UNIVERSITY OF CALIFORNIA

SANTA CRUZ

THE GRAIL FRAMEWORK: MAKING STORIES PLAYABLE ON THREE LEVELS IN CRPGS

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

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Anne Margaret Sullivan

June 2012

The Dissertation of Anne Margaret Sullivan
is approved:

_____
Associate Professor Michael Mateas, Chair

_____
Associate Professor Noah Wardrip-Fruin

_____
Professor Jim Whitehead

_____
Associate Professor R. Michael Young

_____
Tyrus Miller
Vice Provost and Dean of Graduate Studies

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

THE GRAIL FRAMEWORK:

MAKING STORIES PLAYABLE ON THREE LEVELS IN CRPGS

by Anne Margaret Sullivan

Computer role-playing games (CRPGs) have strong narratives, but in general lack interesting and meaningful choices for the player within the story. As a result, the stories are not *playable.* This breakdown happens at four different levels within storytelling: the player, quest, game and world story levels. The world story level is the structure and setting of the world that the game inhabits and is generally static. The game story level describes the over-arching story which is at best minimally branching in most CRPGs. The quest story level is the level in which players are given a series of tasks to complete, with little choice in the outcome. Finally, the player story level of storytelling within CRPGs is defined by a sequence of moment-to-moment decisions a player makes within the constraints of the game mechanics, which the player feels is worth re-telling. The Grail Framework was created to address playability at three of the levels: the game, quest, and player story levels.

This dissertation presents the Grail Framework and its use in creating more playable stories. It first describes CiF-RPG and how social mechanics give rise to a new set of player stories. It then moves on to discuss GrailGM which uses content selection methods to create dynamic quest and game stories. From there, the discussion turns to the tools, Brainstormer and Social Mechanics Design Tool, created to ease the authorial burden of creating content for

dynamic systems, followed by the playable experiences, *EMPath* and *Mismanor*, designed and built to explore the possibilities of playable story. Finally, it ends with evaluating the expressive range of the systems.

By constructing the Grail Framework and *Mismanor,* we have created an existence proof for a new approach to story in role-playing games — one with power and flexibility that addresses known problems in the game genre and allows it to move toward experiences that have proven impractical with today's technical and design approaches.

ACKNOWLEDGEMENTS

While writing a dissertation is a solo undertaking, the work that goes into preparing for one is far from a single-player adventure. I'm not sure my family realized the course they were setting me on when they bought the Odyssey2 and a copy of K.C. Munchkin, but I appreciate that they were always supportive of my perhaps overly obsessive use of the computer. My path became firmly set when my brother introduced me to online BBSes and I helped one of his friends create a music playing program in BASIC. I am thankful that my brother was open to having his little sister tag along on his online adventures, and supported my interest without judgment. Hopefully now that I am a doctor, he will finally forgive me for not learning how to pack my wheel bearings.

Graduate school was not an obvious choice to me, and I have Professors Darrell Long and Jim Whitehead to thank for pushing me towards taking on this quest. It has been one of the most rewarding things I have ever done, and I am truly grateful for your help getting me started.

Research requires the hands of many and without the help of Sherol Chen, Chris Lewis and Mark Nelson, EMPath would not have been completed or as successful. Likewise, Mismanor would not be the game it is without the much appreciated help of Max Stokols, Tabitha Chirrick, Lauren Scott, Joe Allington, Shawn Hampton, Vanessa Valencia, Kevin Meggs, Alex 'Strasza' Schneider, and Alex Mathew. I want to especially thank Josh McCoy for his help with CiF and the many interesting discussions on social aspects in games, and April Grow for her continued support and help on aspects of Mismanor and the Grail Framework, as well as her steady supply of Japanese treats and cute things to keep me sane.

Without the guidance of my stellar committee, I would still be mired in research complications and uncertainty. My thanks to Michael Young for his very much needed calm and his always helpful feedback, to Jim Whitehead for the laughs among the serious research and reminding me to not take

myself too seriously, to Michael Mateas for the many helpful conversations and exposing me to some of the coolest areas of research I never would have known about, and to Noah Wardrip-Fruin for helping me find my voice, helping me find a way to express my design side in my research and for the mental support on many levels. I truly cannot thank you all enough.

To Mirjam Eladhari, Jane Pinckard and Gillian Smith, thank you for the evenings at the Hinds House, those are some of my all-time favorite memories of grad school. Who would have thought those conversations would have led us where they did? Thank you to Tina Michalik, Julia Williams, Dana Harris, Susy Trower, Nadine Schaeffer, and the South Bay Area Modern Quilt Guild for helping me find a way to appease my creative side and find a balance to the research.

Grad school would not have been nearly so enjoyable without my best friend and partner-in-crime Gillian Smith there to experience it with. I would not want to be oft-confused with anyone else. Thank you for the lively debates, the spontaneous adventures (the waffles and that geocoin were totally worth it), the crafting, your generosity, and your unwavering friendship. I could fill a new dissertation with our shared adventures (which would likely be more fun to write!), but this acknowledgement section is far too short to capture all the thanks I owe you.

