paths. The wrappers took the form of a loop which repeatedly calls a search until it has found a state fulfilling the goal conditions while creating a sequence of actions and states, described in Algorithm 1.

I use the following pieces of domain knowledge in the implementations of each of the algorithms used for testing. For my A* implementation, I use a cost calculation of $||\Delta max\_val| - |\Delta goal\_val||$ and a heuristic described in Algorithm 2. In this

Algorithm 2 A* Heuristic

Input: Current state and goal information
Parameter: *vector*, *expressed*, *goal*
Output: Distance to goal estimate

1: Let $nodes = max(vector)$
2: Let $m\_v = value(nodes)$
3: **if** $goal \in nodes$ and $expressed \neq goal$ **then**
4: $\quad heuristic \leftarrow large\_num$
5: **else**
6: $\quad heuristic \leftarrow dist(m\_v, goal\_value)$
7: **end if**
8: **return** *heuristic*

function, *nodes* stores all affects which may be expressed, m_v stores the value with the highest magnitude in a given affect vector (represented by *vector*), *expressed* is the currently expressed affect, and *goal* is the affect the algorithm must express. The cost and heuristic calculations are available in the public Puppitor repository and described by [125].

Both greedy search and MCTS use the function described in Algorithm 3 to evaluate Puppitor state. In this function, m_a is the set of affects with the greatest magnitude taken from a given affect vector (represented by *vector*), c_a represents the currently expressed affect, and *goal* is the affect that must be expressed. This function is a part of Puppitor itself in its public repository on GitHub. This evaluation function scores Puppitor moves based on the magnitude of positive change to a given state's value corresponding to the specified goal affect. When the goal affect is in m_a and yet is not c_a, the function heavily penalizes the state.

This specific state is penalized heavily as otherwise the score calculation would potentially encourage any algorithm using the evaluation function to continue to choose

Algorithm 3 Evaluate Affect Vector

Input: The affect vector and goal
Parameter: *vector*, *c_a*, *goal*
Output: Score of *vector*

```
1: Let score = 0
2: Let m_a = max(vector)
3: if c_a = goal then
4:    score ← 10
5: else if len(m_a) > 0 and goal ∈ m_a and c_a ≠ goal then
6:    score ← −10
7: else
8:    for all a in vector do
9:       if a ≠ goal then
10:         score ← score + (val(goal) − val(a))
11:      end if
12:   end for
13: end if
14: return score
```

actions which would not in fact enable it to achieve its goal. When using Algorithm 3 to score a given affect vector, actions which increase both the value of the goal affect and another affect of equal value may still be scored highly when both affects are at their ceiling values. In such a case, scoring an affect vector based on the increase in value of the goal affect compared to all other affects will cause algorithms to become stuck choosing actions which do not help them reach their goal. By penalizing this state, the search algorithms are forced to choose an action that would normally be less ideal but in this particular state enables them to choose an action which rather than focusing on increasing the goal's value instead lessens the current affect's value.

When performing ACA on Puppitor, the individual entries in the generated paths use the format of: (affect vector, action, modifier, prevailing affect). The *affect vector* is the full set of affects and their strengths relative to each other. The *prevailing*

affect is the affect in the vector with the highest relative value to the others. Together the affect vector and prevailing compose ACA's definition of state. The *action* and *modifier* make up the chosen Puppitor move which led to the current entry as well as ACA's definition of action.

9.4.3 Domain Selection and Generation

For this case study of Puppitor rulesets using ACA, I chose rules from three different sources for a total of three distinct sub-domains. Importantly, each of these rulesets uses the same set of affects and moves to allow us to characterize the rulesets relative to each other. The first two rulesets are the character definitions extracted from the *Tracks in Snow* visual novel. These rulesets provide examples of domains authored with responsive interaction between human and AI players in mind. The final ruleset is the *test_passions_rules.json* provided in the public Puppitor repository. This ruleset is useful for providing a baseline of Puppitor functionality as it is provided with the system and is not expressly designed for particular aesthetic goals like the character definitions from *Tracks in Snow*.

9.4.4 Initial States and Goals

The creation of a full test suite required us to create six different initial states and six different goals to create a representative sample of Puppitor playtraces. The six initial states are affect vectors each feature a different prevailing affect and affect vector topology. I use each of the six affects in my chosen Puppitor rulesets: anger, fear, love,

joy, sadness, and worry for my goals. Due to the scale of the possibility space created by Puppitor rulesets, a fully comprehensive mapping is beyond the scope of this chapter.

Even my relatively small set of initial state and goal pairs requires 36 trials per algorithm per domain. My full test suite requires a total of 432 trials to test each algorithm with each ruleset and cover all possible state and goal pairs. As a result, I only discuss a subset of these trials in this chapter.

9.4.5 Creating Puppitor ERCs

As I use similar domains, my synthesis of ERCs from playtraces follows the methodology described by Junius and Carstensdottir [125]. From each run of an algorithm on a given initial state and goal pair, I highlight all the changes to Puppitor actions and modifiers along with any changes in prevailing affect to construct the ERC. Each of these changes has a corresponding simulation time stamp which exposes the responsiveness of parts of a given domain.

Additionally, as part of my testing, I use a step multiplier similar to Junius and Carstensdottir [125] to reduce calculation time as well as reflect how these algorithms are used in *Tracks in Snow*. Puppitor rulesets generally assume operation over continuous state. As a result, any individual state change using Puppitor is extremely small. A step multiplier in this case enables each algorithm to commit to performing the same action for a set amount of time. This commitment reduces the number of times an algorithm must make computationally expensive decisions. Additionally, locking in a choice of actions allows state updates to compound into meaningful differences rather

than asking the algorithm to re-evaluate every incremental change. For the rulesets extracted from *Tracks in Snow*, I use a step multiplier of 90, the same as the visual novel. For the ruleset taken from the Puppitor repository as well as the one I generated, I use a multiplier of 180.

I used a seed of 4157841963 when creating playtraces to compare MCTS with A* search and greedy search. This seed forces MCTS into having consistent behavior across trials, necessary for comparing choices of actions between algorithms. By happenstance, my chosen seed saw MCTS producing playtraces of similar simulation length to A* search and greedy search, described in Tables 9.1 through 9.3.

9.4.6 AAR Comparison

I compare pairs of ERCs generated by different algorithms using a slightly modified version of AAR. Rather than querying a second algorithm using states from an ERC, I directly compare the generated ERCs themselves. Characterizing the aesthetic behavior of algorithms benefits from seeing the points at which algorithms' decisions converge or diverge. Characters are defined by how they express themselves and accomplish goals, not only by what they express or which goals they achieve [29, 158, 223].

My AAR calculations thus increments the ratio's denominator when either algorithm takes a new action. The numerator is only incremented when the two algorithms' chosen actions diverge. The step multiplier ensures that each algorithm is able to choose a new action at the same simulation time. For example, if one ERC shows a change in action at the 90th time stamp and the other has no entry for the 90th time

stamp, I know the two ERCs diverged as the latter chose to continue the same action as before. As a result, I would increment the denominator but not the numerator of the AAR between these two algorithms.

9.5 Results

Due to the volume of data my full test suite produces, I only highlight a subset of the trials in this section. I focus on comparing A* search, greedy search, and MCTS across three different initial state and goal pairs using the Chiara character definition extracted from *Tracks in Snow*, seen in Tables 9.1 through 9.3. These initial state and goal pairs are: anger and joy, fear and love, and worry and sadness. I designed the initial states to be less than ideal scenarios, where the goal affect's value is set to near 0 and most other affects' values are set over 0.5. This spread of values meant that the algorithms were not always able to perform a single action to express their goal, exemplified by the anger initial state:

```
"joy" : 0.0,
"anger" : 0.8,
"sadness" : 0.4,
"fear" : 0.6,
"worry" : 0.7,
"love" : 0.5
```

Additionally, I include a within algorithm comparison of MCTS across ten trials using the worry and sadness case on the Chiara definition, shown in Table 9.4. These trials were conducted without the random seed of 4157841963. I chose this set of conditions primarily because of the perfect agreement shown in the between algorithm

comparisons to demonstrate that a stochastic algorithm will not exhibit completely consistent behavior even in highly constrained domains.

The tables show ERCs produced by each algorithm, with time, affect, and move components, separated by vertical double lines. Time is the number of simulation steps since beginning the trial. Affect is the expressed affect at the given time. Move is the Puppitor action and modifier combination used to reach the current state. Puppitor actions and modifiers are condensed down to a two letter format. The left letter corresponds to a Puppitor action and the right corresponds to a Puppitor modifier, described below:

```
projected_energy: p
closed_flow: c
open_flow: o
resting: r
neutral: n
tempo_up: u
tempo_down: d
```

Additionally, each table highlights where algorithms are able to change their actions with gray bars. These occur in increments of 90 simulation steps when using Chiara's character definition. This value correspond to the step multipliers described previously. Algorithms re-evaluate state at simulation step 1 rather than 0 as the initial state is fixed.

9.5.1 AAR Calculations

I calculate the AAR of the each pair of ERCs using the Chiara character definition with an initial state of anger and a goal of joy, shown in Table 9.1. The AAR

A* Search			Greedy Search			MCTS		
Time	Affect	Move	Time	Affect	Move	Time	Affect	Move
0	anger	r-n	0	anger	r-n	0	anger	r-n
1	anger	p-u	1	anger	p-u	1	anger	r-n
91	anger	p-u	91	anger	p-u	91	anger	r-u
149	anger	p-u	149	anger	p-u	149	worry	r-u
181	anger	p-u	181	anger	p-u	181	worry	o-u
185	anger	p-u	185	anger	p-u	185	anger	o-u
271	anger	o-u	271	anger	p-u	271	anger	p-u
327	joy	o-u	327	anger	p-u	327	anger	p-u
358	joy	o-u	358	anger	p-u	358	joy	p-u
361	joy	o-u	361	joy	o-d	361	joy	p-u

Table 9.1: Test with an initial state of anger and a goal of joy. Puppitor moves use the format of action-modifier, where r- is the resting action, p- is the projected action, o- is the open action, and c- is the closed action. Modifiers are represented as: -n for neutral, -u for tempo up, and -d for tempo down. Gray rows denote each time the AI system is allowed to re-evaluate its choice of Puppitor move. These rows and time stamps correspond to the actions used when calculating AAR. The AAR of A* and greedy search is 0.6, the AAR of A* and MCTS is 0.0, the AAR of greedy search and MCTS is 0.2.

of A* search and greedy search is 3/5, or 0.6, as both algorithms choose the same moves until time 271. The AAR of greedy search and MCTS is 1/5 or 0.2, as only at time 271 do they choose the same move. The AAR of A* search and MCTS is 0/5 or 0.0, as even if the algorithms choose many of the same moves, they perform them in near reverse order.

In this test case, each algorithm produces both a mathematically and aesthetically different experience. The underlying states of each ERC are distinct, as the algorithms all express joy at different times, hence the mathematic divergence. Importantly, this is reflected in the aesthetic differences beyond expressed affect. Divergent choices of moves will make characters feel different, even if they ultimately achieve the same goals. Of particular note is that MCTS expressed worry from time 149 to 181,

A* Search			Greedy Search			MCTS		
Time	Affect	Move	Time	Affect	Move	Time	Affect	Move
0	fear	r-n	0	fear	r-n	0	fear	r-n
1	fear	o-d	1	fear	o-d	1	fear	r-n
91	fear	o-d	91	fear	o-d	91	fear	o-n
131	joy	o-d	131	joy	o-d	131	fear	o-n
181	joy	o-d	181	joy	o-d	181	fear	o-d
271	joy	r-n	271	joy	o-d	271	fear	o-n
290	joy	r-n	290	love	o-d	290	fear	o-n
302	joy	r-n	302	love	o-d	302	joy	o-n
318	love	r-n	318	love	o-d	318	joy	o-n
321	love	r-n	321	love	o-d	321	love	o-n
361	love	r-n	361	love	o-d	361	love	o-n

Table 9.2: A* search, greedy search, and MCTS operating on the Chiara character definition with the initial state of fear and a goal of love. The AAR of A* and greedy search is 0.6. The AAR of A* and MCTS 0.2. The AAR of greedy search and MCTS is 0.2.

while A* search and greedy search only ever expressed anger or joy.

The ERCs produced using the initial state of fear and a goal of love with the Chiara character definition, shown in Table 9.2, are more similar to each other than with the anger and joy trial. The AAR of A* search and greedy search in this case is 3/5, or 0.6, as the two algorithms diverge at time 271 again. The AAR of greedy search

A* Search			Greedy Search			MCTS		
Time	Affect	Move	Time	Affect	Move	Time	Affect	Move
0	worry	r-n	0	worry	r-n	0	worry	r-n
1	worry	p-d	1	worry	p-d	1	worry	p-d
9	fear	p-d	9	fear	p-d	9	fear	p-d
65	anger	p-d	65	anger	p-d	65	anger	p-d
163	sadness	p-d	163	sadness	p-d	163	sadness	p-d
181	sadness	p-d	181	sadness	p-d	181	sadness	p-d

Table 9.3: A* search, greedy search, and MCTS operating on the Chiara character definition with the initial state of worry and a goal of sadness. The AAR between all algorithms is 1.0 as each produces an identical ERC.

MCTS$_a$			MCTS$_b$			MCTS$_c$			MCTS$_d$			MCTS$_e$		
T	A	M	T	A	M	T	A	M	T	A	M	T	A	M
0	w	r-n	0	w	r-n	0	w	r-n	0	w	r-n	0	w	r-n
1	w	p-d	1	w	r-d	1	w	r-u	1	w	r-n	1	w	r-n
9	f	p-d	9	w	r-d	9	w	r-u	9	w	r-n	9	w	r-n
65	a	p-d	65	w	r-d	65	w	r-u	65	w	r-n	65	w	r-n
91	a	p-d	91	w	p-d	91	w	p-d	91	w	r-d	91	w	r-n
113	a	p-d	113	f	p-d	113	w	p-d	113	w	r-d	113	w	r-n
119	a	p-d	119	f	p-d	119	f	p-d	119	w	r-d	119	w	r-n
150	a	p-d	150	f	p-d	150	a	p-d	150	w	r-d	150	w	r-n
163	s	p-d	163	f	p-d	163	a	p-d	163	w	r-d	163	w	r-n
166	s	p-d	166	a	p-d	166	a	p-d	166	w	r-d	166	w	r-n
181	s	p-d	181	a	p-d	181	a	p-d	181	w	p-d	181	w	p-d
199	s	p-d	199	s	p-d	199	a	p-d	199	w	p-d	199	w	p-d
219	s	p-d	219	s	p-d	219	s	p-d	219	w	p-d	219	w	p-d
220	s	p-d	220	s	p-d	220	s	p-d	220	f	p-d	220	w	p-d
224	s	p-d	224	s	p-d	224	s	p-d	224	f	p-d	224	f	p-d
245	s	p-d	245	s	p-d	245	s	p-d	245	f	p-d	245	a	p-d
251	s	p-d	251	s	p-d	251	s	p-d	251	s	p-d	251	a	p-d
255	s	p-d	255	s	p-d	255	s	p-d	255	s	p-d	255	s	p-d

Table 9.4: Ten trials of MCTS operating on the Chiara character definition with the initial state of worry and a goal of sadness. The MCTS$_a$ column represents 6 ERCs with the exact same action sequence. The T, A, and M columns are the same as the Time, Affect, and Move columns in the other tables, just compacted for space. Similarly, the letters in the A columns correspond to the affects shown in other tables. These ERCs demonstrate how much importance Puppitor rulesets place on early actions due to the continuous nature of the state.

and MCTS is 1/5, or 0.2, as MCTS chooses the same move as greedy at time 181 and nowhere else. The AAR of A* search and MCTS is 1/5 or 0.2, as the two algorithms make the same choice at timestamp 181.