I want to thank my partner Eli Patterson, for his constant love and support, for putting up with the unending stress, late nights, freak outs and general craziness. Thank you for your willingness to always try a new adventure with me – hopefully the next one will take a little less time and cause fewer grey hairs!

And finally, thank you to Kita, Mishka, Tally and Obi for your lovable and fluffy presence, reminding me that sometimes there are more important things than just a few more minutes on the computer.

CHAPTER 1

INTRODUCTION

"A game is a series of interesting choices," (Rollings & Morris, 1999, p. 38) influential designer Sid Meier is often quoted as saying. This quote points to one of the central elements that make games an effective form of interactive media. It also points to one of the central sites of audience attention for games: the systems that present interesting choices are an ongoing focus during play, while players may increasingly "look past" those elements that don't, as their familiarity with a game grows.

But what makes an interesting choice? Rollings and Morris further observe, "In an interesting choice, no single option is clearly better than the other options, the options are not equally attractive, and the player must be able to make an informed choice" (Rollings & Morris, 1999, p. 38). This outlines the moment of choice, but Salen and Zimmerman point out that such moments must also be meaningful within our larger experience of play. A meaningful choice is one in which the outcome of the choice is both discernible and integrated (Salen & Zimmerman, 2005). Not only should the player be able to have interesting choices, but choices should have a noticeable (discernible) and significant (integrated) impact on the game world.

With these definitions in mind, in order for an experience to be a game, and therefore playable, the player must be presented with interesting and meaningful choices within that experience.

1 Playable Stories

The computer role-playing game (CRPG) is often considered one of the game forms with the strongest storytelling. However, while players are given interesting and meaningful choices, these choices are often confined to combat. Within the story, CRPGs are typically restricted to a linear narrative, which the player advances by fulfilling checkpoints. This leads to a strong dichotomy within the game. The player is able to create their own combat-based mini-stories – such as how they managed to kill the enemy boss – while filling a pre-determined character role along the overarching narrative backdrop. When recounting their game play, a player will often avoid mentioning the overarching narrative, as information about it is considered "spoilers," but instead will focus on the parts of the game in which they had the most control. Mentioning moments of the overarching narrative is considered a spoiler because every player will face the same moments, and the path through those story moments is the same with very little, if any, variation.

This lack of choice signifies that stories are not currently playable in many CRPGs. This is part of the reason that players, particularly in Massively-Multiplayer Online Role-Playing Games (MMORPGs) such as *World of Warcraft* (Blizzard Entertainment, 2004), look past the fictional specifics of quests and the game world (often referred to as "flavor text") as they do not involve their choices or character in any way.

The CRPG will never be able to reach its potential until its stories become playable. Only then will its players be able to engage in the activity from which it gets its name—role-playing—in a manner that approaches the power of true role-play, as seen in forms ranging from tabletop RPGs to Boalian "forum theater" (Boal, 1993).

1.1 Table-Top Role-Playing Games

To fully understand Computer Role-Playing Games (CRPGs) we must first look at their spiritual predecessors, table-top role-playing games (RPGs). Table-top RPGs evolved from war games, which were historical battle re-enactments played with miniatures. Over time, a contingent of war gamers moved away from the strictly historical battles and began to create their own battle campaigns. Some players began to experiment with changing the rule-sets, and others began to focus on single characters (such as the *Chainmail* "Fantasy Supplement" (Perren & Gygax, 1971)) as opposed to an entire unit, and campaigns moved towards a setting with less focus on historical accuracy (Mona, 2007).

In 1974 Gary Gygax and Dave Arneson created what would become one of the first and most well-known rule-sets for role-playing games, *Dungeons & Dragons* (Gygax & Arneson, 1974). *Dungeons & Dragons* (D&D) is set in a fantasy world and has remained the most popular role-playing system to this day (Costikyan, 2007).

Table-top role-playing games are played with a group of people, where one person takes the role of the Dungeon Master or Game Master (DM or GM) who describes the world and story, while the other players create characters that give meaning to the scenario through their actions. As the players move through the world, the DM adapts the story to incorporate the players' actions into the overall narrative. This type of collaborative play involves a constant negotiation between DM and other players to create a story and good game experience with both interesting and meaningful choices.

1.2 Game Master

In table-top RPGs, the Game Master (GM) is the "God" of the game (Fine, 2002). The players' available actions are ultimately defined by what the GM will allow in a game. Because of the critical role the GM has in the experience, each game is heavily influenced by the GM's abilities. At the core, an unsatisfactory GM is incapable of properly negotiating their overall vision of the story with their players' actions. In general, a GM will fall somewhere on a spectrum of story control, with bad GMing defined by the two extremes of the spectrum.

At one extreme of story control is the "railroading" GM. This style of GM has a story in mind and does not allow the players to deviate from it. Players are given no meaningful choices within the game, as their actions have no effect on the progression of the game. When a player offers an action that does not fit the GM's ideal storyline, it is not allowed. This often leads to the players feeling as if they do not have true control over their own character, and can be highly frustrating. A more skilled GM will allow an action, but adapt the story such that the action ultimately does not have a negative effect on the story.

Many humorous examples can be found in "DM of the Rings," a satire comic told as if Tolkien's *Lord of the Rings* was a table-top RPG led by a railroading GM. In one instance, recounting Aragon riding the warg, the player attempts to dismount the warg. The GM has already decided that the player and warg will fall off a cliff, so he disallows the player to dismount, even given the player's dice roll (S. Young, 2006). A more skilled GM would have allowed the player to dismount, even if the GM was guiding the players through a story in which falling off the cliff was absolutely necessary. The GM, for example, could have

adapted the story through environmental changes (e.g. the cliff below the player's feet collapses) or NPC actions (e.g. another warg runs into the player pushing them off the cliff) to lead to the same outcome. This allows the player a feeling of control over their character, even if the world works against them.

On the opposite end of the spectrum of story control is a GM that does not exert any control over the game's shape. With a large world and many options, players often feel lost and unsure of where to go. Without any guidance from the GM, the players will get easily side-tracked or bored with the gaming experience. Experienced players may use their previous play sessions to guide them through the game, but without any constraints, the game lacks focus and interesting choices, and can become an exercise in frustration (Fine, 2002).

Bad GMs are of particular interest to us because they are analogous to prevelant styles of game play within computer role-playing games. This is discussed in more detail in the next section.

2 Computer Role-Playing Games

Throughout the late '70s and early '80s, table-top role-playing games found their way into the computer domain. The well-defined combat rules from *Dungeons & Dragons* easily translated to the systematic nature of computers. Computers had an advantage over the table-top counterparts in that the computer could quickly and easily store all the rules for the game and effortlessly calculate turns within the combat. Instead of a single encounter taking possibly hours; it could be completed within minutes, allowing for more progress in a shorter time.

However, the playable storytelling aspects of table-top RPGs were more difficult to represent computationally, and as such have long been less playable in computer role-playing games. CRPG gameplay instead often focuses on battle, allowing very few player choices outside of those related to combat.

Due to the lack of storytelling support, CRPGs tend towards the two extremes of bad GMing mentioned in the previous section. Many classic CRPGs and Japanese-style CRPGs (also known as JRPGs), such as the *Final Fantasy* series (Square Enix, 1987), have finely-crafted stories which the player is railroaded into playing. While the stories may be grandiose and well-constructed, the player lacks meaningful choices, and is thus merely a character in someone else's pre-arranged story, given no options or chances to influence the narrative. In fact, major elements of such CRPG stories are often presented in non-interactive "cutscenes" (linear animations) driving home the fact that they are a part of the game that cannot be played, only consumed in set form.

At the other extreme are games such as *Oblivion*, (Bethesda Game Studios, 2006) open worlds with a multitude of options for the player, but the overall player experience lacks cohesion and interesting choices for the player. *Oblivion's* designer, Ken Rolston went so far as to say that when designing the game, he kept in mind, "In games, stories suck, so focus on the other elements of narrative: setting and theme" (Rolston, 2009). These shortcomings lead Chris Crawford to point out, "no games have approached what a good DM [Dungeon Master] could do with players around a table" (Chris Crawford, 2003).

How can this be addressed? One option would be to completely eschew CRPG conventions

and develop a new game genre. However, I am interested in extending the CRPG genre—approaching a future in which CRPGs more strongly resemble their table-top counterparts. I am encouraged by the fact that stand-out games within the genre such as *Planescape: Torment* (Black Isle Studios, 1999), *Star Wars: Knights of the Old Republic* (BioWare & LucasArts, 2003), and *Dragon Age: Origins* (BioWare Edmonton, 2009) all have stories that are more playable than games at the polar extremes of the GMing spectrum. They accomplish this through embedding interesting and meaningful choices within their game mechanics and systems of quests, as well as by adding significant branching points along the main narrative.

3 Introducing Choice in Story

Through examination of exemplary CRPGs, I have identified three ways in which interesting and meaningful choices can be introduced into these games. Each method is at a different level of storytelling; ranging from the moment-by-moment stories created by the players at the game mechanic level within the overall narrative, to the overarching designer-specified story which gives meaning to the player's actions. Each level of storytelling is influenced by the others, leading to a rich and deep story experience.

At each level, it is possible to introduce more interesting and meaningful choices to the player, giving the player a more integrated storytelling role, and more closely emulating the experience of a table-top RPG. Briefly, the three levels are the *player level* in which the player's moment-to-moment choices shape their personal role within the story, the *quest level*, in which series of player level choices merge together to create a story arc within the larger narrative, and the *game level*, in which the overall narrative give meaning to the

player's actions. These will be defined in more detail in Chapter 3. There also exists a fourth layer – the world level – that describes the setting and back story of the world in which the player and game inhabit. While it is possible to think of ways to make this level of story playable, it is not the focus of this dissertation.

3.1 Agency

Giving the player interesting choices within story relates strongly to the theory of agency found in game design literature. In particular, Wardrip-Fruin, et al. describe agency as "a phenomenon involving both player and game, one that occurs when the actions players desire are among those they can take (and vice versa) as supported by an underlying computational model" (Wardrip-Fruin, Mateas, Dow, & Sali, 2009, p. 1).