Again, each algorithm produces an ERC that is both mathematically and aesthetically distinct. At most, there is agreement about action choice early in the ERCs, and the diverging moves indeed change the timing of expressing the goal affect. MCTS again provides the most divergent result, reflected in both the AAR and how

	a	*b*	*c*	*d*	*e*
a	1.0	0.66	0.66	0.33	0.33
b	0.66	1.0	0.66	0.33	0.33
c	0.66	0.66	1.0	0.33	0.33
d	0.33	0.33	0.33	1.0	0.66
e	0.33	0.33	0.33	0.66	1.0

Table 9.5: The AARs of all MCTS trials using the initial state of worry and a goal state of sadness run over the Chiara character definition. The letters correspond to the MCTS labels in Table 9.4.

the ERC itself spends most of the playtrace expressing fear.

The ERCs produced by the initial state of worry and a goal of sadness break with the other two Chiara definition based examples. There is perfect agreement across all three algorithms for an AAR of 1.0 and identical timing of affect changes. These ERCs are also significantly shorter in length than any of the other examples, achieving their goal before they are given a third opportunity to change their move choice at time 181.

Because of this complete convergence across algorithms, I also used this case to demonstrate a within algorithm comparison using MCTS, shown in Table 9.4. The ERC labeled as MCTS_a in this table represents 6 of the 10 total trials. This convergence means that, even without a specified seed, the algorithm is likelier than average to perform the same sequence of actions, unlike other trials I performed with non-seeded MCTS. The AAR comparison across all the ERCs is shown in Table 9.5, where there was a convergence on a single move by time 181. AAR alone does not give quite the full picture of differences between the ERCs. With the exception of MCTS_a, the algorithm would decide to begin its behavior with some version of the resting Puppitor action.

9.6 Discussion

By combining the evaluation methods of AAR and ERC into ACA, I can gain far more insight into the behavior and decision making process of algorithms than I otherwise would with either method alone. AAR is useful for high level analysis, describing the extent to which a domain constrains algorithmic decision making. However, AAR tends to flatten the actions and state when it is the only form of analysis available. An ERC, in contrast, is useful for low level analysis, exposing the effects of actions on state changes. That said, ERCs over-emphasize state transitions when used alone. Performing ACA requires both high and low level comparisons between algorithmic behavior. Thus both AAR and ERCs provide a balanced set of tools for analysis.

The ERCs in Table 9.3 show a highly constrained part of the Chiara ruleset to the point that all three algorithms performed identically. While a highly likely outcome, it is not guaranteed as Table 9.4 shows. Even under extreme constraints, a stochastic algorithm such as MCTS is not guaranteed to always choose the same actions. Depending on a projects goals, this may or may not be desirable behavior. It does, however, highlight my reason for making within algorithm comparisons a necessary part of ACA.

The goal of ACA is to provide designers with more information about the effects of technical choices on aesthetic elements. I therefore do not make any blanket recommendations about which algorithm is universally better. Design goals and player experience goals are not standard across games, nor even necessarily within games. As a

result, the best algorithm to fulfill a particular development goal may not be consistent with the requirements of other goals. As I take my domain and set of algorithms from *Tracks in Snow*, it serves as a central example of how to use ACA to support informed design choices.

9.6.1 Greedy Search Characterization

Greedy search always provides the most stable ERCs out of my test set. It chooses a single move to make until it notices that move will not allow it to reach its goal. This characteristic is most noticeable in the Table 9.1. The search performs projected energy, tempo up as soon as it is able, continuing until that move is no longer capable of making progress to its goal of expressing joy, then switching to the open, tempo down move. While greedy search's choices aren't the most interesting or dynamic, its consistency allows a player to more quickly understand the search's goal, one way a character can express a particular affect, and ultimately the shape of a character definition. Thus, greedy search is useful when the goal is for a character to behave in a consistent and readable manner. In *Tracks in Snow*, this use case includes when a player is learning basic interactions or when they want to understand more about the underlying rules governing character behavior.

9.6.2 A* Search Characterization

A* search produces stable yet somewhat varied ERCs in my testing. It has rather high AAR when compared with greedy search, though will switch its actions

more frequently than the latter algorithm. This behavior is most obvious in Tables 9.1 and 9.2 where A* search completely diverges from greedy search and MCTS in its choice of moves. The variance in A* search's choice of moves enables the potential for nuanced or surprising character behaviors while still remaining a deterministic algorithm. The variance of A* search can obfuscate its goals to a degree and introduce ambiguity into the relationship between a human and AI character. Considering *Tracks in Snow*, A* search is useful when a player desires a degree of leeway in interpreting character behavior while still remaining interpretable.

9.6.3 MCTS Characterization

MCTS produces the most varied and divergent ERCs of my set of algorithms. When in an unconstrained parts of the domain, it has very low AAR with both A* search and greedy search. Additionally, MCTS is the algorithm most willing to change its chosen move. This behavior is especially apparent in Table 9.2, where MCTS switches to from open flow, neutral to open flow, tempo down and back in the span of three decision cycles. The lack of consistency in MCTS, even in highly constrained environments, enables it to make unexpected choices and move through parts of a domain that may otherwise not be obvious. Table 9.1 demonstrates this as MCTS is the only algorithm to express worry. When put in the context of an experience like *Tracks in Snow*, MCTS is capable of behaving in both uniquely sublime and incoherent ways when performing with a human player. As a result, MCTS is useful when wanting to give players unpredictable characters where repeating moments are rare if not impossible to achieve.

9.7 Conclusion

My description of ACA in this chapter combines work in automated playtesting and the quantitative analysis of systems to more fully expose the relationship between technical system implementation and their aesthetic qualities. Specifically, I combine the techniques of ERCs and AAR to provide a balanced method of analyzing the effects of algorithmic decision making on player visible state. I believe this form of analysis is important to enable designers to make informed technical choices in relation to their aesthetic and experience design goals.

In this chapter I propose the Aesthetic Comparison of Algorithms method, an approach to analyzing the aesthetic qualities of algorithm behavior. I demonstrate ACA in a case study using Puppitor rulesets and three search algorithms: greedy search, A* search, and MCTS. I then synthesized ERCs based on the output of these algorithms and calculated the AAR between each pair of algorithms as well as within MCTS trials.

The results of my testing demonstrate the importance of comparing algorithms used as part of interactive experiences. The three algorithms' behavior diverged frequently and only in highly constrained parts of the chosen domain did they ever reach perfect agreement. Even with A* search and greedy search commonly agreeing on actions more frequently than half the time, their divergent choice could dramatically effect the resulting ERCs. While these differences can be subtle, when driving character behavior or interaction with a human, these subtleties can be important parts of the perception of an AI character.

Additionally, I identified design goals supported by each algorithm I tested. My goal with ACA is to inform designer decision making, not provide blanket recommendations about the best implementation across all circumstances. Each algorithm tested has desirable technical and aesthetic qualities depending on the kind of player experience a designer may want. There is a place for simple, predictable behavior in interaction design as much as complex, chaotic behavior.

I showcase ACA using relatively simple search algorithms commonly found controlling characters and agents. I expected some degree of divergence across the three algorithms but the level of MCTS's divergence when in less constrained parts of the domain were surprising. Following from my results, I argue that if ACA provides insight about these searches that would otherwise be difficult to understand, it is even more pressing to perform this kind of analysis on more complex algorithms, such as evolutionary, reinforcement, or self play. The more complex an algorithm, the less likely its behavior can be inferred from its code or data structure. If designers are to use these increasingly complex systems to create interactive work, they must have tools to support their understanding of these systems' behavior.

Chapter 10

Authoring Computational Performance and Expression

When we make systems, they should be able to create at least one complete experience. This holds doubly true for interactive and expressive systems. These experiences don't have to be polished to commercial game level, but they should be something which can meaningfully exist beyond the confines of a proof of concept. If they are interactive, they should be something which non-researchers can access and experience.

Interactive narrative systems must be able to tell at least one story. If they cannot, then what purpose do they serve if the goal of such a system is to support storytelling experiences? Arguably very little if any. We learn about systems through their use. Arguably this is why my dissertation is focused on the same system and topic as my MS and MFA theses. Puppitor would not be what it is today without my time developing *Tracks in Snow*. Some of this comes from *Tracks* being a Ren'Py game and

thus testing my goal of making Puppitor a modular system. More importantly, the mix of narrative and design concerns that came with making art using the system forced me to iterate on the technical aspects of Puppitor far more than if I had kept everything in a purely engineering context.

One of the core goals I had with Puppitor was to be able to create characters which physically felt different to play, the same way fighting game characters do. The system must then be able to support at least two different archetypal characters in its rulesets. The first being a character whose gestures alone allow for transitions between expressions, that is, Rika's design. The second being a character who must make small adjustments to gestures to accomplish the same goal, that is, Chiara's design. I have discussed the technical design of the *Tracks in Snow* cast before [126, 128], but not the experience of authoring the characters with the system in detail.

This chapter focuses on my experience authoring both for Puppitor itself and the AI systems which drive its characters. I want to stress once again that Puppitor in its minimal form is not an AI system. It is a domain for AI to operate in. The public repositories contain examples of generic search algorithms which work out of the box with Puppitor, but they are not core to the system. If you who are reading this were so inclined, you could train a reinforcement learning agent to work with any of the rulesets I discuss in this dissertation. The same holds true of evolutionary or self-play learning algorithms. I chose search algorithms for both the projects I discuss in this chapter because of their relatively quick iteration time in development and their lighter resource usage in exchange for a higher authoring burden.

The first half of this chapter is dedicated to the characters of *Tracks in Snow*. *Tracks* is the interactive storytelling experience developed alongside Puppitor. As the first project I ever made using the system, its needs heavily influenced much of my approach to authoring both rulesets and the AI architecture. The second half is devoted to discussing working with Puppitor at scale in the MIMIC crowd simulator. Rather than the two characters in a relatively intimate setting of *Tracks*, MIMIC required dozens of agents to be created dynamically.

These two projects represent extremes when working with Puppitor. *Tracks in Snow* features the minimum number of characters possible but with large amounts of feedback for tracking and exposing a character's state, all wrapped in a tightly focused, narrative experience. MIMIC features upwards of 50 agents in its scenes, no narrative framing to speak of, and, importantly, few mechanisms for feedback. And Puppitor works in both contexts. Not without considerable effort on my part in either case. Creating *Tracks*, its interaction model, and integrating it all with Ren'Py, was an undertaking to say the least. The MIMIC simulator's significant complexity, scope, and purely AI interaction posed a challenge for adding the dynamism the system is capable of. And yet, Puppitor accomplished my goals with both projects admirably.

10.1 Authoring the Characters and AI of *Tracks in Snow*

Puppitor would not exist without *Tracks in Snow*. I have joked about the system beginning its life as a way for me to make a yuri visual novel in an engineering

department, but, with the hindsight of another five years of development, I say this quite seriously. Without *Tracks*, Puppitor would be neither as flexible nor capable as it currently is.

My initial designs for the system featured numerous baked in assumptions about things like how many actions a character would be allowed, the set of emotional affects, and the names for everything. A few traces of these originally very deep assumptions remain in the default settings when you initialize Puppitor. Most of them, thankfully, are gone, in no small part because of working on *Tracks*. Gone are the original fixed set of actions and modifiers where I forced you to use my Stanislavsky and Viewpoints states whether or not they meant anything to you. Gone is the fixed set of affects, modified from Descartes' passions [224]. In there place are customizable sets of actions, modifiers, and affects, exemplified in the below example of an affect entry:

```
"joy" : {
    "actions" : {
        "resting" : -0.001,
        "open_flow" : 0.007,
        "closed_flow" : -0.005,
        "projected_energy" : -0.0002
    },
    "modifiers" : {
        "tempo_up" : 1.4,
        "tempo_down" : 0.75,
        "neutral" : 1.0
    },
    "adjacent_affects" : {
        "love" : 0
    },
    "equilibrium_point" : 0.65
},
```

I quickly stepped back from my initial hard coded details. I realized that if

I was going to be making a domain specific language anyways, I should take a page from the Tracery context free grammar format [56] and make it as flexible as possible within its necessary structure. A Puppitor ruleset must have actions which update a state, modifiers which change the rate actions update, and a state of affects and corresponding values, as these represent the structure of the interaction model. The exact labels for each of these matter for the designer *using* the system, not the designer *of* the system.

10.1.1 Creating Character Definitions

I have talked in various places [126, 127, 130] about where the exact sets of actions, modifiers, and affects I use for *Tracks* came from. The actions and modifiers were a very literal way of bringing the work of Stanislavsky [158] and Bogart and Landau [29] into the computer. The actions were named for Stanislavsky's three states of energy, with the addition of *resting* to enable players to be able to return to an equilibrium point. The modifiers were named after one of the viewpoints of time and feature a neutral state to enable the sort of character I envisioned for Chiara. I modified the set of affects from Descartes' set of passions [224] from: wonder, love, hatred, desire, joy, sadness; to: joy, anger, worry, sadness, fear, love[1] which better matched the spectrum I wanted for *Tracks*' story.

My modifications to the set of affects would neither have happened at all nor be this exact set if I had not decided that the story I was telling was one ultimately

[1]This probably says a lot about my general proclivities as a writer.

of self and relationship healing. The slight bent towards what are popularly considered "negative" emotions reflects this thematic element. My characters, even in their brighter and lighter moments, still have their doubts and fears and traumas weighing on them and the rulesets, more than just through the set of affects needed to reflect this fact.

Authoring a Puppitor ruleset is unlike authoring for many other interactive narrative systems. There is of course the set of affects to consider, as this is the spectrum of emotional tones you will be working with. Even more importantly is the level of granularity these rulesets operate at. Puppitor, as I initially envisioned it at least[2], exists in the realm of the micro, of individual frames, much like its fighting game inspiration.

The micro level mode of interaction is why the update values for both Rika and Chiara are of such low magnitude. These values, are added to their corresponding parts of the affect vector at a rate of 60Hz in *Tracks in Snow*. Therefore, in order to make changing expressions take a noticeable amount of time and avoid too much flickering between them, I settled on values usually in the thousandths or ten thousandths. Assuming a 60Hz update loop, the values in the thousandths will be responsive to change fairly quickly, often around one or two seconds to fully transition between emotions, with extreme cases taking upwards of four seconds. Such casees include the case of expressing sadness from worry in Figure 8.5, which took approximately 4.5 seconds.

One of my authoring goals was to make Rika a more mercurial character than Chiara. This manifested as Rika's rulset featuring similar orders of magnitude across actions and affect entries. These values most frequently fell in the mid to low thou-

[2]Puppitor itself makes no assumptions about update speed, though all publicly available examples assume realtime update loops.

sandths. I included some dips into the ten-thousandths to make affects like fear sticky even if they weren't easy to permanently express, shown below:

```
"joy" : {
    "actions" : {
        "resting" : -0.001,
        "open_flow" : 0.007,
        "closed_flow" : -0.005,
        "projected_energy" : -0.0002
    },
    "modifiers" : {
        "tempo_up" : 1.4,
        "tempo_down" : 0.75,
        "neutral" : 1.0
    },
    "adjacent_affects" : {
        "love" : 0
    },
    "equilibrium_point" : 0.65
},
```

The exception to my approach to Rika's generally quick and responsive relationship between actions and expressions is fear. Unlike the other affects in her ruleset, fear exclusively uses values in the ten-thousandths place and thus never will outpace the updates of the other affects. This effectively means that fear is impossible to express for any length of time but when it pokes through the state changes, it will stick around for a noticeable length of time[3]. The result is the below affect entry:

```
"fear" : {
    "actions" : {
        "resting" : 0.0003,
        "open_flow" : -0.0002,
        "closed_flow" : 0.0005,
```

[3]Fear caused no small amount of trouble when implementing A* in *Tracks in Snow* as the near impossibility of having fear reach the state ceiling value first meant the search would always fail in finding a path to expressing the affect. This did however result in the concept of characters interpreting affects becoming more fully supported in the AI architecture

```
            "projected_energy" : -0.0004
        },
        "modifiers" : {
            "tempo_up" : 1.12,
            "tempo_down" : 1.07,
            "neutral" : 1.0
        },
        "adjacent_affects" : {
            "anger" : 90,
            "sadness" : 10
        },
        "equilibrium_point" : 0.45
    },
```

When compared to Rika's ruleset, Chiara's features much more variance between update values within an affect entry. The most common pattern I used was to designate a particular action as the primary expressor of a particular affect. This relationship manifested as the chosen action having the highest thousandths place value in the entry, especially when combined with a modifier. Additionally, I associated each affect with one of the modifier keys by making one modifier greater than 1.0 and the others 1.0 or less. All of this was to make playing Chiara a more deliberate and slightly slower paced affair than Rika, and an example is shown below:

```
    "love" : {
        "actions" : {
            "resting" : -0.00082,
            "open_flow" : 0.002,
            "closed_flow" : -0.00088,
            "projected_energy" : -0.0006
        },
        "modifiers" : {
            "tempo_up" : 0.8,
            "tempo_down" : 1.27,
            "neutral" : 1.0
        },
        "adjacent_affects" : {
```

```
            "joy" : 90,
            "worry" : 10
        },
        "equilibrium_point" : 0.28
    }
```

The exception to my pattern for authoring for Chiara is the entry for worry. I have described Chiara as needing to push her worries aside, and this entry combined with its relationship to the others is the crystallization of this concept. Each of the non-resting actions are paired with high magnitude update values and resting features the lowest magnitude value in the ruleset, dropping down to the hundred-thousandths place. Additionally, it features no modifiers of 1.0. Neutral has a generally high modifier of 1.4 meaning that if no modifiers are intentionally invoked, any action will quickly turn to expressing worry. The other two modifiers, however, feature the most notable instance of negative modifiers in all of *Tracks in Snow*. They are both the single greatest magnitude multiplier across both Rika and Chiara's rules at 1.87 each and importantly, are what allow the player to push Chiara's worries aside, even relatively quickly compared to the change in other affects:

```
    "worry" : {
        "actions" : {
            "resting" : 0.00006,
            "open_flow" : 0.003,
            "closed_flow" : 0.0024,
            "projected_energy" : 0.0027
        },
        "modifiers" : {
            "tempo_up" : -1.87,
            "tempo_down" : -1.87,
            "neutral" : 1.4
        },
        "adjacent_affects" : {
```

```
            "fear" : 0,
            "sadness" : 0
        },
        "equilibrium_point" : 0.77
    },
```

Authoring a character with Puppitor requires a broad perspective in the ruleset's relationship to the larger software context it integrates with. A character ruleset must be responsive enough so players know their input is in fact being listened to but still maintain a degree of friction and stickiness to force a degree of commitment in action choice. The magnitude of update values is hugely dependant on the technical choices of the larger experience. My rules of thumb for authoring in *Tracks in Snow*, therefore, will not necessarily apply to projects with different technical or aesthetic priorities.

Additionally, as important as the individual affect entries are for defining the way a character expresses themselves, they must act in concert together. The most common pattern I found myself using was having some strong association between a particular action and affect. This decision is most obvious in Chiara's ruleset but still holds true for Rika's, where joy is most associated with the open flow action. I would then break this pattern to add character flourishes. In Rika's case this was the omnipresence of fear under the surface of all her expressions that would only occasionally poke through. For Chiara, this was the overriding nature of worry which must be actively worked against.

There are more possibilities for character styles and implementations than I

found for *Tracks in Snow*. I need to design more fully realized characters with the system to find them. Without the specificity of characters, different rulesets can become an exercise in mathematical differences without the goal of capturing the particulars of how a character should feel to play and play off of.

10.1.2 Developing AI for Puppitor

The AI architecture within *Tracks in Snow* is relatively simple to offload as much of the authoring burden onto character rulesets as possible. Changing which action and modifier a character primarily uses to express a particular affect of course can create a completely different kind of performance. Additionally, with the level of feedback present, both of the immediate and delayed varieties, even small changes to how quickly a character can transition between states becomes very noticeable. These qualities ultimately mean that the AI's architecture's primary role is goal selection and pathfinding, as Puppitor handles all the specific character details.

I chose to use search algorithms as the base of the architecture in *Tracks in Snow* because their relative simplicity of implementation allows for more time iterating on the rules and game feel. My goal with the AI architecture was to make something that required little maintenance even as rulesets were dramatically altered and would allow for changes to be immediate rather than needing the additional step of training. The result is a high level architecture with no specific domain knowledge and search algorithm implementations with minimal internal domain knowledge.

The general flow through the AI's expressive process is as follows. First, the

AI determines whether it is paying attention to the player's performance or the scene. Next the AI consults the adjacency list of the affect expressed by the current subject of its attention. The selected affect is now the AI's expressive goal and is then passed to an AI search function. The search function then returns the action and modifier to perform. Next, the action and modifier guide the update of the character's state, namely the affect vector and animation pose. Finally, the character's facial expression and portion of the background music react to the change in state.

The attention mechanism consists of two components: a reaction timer and state transition logic. The reaction timer forces the AI to commit to a single Puppitor move for a specified amount of time. In *Tracks* the standard time between reactions is 90 updates or a second and a half. The chosen time allows for the AI to still regularly make new acting choices and react to a player's performance without becoming inhumanly jittery or indecisive. The state transition logic is governed by a sin curve to move between paying attention to the player and making choices based on its interpretation of the scene. Approximately 30% of the time the AI interprets the scene on its own while the remainder of the time it pays attention to what the player expresses. This balance means that while the AI's acting is largely governed by the player's choices, it is not completely dependent on the player to give some life to a scene.

10.1.3 Adjacency Lists and Interpretation

The adjacency lists in Puppitor rule files (the `adjacent_affects` section) are a feature I initially added primarily as a hook for designers to do *something* with. What

exactly I wasn't sure, but it seemed useful to have a directed graph of associations between a character's affects. My initial implementation of adjacency lists in Puppitor was missing the weights but otherwise has been consistent over the course of the system and library's development. Within Puppitor, I use the adjacencies to give an extra layer of authorability to some specific states a character can be in. When a character switches which affect they are expressing, they first check if any of the affects they could choose are in the adjacency list, and if so, will choose that one. If there are multiple adjacent affects to choose from, the system chooses randomly. Once I added weights, a weighted random roll is used.

As a part of the character's ruleset, the adjacency lists describe the affects the character associates with the entry. In the case of Chiara's above worry entry, she associates worry with fear and sadness in equal weight. This ultimately means that when choosing a new affect to express, if she cannot continue to express worry, she will prefer to express fear or sadness when possible, with no preference between the latter two if both are an option. This behavior is the limit of how Puppitor uses adjacency lists internally, but as a way to describe a characters association of affects, they would prove a very useful component of a character's ruleset when interfacing with AI systems and other characters.

To interpret another character's expression or a scene, the AI performs a weighted random selection of the affects in the ruleset entry corresponding to the affect it receives. From a character, this affect is taken directly from that character's currently expressed affect. That affect is then used to look up the responding character's

adjacency list. The responding character then chooses a new affect as its goal from the adjacency list using a weighted random roll. When reading from a scene in *Tracks in Snow*, the process is the same but instead of the affect to respond to coming from another character, it is coming from the scene's tagged affect.

At a high level, adjacency lists are the way Puppitor expresses the relationships between affects specific to a character. This relationship is reflected in the directed graph of affects the adjacency lists create. The shift from using them as more than just a slight additional bit of authorability in the way a character switches affects to the basis for characters' ability to interpret actions came from Victor, Carnegie Mellon's Scrabble playing robot [49]. Victor has a gradient of emotions to move between depending on how the game was going to make for smoother, more believable reactions to the human player's actions. Importantly, Victor's mood is based on the direction the game takes, getting happier and more smug the better he is doing while conversely getting angrier and more combative the worse he is losing. While nominally a robot which plays scrabble, the systems behind Victor are about shaping the mood around a game of scrabble and interpreting both the game and his opponent's actions. In turn, the AI system interfacing with Puppitor and driving *Tracks in Snow* has ended up relying heavily on the adjacency lists to enable a similar approach to interpretation.

10.1.4 Authoring Limitations

While Puppitor rulesets in general are flexible, there are some thorny problems they can run into. The most catastrophic of which is when creating a transient affect.

A transient affect is an affect designed to only appear for brief moments as a character performs and is not one they can consistently and permanently express. This design manifests as an affect which maintains a consistently high value relative to other affects in the domain, but is very slow to change. As a result it will rarely if ever reach the affect value ceiling before another affect in the domain does. Authoring an affect to be transient is a matter of making its update values a significantly lower magnitude than every other affect in the domain, and, potentially, not giving any action or modifier a strong (high magnitude value) association with a specific action or modifier. This design in turn makes it extremely difficult for an AI system to purposefully express a transient affect as there is no consistent expressive end point, the affect will almost always be superseded by another during a state change.

In *Tracks in Snow*, Rika's ruleset features fear as a transient affect. When a human plays her, the effect works great and gives her some unique flavor. When a non-pathfinding algorithm based AI, such as greedy search or MCTS, uses her ruleset, fear can be a bit less readable as a goal but will still work fine overall. When a pathfinding algorithm such as A* encounters a transient affect as its goal, it can crash the entire game if no precautions are in place.

The reason for this is because, in the current implementation of Puppitor in *Tracks in Snow*, the pathfinding algorithm will exhaust its search space looking for what is effectively an impossible outcome: consistently expressing a transient affect. My solution to this problem was to effectively make the Puppitor rulesets the only point of failure when authoring a character for *Tracks*. The adjacency lists provided a

buffer I could use between the affects a character received and their true expressive goal at any given time. As a result, fear (or any other transient affect) is not an affect in any of Rika's adjacency lists, meaning she effectively does not associate any other affect with it.

This bit of characterization works well for her but is not a very effective solution outside of *Tracks*. When working with Puppitor, transient affects are something every designer must consider. They are easier to work with when created intentionally, as I did with Rika's ruleset using self-imposed authoring rules, but can emerge from character definitions regardless. An unintentionally created transient affect will not always be obvious to a human or AI system at first glance. With this in mind, the first solution to the problem of transient affects is to make the AI system itself more robust. Constraining its search resources more so even exhausting them does not produce a noticeable pause in the simulation along with giving it a fallback for when no path is found. The second solution would be to make transient affects a first order component of a Puppitor ruleset rather than an emergent property.

Altering Puppitor rule file structure is a more involved process than making the AI system more robust and would break compatibility across versions. One possible addition to the rule file entry format is simply a boolean for flagging if the entry is for a transient affect. An issue first order support for transient affects is that they arise out of the entire character definition. When looking at Rika's fear entry in isolation, there is nothing about it alone that says it will be a transient affect. Two of the actions have positive update values and two have negative values. It only becomes clear that fear

is a transient affect when in relation to the other entries. As I pointed out in Section 10.1.1, Rika's fear entry updates at around one order of magnitude slower than the other affect entries. Because of the highly relative nature of these affect entries and transient affects being an emergent property of the rulesets themselves, a static analysis tool which highlights any potential transient affects may be a necessary addition to the existing development tools moving forward.

10.1.5 Last Thoughts on *Tracks in Snow*

My architectural decisions for *Tracks in Snow*'s usage of Puppitor focused on simplicity, flexibility, and the character rulesets. Some of these priorities grew out of the nature of *Tracks* as a highly constrained experience in many ways. There is no navigation, a single room, and only two characters, so a minimal decision making engine provides more than adequate acting performances. Additionally, as *Tracks* is the first project of any kind using Puppitor, I wanted the example integration to be as approachable as possible[4]. This priority was further reinforced as *Tracks* is a visual novel and built in Ren'Py, my ultimate hope being that the system could be used in more projects using the engine if anyone was so technically inclined.

None of what I discussed in this chapter so far would have been possible without building Puppitor in parallel to *Tracks in Snow*. Without the long term authoring problem the visual novel forced me to confront, I wouldn't have been able to as effectively confront my assumptions about the structure and details of the system. I wouldn't have

[4]Well as approachable as any even minorly complex AI code can really hope to be.

had to confront and work with the limitations of technical choices with my character designs. Not only would *Tracks in Snow* not exist in the unique way it does today, Puppitor would not be as flexible a substrate for creating expressiveness with AI systems. *Tracks* made it ready for prime time.

10.2 Scaling to Crowds in MIMIC

And by prime time I mean using Puppitor outside of its original context, such as in the MIMIC crowd simulator. MIMIC is, at its core, everything *Tracks in Snow* is not. It's big, it's impersonal, it's about navigating through physical space, and it has no story to speak of. The goal of adding Puppitor to MIMIC was to effectively add a toggleable chaos engine to an otherwise uniform simulation. In more serious terms, I wanted to enable agents to be responsive to each other, dynamic in how they reacted, and to have more personality. My goals dovetailed nicely with the project's goals of creating more realistic crowd simulations.