Achieving agency, in this account, does not require enabling players to "do anything." In fact, it is in many ways the opposite. It requires crafting the dramatic probabilities of the fictional world and developing player understanding of the underlying computational system so that the two are in concert with one another.

We see a version of this in traditional tabletop role-playing games. A set of dramatic probabilities are established for characters that connect to the underlying game system. For example, the expectations for classes like Wizards and Clerics – and races like Dwarves and Elves – are in part derived from literary sources (e.g., Tolkien (Tolkien, 1954)) and in part formed by player knowledge of the game system (e.g., Clerics have a wide range of healing spells, Elves are generally better at agility-heavy tasks than Dwarves). These then come into action during gameplay, as players use their characters to attempt actions that are

appropriate both to the dramatic probabilities of the situation and the specifics of their character – one doesn't role-play Aragorn and Gandalf the same way, even if they find themselves in the same situation. The DM facilitates play that is appropriate to the different dramatic probabilities of each character, using the rule system, enabling an experience of agency.

This connects directly to our critique of story in current computer role-playing games. Because the quest and game level story is fixed, the story assumes that the dramatic probabilities are the same for a fighting character, a healing character, and a stealthy character. It assumes the probabilities are the same for a character no matter what actions, quests or decisions they have made in the past, what special abilities they have, and so on. This, in turn, requires that combat be the center of almost every story – because this is one of the few things that every character can do, even if it doesn't always make dramatic sense if we take characters seriously.

We are not the first to note the need for a system that better models the collaborative nature of the DM and players. In *Hamlet on the Holodeck*, Murray mentions that in the future, the challenge in games will be to “capture a wider range of human behavior than treasure hunting and troll slaughtering” (Murray, 1997). Many games are limited to these actions because more complex actions require a more complex gaming system. Adding to this in *Character Development and Storytelling for Games*, Sheldon mentions that video games “need to be able to adjust to the improvisation of players” (Sheldon, 2004).

More specifically, Susana Tosca states, “In computer games, we need to prepare the quest events in a much more fixed way [than table-top RPGs.] The result is less lively and immersive […] To my mind, the success of pen&paper games is precisely in the common creation of a story […] that depends on all participants being human and changing the script constantly” (Tosca, 2003).

3.2 Grail Framework and Mismanor

The Grail Framework is designed as a first step in increasing agency and adding interesting and meaningful choices at three levels of storytelling within games. The system encompasses both authoring tools (including a brainstorming tool and a game design tool) and in-game systems CiF-RPG (*Comme il Faut* for Role-Playing Games) and GrailGM (Grail Game Manager). The Grail Framework in its entirety is designed to create a framework in which the designer is able to author high-level rules together with relatively-atomic pieces of traditional content, allowing for a large amount of dynamism in play. By not requiring the designers to use traditional scripting methods to create the game story, they are able to focus on the tasks of designing and writing, as opposed to programming and scripting.

At the player level, CiF-RPG allows for the development of new kinds of CRPG stories, those built upon social mechanics instead of combat. The social mechanics are used to create quests and game-level stories built around social behavior. Story plot points and quests are designed with content and rules – GrailGM is able to dynamically combine these in new and interesting ways. The designer is able to maintain authorial control over the world via designer goals—similar to a Game Master presiding over a table-top role-playing game. Story focus, structure, and quests—and their possible solution states—are shaped by the

player's own history. Details such as who the player has talked to and the types of relationships they have developed with these non-player characters (NPCs), as well as how they have solved previous quests all are used to inform the shape of the game narrative at three levels.

To create enough content for the player to be able to make these choices requires the designer to create additional content and manage the inter-dependencies within the system. Because of this, we designed author support tools such that designers are realistically able to create the content required for such a system.

The game *Mismanor* was created to explore the capabilities of the Grail Framework. The game was designed to support non-combat play, dynamic quest selection, and multiple re-orderable storylines. While the high-level story framework remains the same through each play, the particular story a player experiences will be different based on their actions throughout the game, which characters they choose to interact with, and how they choose to interact with them.

4 Research Contributions

The primary motivation for my work is to create playable stories within computer role-playing games. Towards this goal, I have made the following research contributions.

I have identified four levels of story within CRPGs, focusing my work on making three of them playable. This is discussed in more detail in Chapter 3.

I adapted relationship-oriented game mechanics to the RPG domain and created the first meaningful connection between relationship-oriented gameplay and story structure through the creation of CiF-RPG. Chapter 4 explains this more thoroughly.

I implemented the first model of goal-based quests for RPGs and demonstrated how they allow players to have multiple paths to completion and meaningful system responses to different completion states using the GrailGM system. The GrailGM system also models intelligent quest selection that can be guided by author-specified values and functions. GrailGM employs a method for the results of player choices in the social mechanics and quest completion levels to meaningfully influence the larger story development and outcome, while not requiring the explosion of authoring effort associated with other approaches to this problem. Chapter 5 details GrailGM and its capabilities.

I built tools that demonstrate that authoring such RPGs is within reach for novice designers without any special background in logic, story generation, or any other domain. The tools created for this project are detailed in Chapter 6.