Now you may be wondering at this point, how does Puppitor add realism to something, nothing you have talked about in your entire dissertation has mentioned realism. Well that's the trick isn't it? Puppitor may be entirely unconcerned with simulating the inner workings of a human being, but it is a system focused on enabling action, reaction, and interpretation between characters. If we want to make a realistic simulation, we must first ask what we are simulating. In a crowd simulation, we are attempting to capture the way humans behave in groups. Physically.

This priority is why a system which does not attempt to simulate the intricacies of the human mind can still produce even semi-naturalistic feeling group behavior. In a crowd, we are less concerned with the way an agent thinks than how they move in relation to other agents. Puppitor is designed to be an added layer of dynamism and expressivity and that is its place in MIMIC: to give agents dynamism and reactions that capture something of the way humans move through physical space.

MIMIC's Puppitor integration uses the same core flow as *Tracks in Snow*. The attention system, interpretation system, and decision making engine are all the same. The primary differences are in how attention is focused and the way an agent expresses its choices. As there are many more possible ways of paying attention in a dynamic environment like a crowd, rather than simply alternating between a single agent and a scene tag, a MIMIC agent must decide who to pay attention to before handing information over to Puppitor. This process manifests as attention modes based on the agent's field of view. These modes focus on either: the most recent agent still in view, the least recent agent still in view, the nearest agent, the furthest agent, and the aggregate affect of all agents in view. While determining where an affect comes from is more complex, especially in the aggregate case, its handling is essentially the same as in *Tracks*. What all the AI decision making becomes, however, is not.

10.2.1 The Feedback Problem

The biggest problem with integrating Puppitor into MIMIC was exposing what the system was doing to a user. Puppitor is a system designed around tight integration

with feedback, as *Tracks in Snow* demonstrates. But what can you do when there are no animations to hook into? no music to dynamically mix? no facial expressions to swap between? First, evaluate what you have. In MIMIC, the potential avenues for feedback were incredibly sparse: walk speed and proxemic radii. Second, connect to what you can. Third, sometimes you have to make a new component.

Walk speed ultimately became the only viable source of feedback in the project, beyond the emoji hats and explosions I made for when agents' change what they are expressing. My original integration had walk speed governed by Puppitor's prevailing affect and the proxemics directly connected to the modifiers. These choices were usable as a proof of concept but presented a number of problems. Most important was that the Puppitor actions were entirely hidden. This decision flew in the face of everything I had done in *Tracks*. It meant that, while there was more dynamism than without the system, it was not in fact using the model I spent so long designing. Similarly, the modifiers being connected directly do updating agent proxemics was more of a grasping at straws for ways to visualize choices than anything that made logical sense or would produce a naturalistic result.

So I refactored these connections to better reflect the philosophy I'd developed through *Tracks in Snow* and have a better theoretical foundation to work from. Actions now directly governed walk speed and modifiers were effectively hidden from view, as the proxemics needed to be responsive to crowd density specifically. The agents' expressions were still exclusively visible via the emoji hats and explosions. While positive changes to the agents' model of expression, none of these changes solve the fundamental problem

of feedback.

This impasse exists primarily because of the mismatch in abstraction between Puppitor and MIMIC. Puppitor is a micro level system where individual frames matter and are felt. MIMIC is a high level, extremely course style of interaction. Ultimately, this led me to abandoning making highly authored rulesets for the project entirely. I mostly iterated on the domain labels rather than the details the way I did for *Tracks in Snow*. A microsecond difference in an agent's reaction in MIMIC does not matter in the same way. The amount of feedback available makes sure of that. So this problem begs the question, where do we go? What can be done with Puppitor in such an environment?

10.2.2 The AI Takes Center Stage

I have stressed in various places that Puppitor is not an AI system itself. It is a domain, a substrate for AI to perform characters and interpret actions. To solve the problem of Puppitor rulesets being unable to quickly and easily alter behavior in MIMIC as they do in *Tracks in Snow*, I had to move an abstraction layer up to the AI system itself.

The architecture I imported from *Tracks* has a number of elements which can dramatically change the way an agent behaves, but were unneeded in that environment. In particular, the speed an agent switches between paying attention to its surroundings and its internal default, replacing scene tags (represented by `curve_delta` below), the amount of time an agent will pay attention (represented by `default_weight` below), and the frequency an agent could update its action (represented by `react_alarm` below)

were the standout parameters.

```
public class JSONProfile
{
    // general options
    public string rule_file_path;
    public string default_action;
    public string default_modifier;
    public string default_affect;
    public int look_mode;
    public double curve_delta;
    public double default_weight;
    public int ai_mode;
    public int react_alarm;

    // speed and proxemic options
    public Dictionary<string, float> walk_speeds;
    public List<float> proxemic_multipliers;

    // AI algorithm options
    public int step_multiplier;
    public int mcts_iterations;
    public int mcts_rollout_depth;
}
```

The first agent personality I designed with the AI system rather than Puppitor ruleset was a jerk. The simplest thing to do with any new set of parameters is to slide them all the way in one direction to see what happens. Also, making a jerk is pretty flavorful when everyone else is pretty considerate by default. So, how do you make a jerk in MIMIC? The first thing to do is make them only perform the default action, so the default weight must be 1.0. For safety we can also set the update to the sin curve and react alarm to 0 as a triple buffer to make sure this agent is *always* a jerk. We can also set their walk speeds for any action they do to be effectively power walking, and of course, set their default affect to be angry. Thus, a jerk is born.

This jerk profile set the stage for the authoring pattern for agent expression in MIMIC, that is, the expression profile. Expression profiles are much smaller and higher level than Puppitor rulesets. They mostly contain the parameters important to controlling an agent's attention. They also allow for the direct authoring of the effect Puppitor actions have on walk speeds, the reactivity of proxemics to crowd density, and the resource usage of Monte Carlo Tree Search (MCTS) [47].

Codifying the AI parameters into small, human-readable and authorable files came with a number of benefits. Expression profiles allow AI personalities to be edited at runtime and are portable across agents, enabling a designer to rapidly iterate the behavioral makeup of crowds. Most importantly, the existence of expression profiles supports the generation of agent personalities.

My implementation of expression profiles allows for hand authoring, similar to Puppitor rulesets. With that said, it is rather impractical to author dozens or even hundreds of expression profiles for a crowd of agents. At such a scale, an individual agent rarely matters in a general case. There of course exist scenarios where an individual agent is hugely important, such as the effect of one jerk on a crowd, to continue with my original experimentation. For the average agent, however, a randomly generated profile will often be sufficient.

The simplicity of the expression profiles thus enables a lightweight and configurable profile generator to be created. As implemented in MIMIC, the profile generator creates arbitrary numbers of expression profiles based on a combination of constant and user specified constraints. Most parameters use a simple, uniform random roll with

Algorithm 4 Random Gradient

Input: Empty list
Parameters: *gradient*
Output: List of floating point values in descending order

```
Let grad_step = 1.0
while grad_step > 0 do
    grad_step = Random(floor, ceiling)
    if grad_step > 0 then
        gradient ← grad_step
    end if
end while
return gradient
```

floor and ceiling values. Most of the corresponding floor and ceiling values have some constant limit to enforce a minimum amount of functionality and avoid users too easily creating completely broken profiles. The parameters controlling AI resources have the tightest constraints as, particularly with MCTS, computational resource usage can quickly get out of hand and crash the entire simulation or can become so minimal the algorithms cannot properly operate.

The exceptions to the uniform random rolls are the walk speeds dictionary and the proxemic multipliers. Walk speeds use a normal distribution instead to capture the natural variation of humans. The proxemic multipliers require a more involved process as they must create a gradient rather than single values. A random value is subtracted from a starting value of 1.0 and appended to the proxemic multipliers list, detailed in Algorithm 4.

Having a generator thus enables unique agents to be dynamically created at runtime with minimal amounts of authoring unless specificity is desired. The minimal possible extent of the authoring required, thanks to the profile generator, is simply

loading a scene in MIMIC and running it if no changes to the agent domain are desired. Otherwise, the generator's configuration file is where I have ended up spending most of my time in the latter stages of my time with the project, the structure of which is shown below:

```
public class Profile_Generator_Config
{
    public string base_file_name;
    public string rule_file_name;
    public float curve_delta_ceil;
    public int react_alarm_ceil;
    public float walk_speed_floor;
    public float walk_speed_ceil;
    public float prox_ceil;
    public int num_generate;
    public int seed_value;
    public bool use_seed;
}
```

The configuration file primarily determines how much variation between the agents is possible, as it mostly contains floor and ceiling specifications. It also allows the specification of the file name for any generated profiles, the Puppitor ruleset to serve as the basis for the profile, the number of files to generate, and seed parameters if uniformity is desired.

An important subtlety with the expression profile generator in MIMIC is that the configuration file is universal. Both the agents themselves, if told to generate their own profile, and manual generation from the simulator's interface, use it constrain their parameters. I chose to structure things this way as my primary mode of interacting with the generator was through the dynamically created agents, rather than generating profiles and connecting them to agents after the fact. Additionally, while generating

profiles outside of agents can be instructive in MIMIC, the interface for loading profiles into agents at runtime is primarily designed for iterating on a single agent rather than at the entire scene level.

MIMIC in some ways is a less than ideal environment for Puppitor, as it does not feature many avenues of feedback for the system to connect to. Additionally, its authoring focus is on scale rather than intimacy. These limitations are why I ended up focusing more on formalizing the AI parameters rather than the rulesets themselves. The AI system provides a more immediately responsive set of parameters for authoring agent behavior in MIMC than Puppitor rulesets. Additionally, it enables the generation of agent personalities, which further reduces the end user authoring burden if variation and not specificity is the use case.

10.3 Evaluation in the Authoring Process

I have used the tools I describe in Chapters 8 and 9 as part of authoring with Puppitor. The original purpose of the utility which would become the basis for the Aesthetic Comparison of Algorithms was to compare A* search implementations across Python and C# as part of developing behavior for MIMIC. That utility inadvertently exposed some flaws in the way A* prioritized actions when it searched. It would generally try to take the most circuitous route towards its goal, choosing the action which would change the desirable state values the least. While this behavior itself is an interesting way to perform, it made interpreting the algorithm's intention extremely difficult

when just looking at a *Tracks in Snow* scene, as the first move in a path is the only one guaranteed to be seen. Further contributing to the lack of interpretability is the frequency with which the AI was able to change its goals. This design choice meant that much of an A* path would be thrown out, especially if the player changes their behavior.

The Expressive Response Curves themselves helped me to shift A* towards more immediate results to thus feel more directly responsive and predictable, while not being as predictable as something like greedy search. Even before the changes to the priority calculations of A*, I made it the default AI option for *Tracks*, for its balance between predictability and chaos. Predictability is important for helping players learn about both the systems and domain, while a sprinkling of chaos can lead to some unique and sublime moments of acting. The Aesthetic Comparison of Algorithms confirmed this choice years ago in my characterization of the three search methods available in *Tracks*. Additionally, thanks to the comparison and characterization in Chapter 9, I feel it is possible to describe the options to a lay person as part of the options menu in *Tracks*. Currently the options are labeled simply with the algorithm names and no other information. I plan to re-label each of them based on the predictability of the acting they offer and add tooltips for additional information about what the options do and how many computational resources they require.

Exposing the internals of computational systems is vitally important to the authoring and development process. It is a process which helps compare the realities of an implementation to design goals and help to find mismatches between execution

and intention. Additionally, developing a better understanding of a system can enable a designer to better support their own design goals and player or end user experience. This support can manifest as altering the underlying system itself or being able to better explain what an available option is for. Ultimately, if the person building the system understands it, the people interacting with it on the other end are more likely to also be able to learn its contours.

10.4 System Limitations

As a model of physicality, Puppitor is a highly successful system, but it is one rooted in specificity. Authoring with the system requires fitting into its model of action and modifier conveying affect. Additionally, even though Puppitor is rooted in theatrical practices from acting to choreography to puppetry, it draws from highly specific sources to construct its model of performance. As a synthesis of Zeami, Stanislavsky, and Bogart and Landau, Puppitor meshes well with their metaphors and approaches to storytelling, particularly when gesture is involved.

Modeling other performance practices with Puppitor is not necessarily easy. When working on MIMIC, I briefly attempted to model Laban movement theory with Puppitor [5]. It only ever remained an initial attempt which didn't work very well as I had spent far less time studying Laban than I had with my main points of reference. While I have no doubt that modelling Laban movement with Puppitor is possible, it

[5]For at most a week which is really not enough time to create a quality Puppitor ruleset from scratch, especially when using a notion of physicality quite different than the ideas I spent years studying and working with.

is a different enough model from my synthesis of Zeami, Stanislavsky, and Bogart and Landau that the model isn't completely trivial to build. For any future designers who work with the system, this is something to keep in mind as well. If you are taken with a particular approach to expression and want to use Puppitor, it may or may not work as nicely as you might hope if the approach is distinct enough from my assumptions about the relationship between action and expression.

Finally, as Puppitor is a model of physical acting and expression, it is not suited to simulating deep cognitive processes or decision making. If an experience is heavily focused on the internality of a character without a connection to some element of character physicality, Puppitor is likely the wrong tool for the job. The system is potentially of capable of helping expose those other processes in conjunction with a system modelling cognition, but, by itself, it does not use the right models or paradigms for the task. This limitation brings me to a question I have yet to answer for myself: can Puppitor do Shakespeare?

Broadly speaking, yes, I do believe the system is capable of working with the majority of a Shakespeare script. But what about soliloquies? How would and AI with Puppitor handle playing Hamlet or Macbeth or Iago or Richard III? There are answers to this question which are relatively simple. One could mark up different parts of a soliloquy with different affects for Puppitor to interpret and thus be able to achieve some dynamism without the presence of another character, but these are highly internal moments, separating these characters from those around them and playing to the audience. I have not tried to use Puppitor with a soliloquy and doing so may enable

me to find a deeper answer to the question of the system's relationship to internality than what I have stated here.

10.5 Conclusion

There is not too much for me to say about future work here beyond "there needs to be more made with Puppitor" to further explore the bounds of the system's capabilities. In an ideal world, I will not be the only one making experiences using the system, but more work needs to be done to smooth out the authoring experience. The most major current issue is a lack of documentation. The public Puppitor repositories feature all the building blocks needed to use the system out of the box, including the core modules and search algorithms which work out of the box. There is, however, no example of how to construct an AI system in the repository. Currently the best and only publicly available example is in the released versions of *Tracks in Snow*. While it is better than nothing, the AI code is buried in an obtusely named text file and, though commented, is not an ideal example to learn from.

One of my now long time ambitions is to create a social stealth game reminiscent of the game *Spy Party* [50] and anime *Princess Principal* [250]. In *Spy Party*, a human player is tasked with blending in with a crowd of AI characters to avoid being assassinated. *Princess Principal* is a series devoted to espionage via social manipulation and blending in with different social classes its cast does not belong to. I see a lot of potential in allowing players to effectively code switch their character by changing the

active Puppitor ruleset to copy the mannerisms of whichever group of people they find themselves in.

Another longstanding concept of mine is a tactics game emphasizing group cohesion over the individual power of units. My time working with the MIMIC simulator further convinced me of the efficacy of the actions a character takes having wider consequences than a simple one off interaction. When watching a MIMIC scene long enough, it is possible to slowly learn who each agent is paying attention to based on when and how they react. A game focused on the attention mechanism I discussed in this chapter has the potential to create novel tactical situations and solutions that would otherwise be impossible to design without Puppitor.

Authoring two wildly different kinds of software with Puppitor has taught me an incredible amount about the system and the way it integrates into larger projects. The ability for the entire system along with its supporting AI architecture to slot neatly into both an intimate visual novel and a complex crowd simulation is a work of engineering I am immensely proud of. Puppitor is indeed a modular system and wonderfully accomplishes my design goal of allowing it to be a layer on top of existing software in addition to being able to drive experiences by itself. Authoring the rulesets does have its limitations, largely due to how much feedback their design expects. Without outlets to connect with, Puppitor itself cannot expose enough of its state to make the low level authoring have a meaningful impact. The AI architecture I designed for both *Tracks in Snow* and MIMIC, however, overcomes this limitation through its small set of attention control parameters.

When I started work on Puppitor in 2018, I wanted to create a little acting system which could let players perform a character in a new way. I did not envision it becoming something as robust and flexible as it is today. All my time spent refining the system through the development of *Tracks in Snow* and working with its limitations in MIMIC has made it something I want to inspire to find ways of using it I would never dream of and surpass my accomplishments here.

Chapter 11

Conclusion

Recalling the research questions laid out in Chapter 1:

- **RQ1:** How is theatrical storytelling applicable to computation and narrative design?
- **RQ2:** How do we translate theatrical practice into a computational model of character acting?
- **RQ3:** How do we create a narrative experience using an acting system as the interaction paradigm?
- **RQ4:** How do we computationally analyze and expose a system's acting choices and its effects on the performance of a character in an interactive narrative context?

The high-level structure of my dissertation answers these questions in order: Chapters 3, 4, and 5 provide the answer to RQ1 using the lens of theatricality; Chapters

6 and 7 answer RQ2 and RQ3 via discussions of narrative and system design respectively; and Chapters 8, 9, and 10 provide answers to RQ4 with discussions of evaluation methods designed to expose algorithmic decision making and descriptions of the authoring experience using the Puppitor system.

Chapters 1 introduces computational theater as an approach to working with interactive narratives by drawing inspiration from theater and highlights its emphasis on playful interaction with stories. Chapter 2 frames my work and computational theater's relationship to existing work in computation also taking inspiration from theater, focused on existing work primarily drawing from theater *criticism*. Computational theater, by contrast, draws from theatrical practice, primarily acting and puppetry. Through this re-examination of sources of inspiration, I set the stage to discuss my approaches to narrative and system design that make up my initial foray into computational theater.

Chapters 3, 4, and 5 serve as the theoretical foundation for my work, further positioning computational theater and my research in conversation with existing commercial games and their approaches to playing with stories. Chapter 3 lays out the foundation of my definition of theatricality, that it is a property of experiences which continuously reinterpret themselves, and apply this lens to the narrative roguelike *Hades* to demonstrate the efficacy of the theory. I then use the lens of theatricality, in conjunction with story volumes, on *Umineko no Naku Koro ni*, a linear mystery visual novel, to further test the bounds of my understanding of theatricality and highlight that systems are an optional component of these kinds of experiences. Chapter 5 is

then the expanding of theatricality to explicitly multiplayer experiences and a reminder that the interpretive and interactive nature of this theatricality is often a communal act. The thread running through each of these chapters is again, the playful interaction with stories. Each of these chapters serves as a reminder that playful interaction does not require any particular form or technology or tool. Dynamic structure, even systems in their entirety are optional. There is freedom here, that playing with stories can take on all manner of forms and it behooves us to embrace that and choose our designs with intention.

Chapters 6 and 7 tackle what it means to design with theatricality and theatrical performance in mind, to ultimately craft playful interaction with a story. In Chapter 6, I discuss ways narrative design invites play, looking at both how the low level details impact a player's influence over the story and the importance of high level design for giving the player a dynamic relationship to a story. Chapter 7 contains a discussion of the system design of Puppitor, describing its origins in theatrical practice, the translation into a computational model by way of fighting games, and the ways the system enables playful interactions between characters and, ultimately, the story of *Tracks in Snow*. Both these chapters, Chapter 7 in particular, are my chosen tools for facilitating playful interaction with narrative. The narrative and system design in each is an embodiment of my own philosophies of storytelling and system design, as well as my desire to expose people to the joys of playing *with* a story. Computational acting games didn't exist when I started the work that would become this dissertation. As a result I had to build my tools from scratch. They turned out to be extremely powerful

and I hope to inspire people to play with Puppitor and design with it or even better, use it as inspiration for their own visions of playing with stories.

Chapters 8, 9, and 10 discuss the aesthetic qualities of systems which facilitate playful interaction as well as the ergonomics of working with Puppitor. I lay out the preliminary evaluation approach of expressive response curves in Chapter 8, which focuses on the need to understand the relationship between AI domain and end player experience. Building from there, I describe the aesthetic comparison of algorithms approach in Chapter 9 which seeks to support the making of informed decisions about AI system usage and integration by enabling standardized comparisons between different AI systems within a chosen domain. Finally, Chapter 10 describes what it is like to design characters using Puppitor, in particular, the ways different project needs impact whether authoring focuses on the domain or AI system. These last three chapters are my way of discussing how to choose the correct tool to best serve design goals. The more complex the system, the more opaque the relationship between its decisions and the ultimate player experience. Intention behind design decisions cannot guarantee perfect experiences driven by playful interaction, but understanding the interlocking systems beneath the surface is crucial in iterating on these types of projects. There is power in playing with the details of a story. Such an emphasis on low-level details thus leads to the minutiae of system implementation having an even greater effect on the overall player experience.

11.1 Broader Impacts and Looking Forward

I structure this dissertation around what is effectively an example of the design loop of creating a computational theater system: choosing real world practices and existing computational designs to draw from; constructing systems based on combining these two elements; and, finally, exposing the system's aesthetic and ergonomic qualities to support design iteration. Each step exists to further inform system design. In this dissertation's particular case, a theatrical AI system.

Through the design and engineering process, we necessarily import assumptions from prior work. My goal with this section, and the dissertation as a whole, is to support designers and engineers in making the import of those assumptions a conscious choice. If there is one parallel between art and engineering to highlight, it is this: working with intention is paramount.

When developing with AI in any capacity, it is vital to understand the aesthetic implications of building or integrating such a system into a larger piece of software, especially when directly touching human lives. The first question asked should be whether AI is a desirable addition at all. Is enabling the system or computer to perform tasks autonomously or guess user intentions, among other uses, necessary to the project's goals? When designing a social simulation, or for that matter many other kinds of simulation, the answer is often yes. Simulating human behavior truthfully frequently requires the creation of autonomous agents who can dynamically react to changing circumstances and environments.

Now, the question turns to understanding the aesthetic differences between different approaches to developing AI. The problem here, as I discussed in Chapters 8 and 9, is that there is effectively no universally correct answer. Finding a correct answer depends on resource constrains, workflow preferences, domain representation, and more. For a satisfactory answer to be found, the technical concerns are of course important, but ignoring the aesthetic side of AI is fraught. The algorithms, structures, and interpretive processes composing any given AI system considerably impact the surface level behavior and user experience.

To then understand the relationship between AI system and the aesthetics of its behavior or interaction, designers and engineers must be able to expose the underlying rationale and interaction with the broader software environment. Simple domains combined with simple systems allow the reality of a system's behavior to, more often than not, align with human intuition. As either the domain or system increases in complexity, especially when both increase together, human intuition will decrease in accuracy or simply fail all together.

Exposing the relationship between algorithm and surface level experience is also important when considering high level design goals. A complex system may in fact be the correct choice for something like a social or crowd simulation project, but even that is not a guarantee. Depending on the exact requirements, a simple solution may produce more useful results. One could even argue that my authoring discussion in the latter half of Chapter 10 fits the description of a relatively simple solution to the problem of individuality in crowds. Thus, ideally, when creating AI systems, multiple approaches

to design and complexity should be tested to find the best fit with development goals. After all, one of the reasons to work with AI is the unpredictability, and the results of such testing may be more surprising than expected.

Building tools to surface these systems underlying logic can thus help bring intentionality back to development decisions. I describe one such approach in Chapter 9. While the example of Aesthetic Comparison of Algorithms uses well known search algorithms, it provides important foundational understanding for AI development more broadly going forward: that algorithmic decision making directly impacts aesthetic experiences. The more subtle contribution of this whole dissertation is that we should be explicitly constructing our problem domains with aesthetics in mind.

AI systems are not independent of their domains. Even if they are modular and do not technically contain any specific knowledge about the problem space, their behavior and output is none the less shaped by what they operate over. Puppitor was such a useful testbed for the evaluations I describe in this dissertation because of its domain architecture. It creates a direct connection between the system's underlying state, choice of actions, and surface level state. This flow is a property of the domain itself and something I hope to inspire more of in future AI problem domain development. AI systems will inevitably run into aesthetics when they interact with or simulate humans in any way. If we build this property into our development practices holistically, we can design systems with intention, integrate them with larger software contexts keeping user experience in mind, and choose the right system to support project goals.

Simulation of any kind is an abstraction. Looking at the MIMIC simulator

specifically, its integration with Puppitor worked shockingly well, far surpassing my expectations. This is of course a win for my little system, but it is also an example of the applicability of acting theories and computational theater to a variety of applications beyond interactive storytelling. Theater deals in mimesis. Computational theater does too. Important to this concept is that the interior does not have to match the exterior. Mimicry thus, is also an abstraction. An abstracion relatively unconcerned with the underlying processes being an exact replica of its subject. Backstage is functional and efficient with no care for aesthetics so that what happens on stage may flourish. Actor drapes themselves in characters so the audience may believe what they see. Puppitor emphasizes intent and emotion so a character may perform regardless of the presence of a narrative.

Mimicry as an approach to design allows systems to function much the same way as a stage. The underlying components of a system exist to support specific behaviors, interactions, or aesthetics and need not be perfect replicas of other processes. The evaluations I perform in this dissertation reflect this philosophy. They exist to help probe at if a particular system or implementation is indeed the best way to support the aesthetic goals of a larger project. Even when I discuss the importance of focusing on underlying processes, it is because they can take many forms that they must be treated with such care. It is also not a failure to have a simple approach be the best option for a project. If an implementation meets or surpasses a project's goals, that is a success, regardless of underlying complexity.

Computational theater, and more specifically Puppitor have been applied to

crowd simulations as I described in Chapter 10, but this implementation and usage is fairly minimal both in scope and in application. There are social elements to the crowd simulation in MIMIC but there is little detail beyond the broad spacial relationships and movement speed of agents as part of the simulation itself. Puppitor makes these simulation elements dynamic and ultimately chaotic, but there is little added to the model of social interaction itself. A deeper integration with a social model extending beyond the spatial, while still emphasizing physicality would enable Puppitor, or any performance focused computational theater system, to add more nuance and variability to interactions between social agents.

A theatrical and performance focused system can add a layer of interpretability for developers and users of a social simulation. On the developer side, such a system enables agents to dynamically react to each others' physical behavior, potentially providing another avenue for creating systemic relationships between agents. For a user, having agents with visibly distinct personalities and resulting behavior can greatly increase the believability of a simulation when compared to the real world. After all, humans are individuals and rarely behave in uniform or consistently logical ways.

When considering social models which a theatrical, performance focused approach would work well with, Erving Goffman's social dramaturgy [88] and the lineage of presentation and performance focused sociology is immediately relevant. Yet as I have described and implemented a performance focused computational system in this dissertation, as it exists now, the system can only simulate a portion of these models of social interaction. Missing is the internal, cognitive portion of this model. While this

type of computational modeling is outside the scope of my dissertation, having an internal model of the self is vital to developing simulations based on this line of sociology. Furthermore, such a model when used in conjunction with a model of physicality could potentially create highly nuanced interactions and relationships between agents which either model alone could not hope to achieve.

Especially when thinking of integrating even more complexity in the systems and relationships they interface with, a significant amount of tooling is required to make the development of such an intricate social simulation possible. Furthermore, using such a simulation to gain insights about human interaction would be nigh impossible without a significant amount of ability to sift through a simulation itself. One such tool is a visualization of all the factors an agent is using to choose their next action. Currently, understanding the reasons behind any individual agent's behavior in a social simulation using a physicality focused system like Puppitor is immensely difficult if there are more than two or three agents in a scene. This problem arises from the fact that it is agents responding to agents all the way down. So at minimum, being able to chase down the paths of agents expressing things to other agents would make any individual's behavior significantly more intelligible on both the development side and the user side of the simulation.

My goal with computational theater is to demonstrate an approach to system design which cares deeply about developing with intention. A large component of intentionality is understanding the relationship between components of a process. While I haven't framed my work as explainable AI, I increasingly see my work growing in

that direction with my fascination for low-level systemic interactions. My hope then, is to highlight aesthetics as a necessary component of explainable AI development and research.

11.2 Final Thoughts

To come full circle, my work in this dissertation can be seen, even more than the birth place of computational theater, as a call to design playful interactions with narrative and to choose the correct tools to fulfill the task at hand. My use of computational theater here describes my process for both designing ways of playing with stories and making informed technical choices to support that form of play. Embracing rather than ignoring, or worse fighting, play within stories allows them to be in conversation with players, to meet them where they are and incorporate their ideas, if not into a singular canon, then into the version meaningful to them. I see it as vitally important for interactive narrative to embrace the playful interaction with stories, whether directly from computational theater or elsewhere, to continue to evolve. Interaction is more than shoving a structure around to achieve a desirable plot outcome. Interaction is playful. Interaction is interpretive. Interaction exists whether we design for it or not.

My greatest hope for my work and this dissertation is that it inspires my fellow designers and researchers to re-examine their assumptions about storytelling and technology. A narrative is an interactive system regardless of any computational components just as a computational system is storytelling regardless of the presence of text. There

is a cyclical relationship between storytelling and technology when combined together, be they works of computational theater or something beyond our current imagination. Because of these interconnections, we need to choose our tools, our technologies, and stories with intent. We must think through the effects a particular technology will have on our storytelling and the way our stories frame the technology supporting them. My call here may sound daunting, but it is also playful. Re-examination, reinterpretation, fresh perspectives, all these things can be used to play with stories, interfaces, and yes, research too. The last seven years of research and design work in this dissertation would not exist without revisiting my ideas, re-examining, twisting them, sharing them, bending them, breaking them, and, yes, playing with them. I leave you here with this final thought: **to create interactive work, we must invite play and the wealth of connections brought with it.**

Bibliography

[1] 07th Expansion. Umineko no Naku Koro ni, 2008. English release in 2016.