I have created the first fully-realized DODM game, which has led to insights and improvements to using this approach in games. This is discussed in Chapter 7.

I have created an existence proof for this new approach to story in CRPGs, solving hard problems on both the player and author levels, with the creation of *Mismanor*. In addition, I created a new model of social interaction UI incorporating a player character and exposing the reasons for the AIs decisions in a status line. *Mismanor* and its UI is discussed in Chapter 8.

Finally, I also created a form of evaluation for this new type of playable story, which is described in Chapter 8, Section 5.

5 Dissertation Outline

This dissertation covers the path towards a story-centric future for computer role-playing games. Chapter 2 begins with a survey of the different visions of the future of CRPGs, looking at both industry and academia. It then moves on to discuss where the Grail Framework fits into the future of CRPGs.

Chapter 3 discusses the general approach towards making stories playable, focusing on the four levels in which stories take place within an interactive experience: the moment-to-moment choices a player makes at the player level, the mini-story arcs created by quests and coherency between choices at the quest level, the high level narrative arc that ties the experience together at the game level, and the story created by the setting, world structure and backstory found at the world level. The chapter ends with a discussion of how each level interacts with the others, combining to create a single experience.

Chapters 4 and 5 delve more deeply into each level of storytelling and the systems that were built to deal with them. Chapter 4 focuses on the mechanics of a game, which enables the player level stories throughout the game. In particular, Chapter 4 focuses on how CiF-RPG was used to create non-combat-based stories, and the requirements of the system to support a new type of gameplay. Chapter 5 moves on to discuss the dynamic quest selection system within GrailGM, and how player choice is incorporated into making quest level stories playable. Then Chapter 5 continues by talking about the game level story, and how

drama management-influenced methods within GrailGM are used in the system to create a dynamic narrative.

In the second half of this dissertation, the focus shifts to discuss the games that were created to explore the capabilities of the AI systems, as well as the tools that were designed to help with the creation process.

Chapters 6, 7 and 8 incorporate the practice of creating an experience based on the systems described in the previous chapters. Chapter 6 discusses the design tools created to help ease the authoring burden for such a dynamic system. Chapter 7 looks at the design of the game *EMPath* and how we used the lessons we learned about Drama Management within the Grail Framework. Chapter 8 talks about design and creation of *Mismanor*, and how the design of both *Mismanor* and the Grail Framework influenced each other. Chapter 8 also looks at the authorial expressivity of the Grail Framework within *Mismanor* and discusses the evaluation of the system. Given a number of player models and automated run-throughs, the chapter presents the expressive range of the system.

Finally, Chapter 9 sums up the conclusions from this work, while discussing some of the future implications and directions based on what has been learned.

CHAPTER 2

THE FUTURE OF ROLE-PLAYING GAMES

The role-playing genre encompasses a large number of sub-genres, each with their own focus and possible futures. While this research is focused primarily on single-player computer role-playing games (CRPGs), the future of CRPGs is influenced by traditional tabletop role-playing games (RPGs), as well as massively multiplayer online role-playing games (MMORPGs). It is difficult to talk about the future of CRPGs without at least touching on these other two sub-genres.

1 Tabletop Role-Playing Games

Tabletop, or pen and paper, role-playing games are the provenance of computer role-playing games, so perhaps it is not surprising that they have had some of the same growing pains. Wizards of the Coast, who now owns the *Dungeons & Dragons* (D&D) (Gygax & Arneson, 1974) property, has experienced market pressures similar to those that CRPGs face and therefore have looked to the CRPG and MMORPG markets as target audiences for their new revisions. Mike Mearls, the head of development on *Dungeons & Dragons* said in an interview that 2nd edition *Dungeons & Dragons* (D. Z. Cook, 1995) catered to the Dungeon Master (DM), allowing them supreme control of the story, while 3rd edition (M. Cook, Tweet, & Williams, 2000) swung too much the other direction in which the players had all the control and the DM became the “rules guy.” (Tito, 2011b) This closely resembles the two

extremes seen in CRPGs, railroading (DM has too much control) and open-world (players have all the control) discussed in Chapter 1.

In 4th edition *D&D*, (Heinsoo, Collins, & Wyatt, 2008) Wizards of the Coast attempted to tap into the CRPG and MMORPG markets, and balance for groups with "limited imagination" (Tito, 2011b), or rather, open the door for new gamers coming from a video game background who had less experience with the storytelling aspects. Unfortunately, this punished their core-gamers who have grown up with the franchise, that didn't need or want the rules to cripple them. This caused a large rift within the community, which has left the *D&D* franchise struggling. In fact, *Pathfinder*, (Bulmahn, 2009) which is a tabletop system which extends the *D&D* 3.5 (Team, 2003) rule set, chose to not support the 4th edition rules because the community disliked 4th edition so strongly (Tito, 2011a).

Mearls feels that the future of *D&D* needs to cater to the differing needs of the community, with board games and card games supplementing the main franchise. For their tabletop rulebooks, he feels that a return to the roots, allowing for flexibility for the DM to tell the story they want to tell, along with allowing the players to influence the story is the way to re-invigorate the *D&D* franchise.