[2] 07th Expansion. Umineko no Naku Koro ni Chiru, 2010. English release in 2017.

[3] Espen Aarseth, Solveig Marie Smedstad, and Lise Sunnanå. A multidimensional typology of games. In *DiGRA Conference*, 2003.

[4] Robert Ackerman. *The myth and ritual school: JG Frazer and the Cambridge ritualists*, volume 13. Psychology Press, 2002.

[5] Shivam Agarwal, Christian Herrmann, Günter Wallner, and Fabian Beck. Visualizing ai playtesting data of 2d side-scrolling games. In *2020 IEEE Conference on Games (CoG)*, pages 572–575. IEEE, 2020.

[6] Arc System Works. Guilty Gear -Strive-, 2021.

[7] Sinan Ariyurek, Elif Surer, and Aysu Betin-Can. Playtesting: What is beyond personas. *IEEE Transactions on Games*, 2022.

[8] UCSC Theater Arts. The barn, 2020. accessed 4/23/2024.

[9] UCSC Theater Arts. Barnstorm, 2020. accessed 4/23/2024.

[10] Vivian Asimos. Playing the myth: Video games as contemporary mythology. *Implicit Religion*, 21(1), 2018.

[11] Ayatsuji Yukito. *The Decagon House murders*. Pushkin Vertigo, London, 2020 - 2017.

[12] Ruth Aylett. Narrative in virtual environments-towards emergent narrative. In *Proceedings of the AAAI fall symposium on narrative intelligence*, pages 83–86, 1999.

[13] Batu Aytemiz, Nick Junius, and Nathan Altice. Exploring how changes in game systems generate meaning. In *DiGRA Conference*, 2019.

[14] Byung-Chull Bae and R Michael Young. A use of flashback and foreshadowing for surprise arousal in narrative using a plan-based approach. In *Joint international conference on interactive digital storytelling*, pages 156–167. Springer, 2008.

[15] Vincent Baker. The Fruitful Void, 2005.

[16] Bandai Namco. Tekken 7, 2015.

[17] Roland Barthes. Mythologies. 1957. *Trans. Annette Lavers. New York: Hill and Wang*, pages 302–06, 1972.

[18] Matthew Barthet, Ahmed Khalifa, Antonios Liapis, and Georgios Yannakakis. Generative personas that behave and experience like humans. In *Proceedings of*

the 17th International Conference on the Foundations of Digital Games, pages 1–10, 2022.

[19] Eric Bass. Visual dramaturgy: Some thoughts for puppet theatre-makers. *The Routledge Companion to Puppetry and Material Performance*, pages 54–60, 2014.

[20] Sarah Bay-Cheng. Theatre squared: theatre history in the age of media. *Theatre Topics*, 17(1):37–50, 2007.

[21] Sarah Bay-Cheng. Digital performance and its discontents (or, problems of presence in pandemic performance). *Theatre Research International*, 48(1):9–23, 2023.

[22] Be-Papas. Revolutionary girl utena, 1997.

[23] Tilde Bekker, Linda De Valk, and Berry Eggen. A toolkit for designing playful interactions: The four lenses of play. *Journal of Ambient Intelligence and Smart Environments*, 6(3):263–276, 2014.

[24] Tilde Bekker, Janienke Sturm, and Berry Eggen. Designing playful interactions for social interaction and physical play. *Personal and Ubiquitous Computing*, 14:385–396, 2010.

[25] Ingmar Bergman. The seventh seal, 1957.

[26] James Bierman. Aristotle or Else.

[27] Black Isle Studios. Planescape: Torment, 1999.

[28] Anne Bogart and Jackson Gay. The art of collaboration: On dramaturgy and directing. In *The Routledge Companion to Dramaturgy*, pages 213–216. Routledge, 2014.

[29] Anne Bogart and Tina Landau. *The Viewpoints Book: A Practical Guide to Viewpoints and Composition*. Theatre Communications Group, 2005.

[30] Ian. Bogost. *Persuasive Games: The Expressive Power of Videogames*. MIT Press, Cambridge, MA, 2007.

[31] Blai Bonet and Héctor Geffner. Planning as heuristic search. *Artificial Intelligence*, 129(1-2):5–33, 2001.

[32] Jorge Luis Borges. *Fictions*. Calderbook ; CB428. Calder Publications, Paris ;, 1998.

[33] Bertolt Brecht. *Brecht on Theatre*. Bloomsbury Publishing, 2014.

[34] Kate Brehm. Movement is consciousness. *The Routledge Companion to Puppetry and Material Performance*, pages 84–90, 2014.

[35] Bungie and Activision. Destiny 2, 2017. As of 2019, Activision is not affiliated with Destiny.

[36] Joseph Campbell. *Pathways to bliss: Mythology and personal transformation*, volume 16. New World Library, 2004.

[37] Joseph Campbell. *The hero with a thousand faces*, volume 17. New World Library, 2008.

[38] Capcom. Street fighter ii: The world warrior, 1991.

[39] Capcom. Street Fighter IV, 2008.

[40] Rogelio E Cardona-Rivera, Bradley A Cassell, Stephen G Ware, and R Michael Young. Indexter: A computational model of the event-indexing situation model for characterizing narratives. In *Proceedings of the 3rd Workshop on Computational Models of Narrative*, pages 34–43, 2012.

[41] Rogelio E Cardona-Rivera, José P Zagal, and Michael S Debus. Narrative goals in games: A novel nexus of story and gameplay. In *International Conference on the Foundations of Digital Games*, 2020.

[42] Elin Carstensdottir, Erica Kleinman, and Magy Seif El-Nasr. Player interaction in narrative games: Structure and narrative progression mechanics. In *Proceedings of the 14th International Conference on the Foundations of Digital Games*, FDG '19, New York, NY, USA, 2019. Association for Computing Machinery.

[43] Elin Carstensdottir, Nathan Partlan, Steven Sutherland, Tyler Duke, Erika Ferris, Robin M Richter, Maria Valladares, and Magy Seif El-Nasr. Progression maps: conceptualizing narrative structure for interaction design support. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2020.

[44] Marc Cavazza, Fred Charles, and Steven J Mead. Character-based interactive storytelling. *IEEE Intelligent systems*, 17(4):17–24, 2002.

[45] Marc Cavazza, Jean-Luc Lugrin, David Pizzi, and Fred Charles. Madame bovary on the holodeck: immersive interactive storytelling. In *Proceedings of the 15th ACM international conference on Multimedia*, pages 651–660, 2007.

[46] Guillaume Chaslot, Sander Bakkes, Istvan Szita, and Pieter Spronck. Monte-carlo tree search: A new framework for game ai. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 4, pages 216–217, 2008.

[47] Guillaume Chaslot, Jahn-Takeshi Saito, Bruno Bouzy, JWHM Uiterwijk, and H Jaap Van Den Herik. Monte-carlo strategies for computer go. In *Proceedings of the 18th BeNeLux Conference on Artificial Intelligence, Namur, Belgium*, pages 83–91, 2006.

[48] Michael Chemers. *Ghost Light*. Sothern Illinois UP, 2010.

[49] Michael M Chemers. 'lyke unto a lively thing' : Theatre history and social robotics. In *Theatre, Performance and Analogue Technology: Historical Interfaces and Intermedialities*, pages 232–249. Springer, 2013.

[50] John Cimino Chris Hecker. Spy party, 2018.

[51] Agatha Christie. *And then there were none*. Harper, New York, 2011 - 1939.

[52] Chunsoft. Nine Hours, Nine Persons, Nine Doors, 2009.

[53] Tyler Colp. Final fantasy 14 player is eating thousands of eggs on stream with no plans to stop, 2021. Polygon article. Accessed June 22nd, 2023.

[54] Tyler Colp. Gamer of the year 2022: Let me solo her, 2022. PC Gamer article. Accessed June 22nd, 2023.

[55] Kate Compton. So you want to build a generator..., february 2016. blog post accessed on 5/26/2023.

[56] Kate Compton, Quinn Kybartas, and Michael Mateas. Tracery: an author-focused generative text tool. In *Interactive Storytelling: 8th International Conference on Interactive Digital Storytelling, ICIDS 2015, Copenhagen, Denmark, November 30-December 4, 2015, Proceedings 8*, pages 154–161. Springer, 2015.

[57] Michael Cook, Jeremy Gow, Gillian Smith, and Simon Colton. Danesh: Interactive tools for understanding procedural content generators. *IEEE Transactions on Games*, 14(3):329–338, 2021.

[58] Core-A Gaming. Analysis: How to Pick a Character.

[59] Hilde Corneliussen and Jill Walker Rettberg. *Digital culture, play, and identity: A World of Warcraft reader*. MIT Press, 2008.

[60] Nicholas G Cragoe. Rpg mythos: narrative gaming as modern mythmaking. *Games and Culture*, 11(6):583–607, 2016.

[61] Caroline Crampton. Honkaku: a century of the japanese whodunnits keeping readers guessing, 2021.

[62] Studio Deen. Descending stories: Showa genroku rakugo shinju, 2016.

[63] Sam Devlin, Anastasija Anspoka, Nick Sephton, Peter Cowling, and Jeff Rollason. Combining gameplay data with monte carlo tree search to emulate human play. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 12, pages 16–22, 2016.

[64] Richard M Dorson. Mythology and folklore. *Annual Review of Anthropology*, 2:107–126, 1973.

[65] Bioware Edmonton. Dragon age: Inquisition, 2014.

[66] Squirrel andWill Emigh Eiserloh, Jake Forbes, Jason Grinblat, Jonathan Hameland Ray Holmes, Thom Robertson, Ian Schreiber, Mike Sellers, Steve Swink, and Linda Law. Group report: Generative systems, meaningful cores. In *The Tenth Annual Game Design Think Tank Project Horseshoe 2015*, 2015.

[67] Magy Seif El-Nasr. A user-centric adaptive story architecture: borrowing from acting theories. In *Proceedings of the 2004 ACM SIGCHI International Conference on Advances in computer entertainment technology*, pages 109–116, 2004.

[68] Magy Seif El-Nasr. Interaction, narrative, and drama: Creating an adaptive interactive narrative using performance arts theories. *Interaction Studies*, 8(2):209–240, 2007.

[69] Magy Seif El-Nasr and Ian Horswill. Automating lighting design for interactive entertainment. *Computers in Entertainment (CIE)*, 2(2):15–15, 2004.

[70] Magy Seif El-Nasr, John Yen, and Thomas R Ioerger. Flame—fuzzy logic adaptive model of emotions. *Autonomous Agents and Multi-agent systems*, 3(3):219–257, 2000.

[71] Mirjam Palosaari Eladhari. Re-tellings: the fourth layer of narrative as an instrument for critique. In *International Conference on Interactive Digital Storytelling*, pages 65–78. Springer, 2018.

[72] Richard Evans and Emily Short. Versu - a Simulationist Storytelling System. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(2):113–130, 2014.

[73] Evo Events. Official evo moment #37, daigo vs justin evo 2004 in hd.

[74] Judith Fathallah. Reading real person fiction as digital fiction: An argument for new perspectives. *Convergence*, 24(6):568–586, 2018.

[75] Dan Feng, Elin Carstensdottir, Sharon Marie Carnicke, Magy Seif El-Nasr, and Stacy Marsella. An active analysis and crowd sourced approach to social training. In *International Conference on Interactive Digital Storytelling*, pages 156–167. Springer, 2016.

[76] Hilmar Finnsson and Yngvi Björnsson. Learning simulation control in general game-playing agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pages 954–959, 2010.

[77] Dom Ford. The haunting of ancient societies in the mass effect trilogy and the legend of zelda: Breath of the wild. *Game Studies*, 21(4), 2021.

[78] Dom Ford. That old school feeling, indeterminable year, after 2020.

[79] James George Frazer. *The golden bough*. Springer, 1922.

[80] French Bread and Ecole Software. Under Night In-Birth, 2012.

[81] From Software. Dark Souls, 2011.

[82] From Software. Elden Ring, 2022.

[83] Kyle Gabler, Kyle Gray, S Shodhan, and Matt Kucic. How to prototype a game in under 7 days. *Gamasutra, October*, 26, 2005.

[84] Ethan Gach. Destiny 2's wild corridors of time puzzle ends with lackluster reward, 2020. Kotaku article. Accessed June 22nd, 2023.

[85] Michael Genesereth, Nathaniel Love, and Barney Pell. General game playing: Overview of the aaai competition. *AI magazine*, 26(2):62–62, 2005.

[86] Robert M Geraci. *Virtually sacred: Myth and meaning in world of warcraft and second life*. Oxford University Press, USA, 2014.

[87] Heiner Goebbels, Jane Collins, David Roesner, Nicholas Till, and Nicole Gronemeyer. *Aesthetics of absence: Texts on theatre*. Routledge, 2015.

[88] Erving Goffman. *The presentation of self in everyday life*. The Overlook Press, 1959.

[89] Golden Glitch Studios. Elsinore, 2019.

[90] Adrian Goldwaser and Michael Thielscher. Deep reinforcement learning for general game playing. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 1701–1708, 2020.

[91] Lindsay Grace. Hauntology, the penumbra, and the narratives of play experience. In *25th International Symposium of Electronic Art (ISEA), Gwangju, South Korea*, 2019.

[92] Kyle Gratton. Struggling in elden ring? head south, 2022. Screenrant article. Accessed June 22nd, 2023.

[93] Jason Grinblat. Emergent narratives and story volumes. In Tanya Short and Tarn Adams, editors, *Procedural Generation in Game Design*. CRC Press, 2017.

[94] Jason Grinblat, Cat Manning, and Max Kreminski. Emergent narrative and reparative play. In *International Conference on Interactive Digital Storytelling*, pages 208–216. Springer, 2021.

[95] Saumya Gupta, Theresa Jean Tanenbaum, Meena Devii Muralikumar, and Aparajita S Marathe. Investigating roleplaying and identity transformation in a virtual reality narrative experience. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2020.

[96] Matthew Guzdial, Devi Acharya, Max Kreminski, Michael Cook, Mirjam Eladhari, Antonios Liapis, and Anne Sullivan. Tabletop roleplaying games as procedu-

ral content generators. In *International Conference on the Foundations of Digital Games*, 2020.

[97] Christopher Hanson. *Recursive Temporalities*, pages 135–155. Indiana University Press, 2018.

[98] Johnathan Harrington. 4x gamer as myth: Understanding through player mythologies. In *DiGRA/FDG*, 2016.

[99] Justin Harris and R Michael Young. Proactive mediation in plan-based narrative environments. In *International Workshop on Intelligent Virtual Agents*, pages 292–304. Springer, 2005.

[100] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[101] Howard Haycraft. *Murder for pleasure : the life and times of the detective story.* Biblo and Tannen, New York, newly enlarged edition, with notes on additions to a cornerstone library and the haycraft-queen definitive library of detective-crime-mystery fiction. edition, 1974 - 1951.

[102] Golden age traditions, 2005.

[103] Kieran Hicks, Patrick Dickinson, Jussi Holopainen, Kathrin Gerling, et al. Good game feel: an empirically grounded framework for juicy design. In *Proceedings of*

the 2018 DiGRA International Conference: The Game is the Message. DiGRA, 2018.

[104] Kieran Hicks, Kathrin Gerling, Patrick Dickinson, and Vero Vanden Abeele. Juicy game design: Understanding the impact of visual embellishments on player experience. In *Proceedings of the Annual Symposium on Computer-Human Interaction in Play*, pages 185–197, 2019.

[105] Dan Higgins. Generic a* pathfinding. *AI Game Programming Wisdom*, pages 114–121, 2002.

[106] Elizabeth C Hirschman. Movies as myths: An interpretation of motion picture mythology. *Marketing and semiotics: New directions in the study of signs for sale*, pages 335–74, 1987.

[107] Christoffer Holmgård, Michael Cerny Green, Antonios Liapis, and Julian Togelius. Automated playtesting with procedural personas through mcts with evolved heuristics. *IEEE Transactions on Games*, 11(4):352–362, 2018.

[108] Christoffer Holmgård, Antonios Liapis, Julian Togelius, and Georgios N Yannakakis. Evolving personas for player decision modeling. In *2014 IEEE Conference on Computational Intelligence and Games*, pages 1–8. IEEE, 2014.

[109] Christoffer Holmgård, Antonios Liapis, Julian Togelius, and Georgios N Yannakakis. Evolving models of player decision making: Personas versus clones. *Entertainment Computing*, 16:95–104, 2016.

[110] Britton Horn, Josh Miller, Gillian Smith, and Seth Cooper. A monte carlo approach to skill-based automated playtesting. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 14, pages 166–172, 2018.

[111] Johan Huizinga. *Homo Ludens*. Random House, 1956.

[112] Ice-Pick Lodge. Pathologic, 2005.

[113] Ice-Pick Lodge. Pathologic Classic HD, 2015.

[114] FromSoftware Inc. Chromehounds, 2006.

[115] FromSoftware Inc. Armored Core: For Answer, 2008.

[116] My Name is Byf. My name is byf's youtube channel, 2011–Now.

[117] Jeremy Tiang Jackie Chan. *Never Grow Up*. Gallery Books, 2018.

[118] Mikhail Jacob, Alexander Zook, and Brian Magerko. Viewpoints AI: Procedurally Representing and Reasoning about Gestures. 08 2013.

[119] Emil Juul Jacobsen, Rasmus Greve, and Julian Togelius. Monte mario: platforming with mcts. In *Proceedings of the 2014 annual conference on genetic and evolutionary computation*, pages 293–300, 2014.

[120] James Ryan, Ben Samuel, Adam Summerville. Bad News, 2015.

[121] Jett. Universal Fighting Game Guide: How to Read Frame Data.

[122] Elizabeth Jochum and Todd Murphey. Programming play: Puppets, robots, and engineering. *The Routledge Companion to Puppetry and Material Performance*, pages 308–321, 2014.

[123] Basil Jones. Puppetry, authorship, and the ur-narrative. *The Routledge Companion to Puppetry and Material Performance*, pages 61–67, 2014.

[124] Nic June, Ronald Murray, C Marshall, Tamara Duplantis, Isaac Karth, and Max Kreminski. Tracks in snow, 2021.

[125] Nic Junius and Elin Carstensdottir. Expressive response curves: testing expressive game feel with a. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 19, pages 284–294, 2023.

[126] Nic Junius, Michael Mateas, Noah Wardrip-Fruin, and Elin Carstensdottir. Playing with the strings: Designing puppitor as an acting interface for digital games. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 18, pages 250–257, 2022.

[127] Nick Junius. *Puppitor: Building an Acting Interface for Videogames*. PhD thesis, UC Santa Cruz, 2019.

[128] Nick Junius. *TRACKS IN SNOW: A DIGITAL PLAY ABOUT JUDAISM AND HOME*. PhD thesis, UNIVERSITY OF CALIFORNIA SANTA CRUZ, 2021.

[129] Nick Junius, Max Kreminski, and Michael Mateas. There Is No Escape: The-

atricality in *Hades*. In *Proceedings of the 16th International Conference on the Foundations of Digital Games*, 2021.

[130] Nick Junius, Michael Mateas, and Noah Wardrip-Fruin. Towards Expressive Input for Character Dialogue in Digital Games. In *Proceedings of the 14th International Conference on the Foundations of Digital Games*, 2019.

[131] Jesper Juul. *Half-real: Video games between real rules and fictional worlds*. MIT press, 2005.

[132] Stephen Kaplin. A puppet tree: a model for the field of puppet theatre. *TDR/The Drama Review*, 43(3):28–35, 1999.

[133] Isaac Karth. Preliminary poetics of procedural generation in games. *Transactions of the Digital Games Research Association*, 4(3), 2019.

[134] Isaac Karth, Nic Junius, and Max Kreminski. Constructing a catbox: Story volume poetics in umineko no naku koro ni. In *Interactive Storytelling: 15th International Conference on Interactive Digital Storytelling, ICIDS 2022, Santa Cruz, CA, USA, December 4–7, 2022, Proceedings*, pages 455–470. Springer, 2022.

[135] Oleksandra Keehl and Adam M Smith. Monster carlo: an mcts-based framework for machine playtesting unity games. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8. IEEE, 2018.

[136] Keiya. *Saishū kōsatsu umineko no naku koro ni chiru: Answer to the golden witch Episode 5-8*. ASCII Media Works, Tōkyō, 2011.

[137] Adam Kjellgren. Mythmaking as a feminist strategy: Rosi braidotti' s political myth. *Feminist Theory*, 22(1):63–80, 2021.

[138] Erica Kleinman, Elin Carstensdottir, and Magy Seif El-Nasr. Going forward by going back: re-defining rewind mechanics in narrative games. In *Proceedings of the 13th International Conference on the Foundations of Digital Games*, 2018.

[139] Erica Kleinman, Elin Carstensdottir, and Magy Seif El-Nasr. Going forward by going back: Re-defining rewind mechanics in narrative games. In *Proceedings of the 13th International Conference on the Foundations of Digital Games*, FDG '18, New York, NY, USA, 2018. Association for Computing Machinery.

[140] Hartmut Koenitz. Towards a theoretical framework for interactive digital narrative. In *Joint International Conference on Interactive Digital Storytelling*, pages 176–185. Springer, 2010.

[141] Hartmut Koenitz. Towards a specific theory of interactive digital narrative. In *Interactive digital narrative*, pages 91–105. Routledge, 2015.

[142] A. Koseki, H. Komastu, and Y. Fukazawa. A method for estimating optimal unrolling times for nested loops. In *Proceedings of the 1997 International Symposium on Parallel Architectures, Algorithms and Networks (I-SPAN'97)*, pages 376–382, 1997.

[143] Max Kreminski, Melanie Dickinson, Noah Wardrip-Fruin, and Michael Mateas. Loose ends: a mixed-initiative creative interface for playful storytelling. In *Pro-*

ceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, volume 18, pages 120–128, 2022.

[144] Max Kreminski, Isaac Karth, Michael Mateas, and Noah Wardrip-Fruin. Evaluating mixed-initiative creative interfaces via expressive range coverage analysis. In *3rd Workshop on Human-AI Co-Creation with Generative Models*, 2022.

[145] Tanya Krzywinska. Blood scythes, festivals, quests, and backstories: World creation and rhetorics of myth in world of warcraft. *Games and Culture*, 1(4):383–396, 2006.

[146] Haruko Kumota. Descending stories: Showa genroku rakugo shinju, 2015.

[147] Quinn Kybartas, Clark Verbrugge, and Jonathan Lessard. Tension space analysis for emergent narrative. *IEEE Transactions on Games*, 13(2):146–159, 2020.

[148] Bjarke Alexander Larsen, Luis Emilio Bruni, and Henrik Schoenau-Fog. The story we cannot see: on how a retelling relates to its afterstory. In *International Conference on Interactive Digital Storytelling*, pages 190–203. Springer, 2019.

[149] Bjarke Alexander Larsen and Elin Carstensdottir. Wrestling with destiny: Storytelling in perennial games. In *International Conference on Interactive Digital Storytelling*, pages 236–254. Springer, 2021.

[150] Bjarke Alexander Larsen and Elin Carstensdottir. Myth, diegesis and storytelling in perennial games. In *Interactive Storytelling: 15th International Conference on*

Interactive Digital Storytelling, ICIDS 2022, Santa Cruz, CA, USA, December 4–7, 2022, Proceedings, pages 634–650. Springer, 2022.

[151] Bjarke Alexander Larsen, Nic Junius, and Elin Carstensdottir. Communal ritual play: Repetition and interpretation of game narratives across communities. In *International Conference on Interactive Digital Storytelling*, pages 472–488. Springer, 2023.

[152] Brenda Laurel. *Computers as Theater*. Addison-Wesley Professional, 2nd. edition, 2013.

[153] Brenda Laurel. *Computers as theatre*. Addison-Wesley, 2013.

[154] Brenda Kay Laurel. *TOWARD THE DESIGN OF A COMPUTER-BASED INTERACTIVE FANTASY SYSTEM (DRAMA, PLAYWRITING, POETICS, EXPERT SYSTEMS, THEORY)*. PhD thesis, The Ohio State University, 1986.

[155] Hans-Thies Lehmann. *Postdramatic theatre*. Routledge, 2006.

[156] Claude Lévi-Strauss. The structural study of myth. *The journal of American folklore*, 68(270):428–444, 1955.

[157] Claude Lévi-Strauss. Structuralism and myth. *The Kenyon Review*, 3(2):64–88, 1981.

[158] Irina Levin and Igor Levin. *The Stanislavsky Secret*. Colorado: Meriwether Publishing, 2002.

[159] Antonios Liapis, Christoffer Holmgård, Georgios N Yannakakis, and Julian Togelius. Procedural personas as critics for dungeon generation. In *Applications of Evolutionary Computation: 18th European Conference, EvoApplications 2015, Copenhagen, Denmark, April 8-10, 2015, Proceedings 18*, pages 331–343. Springer, 2015.

[160] Duri Long, Sanjana Gupta, Jessica Anderson, and Brian Magerko. The shape of story: A semiotic artistic visualization of a communal storytelling experience. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 13, pages 204–211, 2017.

[161] Duri Long, Mikhail Jacob, and Brian Magerko. Designing co-creative ai for public spaces. In *Proceedings of the 2019 on Creativity and Cognition*, pages 271–284, 2019.

[162] Bronislaw Malinowski. *Magic, science and religion and other essays*. Read Books Limited, 1954.

[163] Chris Martens and Owais Iqbal. Villanelle: An authoring tool for autonomous characters in interactive fiction. In *International Conference on Interactive Digital Storytelling*, pages 290–303. Springer, 2019.

[164] Chris Martens, Owais Iqbal, Sasha Azad, Maddie Ingling, Anthony Mosolf, Emma McCamey, and Johanna Timmer. Villanelle: Towards authorable autonomous characters in interactive narrative. In *INT/WICED@ AIIDE*, 2018.

[165] Michael Mateas. An oz-centric review of interactive drama and believable agents. In *Artificial intelligence today*, pages 297–328. Springer, 1999.

[166] Michael Mateas. A neo-aristotelian theory of interactive drama. In *Working notes of the AI and Interactive Entertainment Symposium*, 2000.

[167] Michael Mateas. A preliminary poetics for interactive drama and games. *Digital Creativity*, 12(3):140–152, 2001.

[168] Michael Mateas, Steffi Domike, and Paul Vanouse. Terminal time: An ideologically-biased history machine. *AISB Quarterly, Special Issue on Creativity in the Arts and Sciences*, 102:36–43, 1999.

[169] Michael Mateas and Andrew Stern. A behavior language for story-based believable agents. *IEEE Intelligent Systems*, 17(4):39–47, 2002.

[170] Michael Mateas and Andrew Stern. Towards integrating plot and character for interactive drama. In *Socially Intelligent Agents*, pages 221–228. Springer, 2002.

[171] Michael Mateas and Andrew Stern. Build it to understand it: Ludology meets narratology in game design space. In *Proceedings of DiGRA 2005 Conference: Changing Views: Worlds in Play*, 2005.

[172] Michael Mateas and Andrew Stern. Façade, 2005.

[173] Michael Mateas and Andrew Stern. Structuring Content in the Façade Interactive Drama Architecture. In *AIIDE*, pages 93–98, 2005.

[174] Maxis. The sims, 2000.

[175] Dick McCaw. *The Laban Sourcebook*. Routledge, 2012.

[176] Josh McCoy, Mike Treanor, Ben Samuel, Michael Mateas, and Noah Wardrip-Fruin. Prom week: social physics as gameplay. In *Proceedings of the 6th International Conference on Foundations of Digital Games*, pages 319–321, 2011.

[177] Josh McCoy, Mike Treanor, Ben Samuel, Aaron A Reed, Michael Mateas, and Noah Wardrip-Fruin. Prom week: Designing past the game/story dilemma.

[178] Josh McCoy, Mike Treanor, Ben Samuel, Brandon Tearse, Michael Mateas, and Noah Wardrip-Fruin. Authoring game-based interactive narrative using social games and comme il faut. In *Proceedings of the 4th International Conference & Festival of the Electronic Literature Organization: Archive & Innovate*. Citeseer, 2010.

[179] Josh McCoy, Mike Treanor, Ben Samuel, Noah Wardrip-Fruin, and Michael Mateas. Prom week, 2012.

[180] Joshua McCoy and Michael Mateas. The Computation of Self in Everyday Life: A Dramaturgical Approach for Socially Competent Agents. In *AAAI Spring Symposium: Intelligent Narrative Technologies II*, pages 75–82, 2009.

[181] Joshua McCoy, Michael Mateas, and Noah Wardrip-Fruin. Comme il faut: A system for simulating social games between autonomous characters. 2009.

[182] Joshua McCoy, Mike Treanor, Ben Samuel, Noah Wardrip-Fruin, and Michael Mateas. Comme il faut: A system for authoring playable social models. In *Seventh Artificial Intelligence and Interactive Digital Entertainment Conference*, 2011.

[183] Josh Aaron Miller, Kutub Gandhi, Matthew Alexander Whitby, Mehmet Kosa, Seth Cooper, Elisa D. Mekler, and Ioanna Iacovides. A design framework for reflective play. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, CHI '24, New York, NY, USA, 2024. Association for Computing Machinery.