There are other visions, however. Ryan Dancey feels that tabletop RPGs are dying, and catering to a diminishing hobby crowd. This is what led Dancey, the former VP of RPGs at Wizards of the Coast and marketing executive at White Wolf to move to the video game industry (Tito, 2011b). Others, such as Erik Mona, publisher at Paizo (creators of *Pathfinder*)

feel that tabletop RPGs will start incorporating more technology, but elaborates that he hopes they will always focus on story first (Tito, 2011b).

The significance of story within table-top RPGs, and its important role in the future is a strong parallel to the future suggested by this research. In attempting to draw in more of the CPRG crowd, *Dungeons & Dragons* put more focus on the combat systems, balancing the game for a "lack of imagination." In the process, they undermined the franchise, because the storytelling aspects are the backbone of the RPG genre, and the core strength of table-top RPGs compared to the other sub-genres.

2 Massively Multiplayer Online Role-Playing Games

Massively Multiplayer Online Role-Playing Games (MMORPGs) are a relatively new genre, but they have grown in popularity rapidly, with *World of Warcraft* (Blizzard Entertainment, 2004) topping out at over 12 million players at its height (Chiang, 2010). Because MMORPGs must support millions of players – each at a different point in their character's development – with ongoing play over a number of years, the game worlds and stories can have no end. This means that MMORPGs are necessarily limited in their storytelling capabilities. The main exception to this is *A Tale in the Desert* (eGenesis, 2003) which has year-long story arcs, with the game ending at the end of the year, and everyone starting anew at the beginning of the year.

Because of these differences, at first glance it would seem that MMORPGs have a different focus for the future than that of the CRPG. However, CRPGs and MMORPGs do often appeal to a similar audience, and therefore, the two genres borrow heavily from one another and

have similar market pressure. There have also been ongoing debates in articles and forums (Wilson, 2009) (Phanthapanya, 2011) (Vash108, 2008) (TigerMyth36, 2010) on whether the single-player experience is losing the battle to MMORPGs due to their larger market share. For these reasons, it seems valuable to survey the possible futures for MMORPGs.

According to the survey by Achterbosch, Pierce, and Simmons (Achterbosch, Pierce, & Simmons, 2007), MMORPG players main request for new features of MMORPGs is for their characters to have an effect on the game world. Because of the nature of MMORPGs, the story of the game world is experienced mainly through the quest system. Player characters are given quests with story importance placed upon the tasks they are assigned. However, the suspension of disbelief is easily broken when the enemies a player kills respawn, or the town they attempt to save stays in the same state of disrepair. The game world is unable to incorporate player-based changes since the player is sharing the world with hundreds or thousands of other players following the same quests. Because the quests do not have an impact on the world, quests quickly lose their story significance, but instead become the gateway to the player receiving the rewards for completion.

There is a long-standing joke in *World of Warcraft* about the inn in Westfall which players have been helping to repair for 7 years through an early-level quest. However, no visible improvement was ever made until the latest expansion, when the inn was fixed as a nod from the development team towards this joke. There are obvious difficulties with making millions of people able to be the heroes of the same world, but there are some games which have begun trying to address this issue.

One of the games that has attempted to address this is the most recent MMORPG by Bioware. *Star Wars: The Old Republic* (BioWare, 2011a) was written in an attempt to bring some of the strong story elements of a single-player CRPG to an MMORPG experience. The team used voice acting for all of their quests and cut-scenes, and the Bioware team employed dozens of writers for a number of years (Remo, 2008) to hand-write quest lines for each of the classes, and to take into account the light and dark side of the force. Many agree that the quest system is well done (Johnson, 2012) (Haas, 2012) (Kolan, 2012), but the players still feel like the game world does not adapt to their actions well enough (Castronova, 2012).

Another game which took a different approach is *Eve Online* (CCP Games, 2003). This MMORPG has a set of fairly standard quests which suffer all the same issues as quests in other MMORPGs. However, the difference is that CCP Games took a sandbox approach to the game instead of a story-based approach. This gives players the freedom to create content for the game, including their own stories that they play a major role in. While this is very compelling for many players, it does lead to frustration for new players who do not have an easy foothold into the game, as the content is mainly created by players who have a long history with the game. Instead, they are stuck in high sec space (high security space in which they are unlikely to be killed by other players) where the only content available is the pre-authored quests.

It stands to reason that the desired future of MMORPGs is somewhere between these two extremes and is very similar to that of table-top RPGs: the DM or game designers able to tell a good story, and the players able to play a role in affecting the story development.

MMORPGs have additional technical and design hurdles compared to a table-top RPG, but the desired future is still the same.

3 Single-Player Computer Role-Playing Games

The successes of MMORPGs as well as the historical ties to tabletop RPGs have both played an influential role in the recent development of CRPGs. Some schools of thought feel that MMORPGs will entirely replace CRPGs, while others feel that the strong story-based nature of CRPGs leave a unique niche for the genre to continue to thrive. Many game industry experts agree that the story in CRPGs is both the strength of the genre and what sets them apart from MMORPGs and other genres (Wilson, 2009). Iwinski, CEO of CD Projekt which made *The Witcher* series (CD Projekt, 2007) sums it up with, ". . . single-player RPGs are all about the story and your immersion and the way you experience it" (Wilson, 2009).

However, what is not agreed upon is what role the story plays within CRPGs. Some industry experts feel that the story in CRPGs should move towards a more cinematic experience, and many technological advances in recent CRPGs reflect this. Pre-rendered cut scenes and large amounts of voiceovers have become the mainstay of AAA title (high-budget and mainstream) CRPGs. While this makes for a more cinematic experience, there are issues with both of these storytelling methods.

Cut scenes are at complete odds with the interactive nature of video games, taking control from the player and forcing the story in the direction the designer designates. Jordan Mechner, designer of many of the *Prince of Persia* titles including *Sands of Time* (Ubisoft Montreal, 2003), points out that game's story is there to serve the game play, not the other

way around. To this end, the biggest moments of the story in *Prince of Persia: Sands of Time* were given to the player to act through, instead of sit through (Mechner, 2007). While *Prince of Persia* is not a CRPG, it is an excellent example of story being told through game play instead of cut scenes.

Voiceovers have become more prevalent within CRPGs, and even BioWare's latest MMORPG, *Star Wars: The Old Republic* (BioWare, 2011a) relies on voiceovers to act out the dialogue for quests. While voice acting is believed to increase immersion, it has had an unwelcome side effect. After adding voiceovers to *Dragon Age: Origins,* (BioWare Edmonton, 2009) there was pushback from the users because the main character did not have voiceovers. The developers responded that this was because the main character could be whoever the player wanted to be, and of course they could not assign a voice to that (Wilson, 2009). Because of the pushback, in *Dragon Age II*, (BioWare, 2011b) the main character was given voiceovers, along with a name and a more defined character role. Similarly, in BioWare's *Mass Effect* series (BioWare, 2007) has a pre-named character from the very first game, and the player's character has voice acting throughout the series. This is an example of how technological advances actually hinder improving the interactive storytelling capabilities of CRPGs.

Instead of focusing on creating a more cinematic experience with cut scenes and voiceovers, progressing technology could instead be used to improve the interactive characteristics of storytelling within CRPGs.

One step towards this future is the use of downloadable content (DLC) and patches that extend the narrative of a CRPG, as well as change the story and game based on player feedback. Iwinski feels that the always-online nature of games can take advantage of DLC and patches more extensively in the future (Wilson, 2009). Companies use DLC as a method of allowing players to be a part of the feedback loop within the game; however it means that players all receive the same game changes, regardless of their individual desires. Additionally, developers must find consensus within the player community to incorporate the player's feedback. While this is simple in some cases, in many cases players differ enough that it is impossible to please anyone outside the simple majority.

Emily Gera, a journalist for VideoGamer.com, states that the future of RPGs is about liberating the player from the avatar (Gera, 2011). Gera goes into details about how the move towards a classless system within *Skyrim* (Bethesda Game Studios, 2011) allows players the freedom to create whatever type of character they'd like to play. Classes limit a player's choice within combat, and allowing players to not be constrained by a choice they made before even playing the game allows for it to be much more immersive. She goes on to say that "[b]reaking from the restrictions imposed by a game, that's role-playing. Erasing the lines between where you stop and the character starts – *that's* role-playing" (Gera, 2011).

However, a classless system for character creation only gives the player more control of their character when involved in combat. Giving the player interesting and meaningful choices within a story is the real way towards "liberating the player from their avatar." By creating AI systems that can react to each player, and modifying the story accordingly, there

need not be player consensus, a long patch cycle, or non-interactive storytelling. Allowing the player to customize how their character acts, not only for the base game mechanics such as combat, but also in how they interact with the story, unlocks a whole new depth within CRPGs. *That's* role-playing.

4 Interactive Storytelling

To truly unlock the potential of stories within CRPGs, the stories must become playable; the player must have interesting and meaningful choices within the story structure of the game experience. Without requiring the designers to create an untold number of stories that a player might experience requires the use of some form of generation. Generation used for story often falls under the umbrella of interactive storytelling research. The research in interactive storytelling has several related research paths of interest to the future of CRPGs. While much of the work is not specific to computer role-playing games, it is still relevant given that the goals are often quite similar.

4.1 Narrative Generation

Narrative generation encompasses systems that create game level stories. These generation systems can use off-line methods – in which the story is created in whole, before it is experienced – or reactive methods which create stories as the game progresses in response to the player's actions. For story playability, reactive methods are necessary at some level since the story needs to adapt to player choice. There are two threads of dynamic narrative generation research that are of particular interest to CRPG work: scene-based storytelling, and character-based storytelling.

4.2 Scene-Based Storytelling

Scene-based storytelling uses pre-authored components of a story and these components (sometimes called scenes) are dynamically arranged at run-time to create a narrative. The dynamic arrangement is accomplished using a story management system that chooses a story component in response to the player's actions.

Story managers use a number of different methods to choose one story component over another. Search-based drama management systems, first proposed by Bates (Bates, 1992) and developed by Weyhrauch (Weyhrauch, 1997) search possible paths in the game tree, assigning a score to each outcome based on author preferences. The drama management action that leads to the highest weighted outcome is chosen.