[184] Patrick Miller. On finding your main character, 2019.

[185] Patrick Miller. Why we fight, 2019.

[186] Patrick Miller. Developing the training mindset in fighting games, 2020.

[187] Patrick Miller. Why fighting games are hard, 2020.

[188] Alex Mitchell. Reflective rereading and the simcity effect in interactive stories. In *International Conference on Interactive Digital Storytelling*, pages 27–39. Springer, 2015.

[189] Alex Mitchell and Liting Kway. "how do i restart this thing?" repeat experience and resistance to closure in rewind storygames. In Anne-Gwenn Bosser, David E. Millard, and Charlie Hargood, editors, *Interactive Storytelling*, pages 164–177, Cham, 2020. Springer International Publishing.

[190] Alex Mitchell and Liting Kway. "how do i restart this thing?" repeat experience and resistance to closure in rewind storygames. In *Interactive Storytelling: 13th International Conference on Interactive Digital Storytelling, ICIDS 2020, Bournemouth, UK, November 3–6, 2020, Proceedings 13*, pages 164–177. Springer, 2020.

[191] Alex Mitchell, Liting Kway, and Brandon Junhui Lee. Storygameness: understanding repeat experience and the desire for closure in storygames. DiGRA, 2020.

[192] Alex Mitchell, Liting Kway, and Brandon Junhui Lee. Storygameness: understanding repeat experience and the desire for closure in storygames. In *DiGRA 2020–Proceedings of the 2020 DiGRA International Conference*, 2020.

[193] Alex Mitchell and Kevin McGee. Reading again for the first time: a model of rereading in interactive stories. In *Interactive Storytelling: 5th International Conference, ICIDS 2012, San Sebastián, Spain, November 12-15, 2012. Proceedings 5*, pages 202–213. Springer, 2012.

[194] Alex Mitchell and Kevin McGee. Reading again for the first time: A model of rereading in interactive stories. In David Oyarzun, Federico Peinado, R. Michael Young, Ane Elizalde, and Gonzalo Méndez, editors, *Interactive Storytelling*, pages 202–213, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[195] Mojang. Minecraft, 2011.

[196] Luvneesh Mugrai, Fernando Silva, Christoffer Holmgård, and Julian Togelius. Automated playtesting of matching tile games. In *2019 IEEE Conference on Games (CoG)*, pages 1–7. IEEE, 2019.

[197] Janet Horowitz Murray and Janet H Murray. *Hamlet on the holodeck: The future of narrative in cyberspace*. MIT press, 2017.

[198] Myelin. Myelin's youtube channel, 2014–Now.

[199] Gregory Nagy. *Greek mythology and poetics*, volume 2. Cornell University Press, 1992.

[200] Nintendo. The Legend of Zelda: Ocarina of Time, 1998.

[201] Nintendo. The Legend of Zelda: Majora's Mask, 2000.

[202] Jeff Orkin. Agent architecture considerations for real-time planning in games. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 1, pages 105–110, 2005.

[203] Jeff Orkin. Three states and a plan: the ai of fear. In *Game developers conference*, volume 2006, page 4. CMP Game Group SanJose, California, 2006.

[204] Andrew Ortony, Gerald L Clore, and Allan Collins. *The cognitive structure of emotions*. Cambridge university press, 1990.

[205] Joseph Carter Osborn, Benjamin Samuel, and Michael Mateas. Visualizing the

strategic landscape of arbitrary games. *Information Visualization*, 17(3):196–217, 2018.

[206] Diego Perez-Liebana, Sanaz Mostaghim, and Simon M Lucas. Multi-objective tree search approaches for general video game playing. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 624–631. IEEE, 2016.

[207] Johannes Pfau, Antonios Liapis, Georg Volkmar, Georgios N Yannakakis, and Rainer Malaka. Dungeons & replicants: automated game balancing via deep player behavior modeling. In *2020 IEEE Conference on Games (CoG)*, pages 431–438. IEEE, 2020.

[208] Johannes Pfau, Antonios Liapis, Georgios N Yannakakis, and Rainer Malaka. Dungeons & replicants ii: automated game balancing across multiple difficulty dimensions via deep player behavior modeling. *IEEE Transactions on Games*, 2022.

[209] Platinum Games. Nier: Automata, 2017.

[210] Alexandru Popescu, Joost Broekens, and Maarten van Someren. Gamygdala: An emotion engine for games. *IEEE Transactions on Affective Computing*, 5(1):32–44, 2014.

[211] Julie Porteous, Marc Cavazza, and Fred Charles. Applying planning to interactive storytelling: Narrative control using state constraints. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 1(2):1–21, 2010.

[212] Cody Poulton. From puppet to robot: Technology and the human in japanese theater. *The Routledge Companion to Puppetry and Material Performance*, pages 280–291, 2014.

[213] Edward J Powley, Simon Colton, Swen Gaudl, Rob Saunders, and Mark J Nelson. Semi-automated level design via auto-playtesting for handheld casual game creation. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8. IEEE, 2016.

[214] Interplay Productions. Fallout: A post nuclear role playing game, 1997.

[215] Younès Rabii and Michael Cook. Why oatmeal is cheap: Kolmogorov complexity and procedural generation. In *Proceedings of the 18th International Conference on the Foundations of Digital Games*, pages 1–7, 2023.

[216] CD Projekt Red. The witcher 3: Wild hunt, 2015.

[217] Aaron A. Reed. Subcutanean, 2020.

[218] Rekka Alexiel. YoRHa Stage Play Differences | Fire Sanctuary.

[219] Rekka Alexiel. YoRHa Stage Play, Ver 1.1 | Fire Sanctuary.

[220] Jill Walker Rettberg. *Quests in World of Warcraft: Deferral and repetition*, chapter 8, pages 167–184. MIT Press, 2008.

[221] Nic Reuben. All the realm's a stage: Exploring final fantasy 14's incredible virtual theater shows, 2020. Accessed June 22nd, 2023.

[222] Mark O Riedl and Robert Michael Young. Narrative planning: Balancing plot and character. *Journal of Artificial Intelligence Research*, 39:217–268, 2010.

[223] J Thomas Rimer, Masakazu Yamazaki, et al. *On the Art of the Nō Drama: The Major Treatises of Zeami; Translated by J. Thomas Rimer, Yamazaki Masakazu*. Princeton University Press, 1984.

[224] Joseph R Roach. *The player's passion: studies in the science of acting*. University of Michigan Press, 1993.

[225] Justus Robertson, John Heiden, and Rogelio E Cardona-Rivera. Evolving interactive narrative worlds. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 19, pages 126–135, 2023.

[226] Tom Rothamel. Ren'Py, 2004.

[227] Doris C Rusch. 21st century soul guides: Leveraging myth and ritual for game design. *DiGRA Nordic Subversion, Transgression, and Controversy in Play, University of Bergen, Norway*, 2018.

[228] Marie-Laure Ryan. *Avatars of Story*. University of Minnesota Press, 111 Third Avenue South, Suite 290, Minneapolis, August 2006.

[229] Serdar Sali, Noah Wardrip-Fruin, Steven Dow, Michael Mateas, Sri Kurniawan, Aaron A Reed, and Ronald Liu. Playing with words: from intuition to evaluation

of game dialogue interfaces. In *Proceedings of the Fifth International Conference on the Foundations of Digital Games*, pages 179–186. ACM, 2010.

[230] Ben Samuel. *Crafting Stories Through Play*. PhD thesis, University of California, Santa Cruz, Santa Cruz, CA, December 2016.

[231] Ben Samuel, James Ryan, Adam Summerville, Michael Mateas, and Noah Wardrip-Fruin. Computatrum personae: toward a role-based taxonomy of (computationally assisted) performance. In *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2016.

[232] Ben Samuel, James Ryan, Adam J Summerville, Michael Mateas, and Noah Wardrip-Fruin. Bad news: An experiment in computationally assisted performance. In *International Conference on Interactive Digital Storytelling*, pages 108–120. Springer, 2016.

[233] Eve Kosofsky Sedgwick. Paranoid reading and reparative reading, or, you're so paranoid, you probably think this essay is about you. In *Touching Feeling*, pages 123–152. Duke University Press, 2003.

[234] Robert A Segal. *Myth: A very short introduction*. OUP Oxford, 2004.

[235] Jake Selway. Elden ring - who is let me solo her?, 2022. Gamerant article. Accessed June 22nd, 2023.

[236] Shenpai et al. Lunarcon, 2021–Now. Convention run inside Final Fantasy XIV. Accessed June 22nd, 2023.

[237] Mei Si, Stacy C Marsella, and David V Pynadath. Thespian: An architecture for interactive pedagogical drama. In *AIED*, pages 595–602, 2005.

[238] Mei Si, Stacy C Marsella, and David V Pynadath. Thespian: Using multi-agent fitting to craft interactive drama. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 21–28, 2005.

[239] Mei Si, Stacy C Marsella, and David V Pynadath. Thespian: Modeling socially normative behavior in a decision-theoretic framework. In *International Workshop on Intelligent Virtual Agents*, pages 369–382. Springer, 2006.

[240] Reginald Rose Sidney Lumet. Twelve angry men, 1957.

[241] Mark Silver and Rosemary Herbert. Japan, crime and mystery writing in, 2005.

[242] Sam Smiley and Norman A Bert. *Playwriting: The structure of action*. Yale University Press, 2005.

[243] Adam M Smith and Michael Mateas. Computational caricatures: Probing the game design process with ai. In *Workshops at the Seventh Artificial Intelligence and Interactive Digital Entertainment Conference*, 2011.

[244] Gillian Smith and Jim Whitehead. Analyzing the expressive range of a level generator. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, 2010.

[245] Gillian Smith and Jim Whitehead. Analyzing the expressive range of a level

generator. In *Proceedings of the 2010 workshop on procedural content generation in games*, pages 1–7, 2010.

[246] Square Enix. Final Fantasy XIV, 2013.

[247] Squinky. Coffee: A Misunderstanding, 2014.

[248] Tom Stoppard. *Rosencrantz and Guildenstern Are Dead*. Grove/Atlantic, Inc., 2007.

[249] Neil Strauss. Faster than a speeding bullet, but also humanly fallible, 1995.

[250] Actas Studio 3Hz. Princess principal, 2017.

[251] Sukeban Games. VA-11 Hall-A, 2016.

[252] Adam Summerville. Expanding expressive range: Evaluation methodologies for procedural content generation. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 14, pages 116–122, 2018.

[253] Supergiant Games. Bastion, 2011.

[254] Supergiant Games. Transistor, 2014.

[255] Supergiant Games. Hades, 2020.

[256] Maciej Świechowski, Konrad Godlewski, Bartosz Sawicki, and Jacek Mańdziuk. Monte carlo tree search: A review of recent modifications and applications. *Artificial Intelligence Review*, 56(3):2497–2562, 2023.

[257] Maciej Świechowski, HyunSoo Park, Jacek Mańdziuk, Kyung-Joong Kim, et al. Recent advances in general game playing. *The Scientific World Journal*, 2015, 2015.

[258] Steve Swink. *Game feel: a game designer's guide to virtual sensation*. CRC Press, 2008.

[259] Nicolas Szilas. Idtension: a narrative engine for interactive drama. In *Proceedings of the technologies for interactive digital storytelling and entertainment (TIDSE) conference*, volume 3, pages 1–11, 2003.

[260] Theresa Jean Tanenbaum. Being in the story: readerly pleasure, acting theory, and performing a role. In *International Conference on Interactive Digital Storytelling*, pages 55–66. Springer, 2011.

[261] Theresa Jean Tanenbaum and Karen Tanenbaum. Empathy and identity in digital games: Towards a new theory of transformative play. In *FDG*, 2015.

[262] Telltale Games. The Walking Dead: Season One, 2012.

[263] Bronwen Thomas. Fans behaving badly? real person fic and the blurring of the boundaries between the public and the private. In Bronwen Thomas and Julia Round, editors, *Real Lives, Celebrity Stories : Narratives of Ordinary and Extraordinary People across Media*, pages 171–186. Bloomsbury Academic, 1 edition, 2014.

[264] Rob Tieben, Janienke Sturm, Tilde Bekker, and Ben Schouten. Playful persuasion:

Designing for ambient playful interactions in public spaces. *Journal of Ambient Intelligence and Smart Environments*, 6(4):341–357, 2014.

[265] Steve Tillis. The art of puppetry in the age of media production. *TDR/The Drama Review*, 43(3):182–195, 1999.

[266] Milka Trajkova, Duri Long, Manoj Deshpande, Andrea Knowlton, and Brian Magerko. Exploring collaborative movement improvisation towards the design of luminai—a co-creative ai dance partner. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, CHI '24, New York, NY, USA, 2024. Association for Computing Machinery.

[267] Victor Turner. *From Ritual to Theatre: the human seriousness of play.* PAJ Publications, 1982.

[268] Marcel Violette and Alain Cloarec. String puppet. *World Encyclopedia of Puppetry Arts*, 2009.

[269] Jesse Vitelli. Final fantasy xiv' s club scene is all the rage, 2022. Accessed June 22nd, 2023.

[270] Volition. Descent: FreeSpace - The Great War, 1998.

[271] Volition. Freespace 2, 1999.

[272] Rachel Wagner. The importance of playing in earnest. *Playing with Religion in digital games*, pages 192–213, 2014.

[273] Noah Wardrip-Fruin. *Expressive Processing: Digital fictions, computer games, and software studies*. MIT press, 2009.

[274] Stephen G Ware and Cory Siler. Sabre: a narrative planner supporting intention and deep theory of mind. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 17, pages 99–106, 2021.

[275] Stephen G Ware and R Michael Young. Glaive: a state-space narrative planner supporting intentionality and conflict. In *Tenth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2014.

[276] Rachel Winter, Anastasia Salter, and Mel Stanfill. Communities of making: Exploring parallels between fandom and open source. *First Monday*, 2021.

[277] Yoko Taro. YoRHa Stage Play, Ver 1.1, 2015.

[278] R Michael Young. Notes on the use of plan structures in the creation of interactive plot. In *AAAI fall symposium on narrative intelligence*, pages 164–167. AAAI Press Menlo Park, 1999.

[279] R Michael Young, Stephen G Ware, Brad A Cassell, and Justus Robertson. Plans and planning in narrative generation: a review of plan-based approaches to the generation of story, discourse and interactivity in narratives. *Sprache und Datenverarbeitung, Special Issue on Formal and Computational Models of Narrative*, 37(1-2):41–64, 2013.

[280] ZA/UM. Disco Elysium, 2019.

[281] John Zimmerman, Jodi Forlizzi, and Shelley Evenson. Research through design as a method for interaction design research in hci. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 493–502, 2007.

[282] Alexander Zook, Brent Harrison, and Mark O Riedl. Monte-carlo tree search for simulation-based strategy analysis. In *Proceedings of the 10th International Conference on the Foundations of Digital Games*, 2015.