The Mimesis architecture (R. M. Young & Riedl, 2003) is built to bridge the gap between an interactive experience (such as a game) and planning methods that are usually used as off-line applications, such as the planning approach used by the Universe system (Lebowitz, 1985). The drama manager within Mimesis uses a planning algorithm to create story plans and monitors for player actions that might threaten causal links in the story plan. If a threat is a detected, the manager re-plans or prevents player action.

Marlinspike, (Tomaszewski, 2011) used for the game *Demeter*, similarly uses a content-selection model to choose a scene based on the player's actions. The scenes are weighted based on the Aristotelian notion of completion and unification; that is the story should have a beginning, middle, and end, and the events within the story are connected by a cause. Marlinspike uses the improv technique of reincorporation to refer back to earlier events to provide unification.

All of these methods are helpful in that they provide a means for giving the player interesting and meaningful choices at the game story level while still retaining some amount of authorial control of the story.

4.3 Character-based Storytelling

Coming from the other direction is character-based storytelling, a term first coined by Cavazza in 2002 (Cavazza, Charles, & Mead, 2002). Character-based storytelling refers to a character-centric approach to interactive storytelling, or how the game level narrative evolves based on the player's interactions with the non-player characters at the player level. Pizzi (Pizzi, Cavazza, & Lugrin, 2007) extended Cavazza's initial project to take into account the character's psychology to influence the overall narrative. The characters in the story have long-term motivations which are used for goal formation, specified by the feelings and desires of that character, rather than describing a particular state to achieve.

Figueiredo uses a character-centered approach, but adds the concept of a story facilitator (Figueiredo, Brisson, Aylett, & Paiva, 2008). The story facilitator is an architecture based on the role of an improvisational facilitator, similar to the dungeon master in table-top role-playing games. The story facilitator is told the roles of each of the characters, and is given a set of narrative actions with which the system can manipulate the story to achieve the narrative goals. The system is similar to the drama management systems mentioned above, except that the triggers for story facilitator intervention and the actions to be performed are designated by the author at design time.

Generally, character-based storytelling methods provide a tighter link between the player level and a game level story, as the story is responding much more closely to the player's actions. However, the trade-off is that there is less guarantee of story coherence.

5 Quest Generation

Specifically looking at quest level story, quest generation systems either create quests dynamically during game play, or work off-line to help designers create the quests. There are currently only a handful of systems dealing with quest generation.

Charbitat, a game system consisting of generated terrain tiles with randomly placed components based on player actions, was expanded to include a quest generation system that worked in conjunction with the terrain generation (Ashmore & Nitsche, 2007). *Charbitat* implements lock-and-key style quests based on spatial progression through the world. As the level is generated, a quest can be created on a new tile which uses the world state as context for the goal of the quest.

Grey and Bryson (Grey & Bryson, 2011) suggest an agent-based approach to quest design, in which the NPCs are able to observe actions taken in the game, remember specific events, and communicate the events to other NPCs. The events themselves are then used as the justification for giving a player a particular quest. This provides a feedback system for the player's actions to have a meaningful impact on the quest level story.

Doran and Parberry (Doran & Parberry, 2011) have created a prototype system for generating quests offline. The system takes an NPC motivation, such as *Protection* or *Conquest* and randomly chooses a strategy and action sequence associated with fulfilling

that particular motivation. Many of the actions can themselves be replaced by sub-quests that are made up of the same set of actions. This allows a quest to be generated of arbitrary length and complexity. It seems possible for this system to be extended to work during game play, reacting to NPCs changing motivations during the game.

These systems all work to add more dynamism to the quest level story, and allow for the systems to respond to player input.

6 Conclusion - The Grail Framework

The theme of player influence on story is an issue that continually arises for table-top, single-player and massively multiplayer online RPGs. In tabletop RPGs, *Dungeons & Dragons* faces the challenge of balancing DM storytelling capabilities along with player-influence on the story. This mirrors almost exactly the challenges that CRPGs face with the players wanting more control over the story, while designers want to be able to tell a good story. MMORPGs face the same issue, but on an even larger scale and with more technical and design hurdles to master.

If, as Iwinski states, computer role-playing games are truly to be all about the story, the player's immersion, and how the player experiences the story, then creating stories that can adapt to how a player interacts with the game is the most compelling future of them all.

The Grail Framework was designed and created with this future in mind. At the player level, CiF-RPG supports social mechanics which expand the types of stories CRPGs can tell. Table-top RPGs and MMORPGs have shown that there are players who are interested in non-combat mechanics. By providing a new type of gameplay, the player is given more

interesting choices, one of the requirements for playable story. A more detailed explanation of CiF-RPG and its capabilities is provided in Chapter 4.

GrailGM is the quest and story management system within the Grail Framework. Named after the game masters that preside over table-top RPGs, the purpose of the GrailGM is to give meaning to player's choices, the second half of playable story. The quest generation system uses a character's social feelings for the player to choose which, if any, quests to offer the player. The story management system uses content selection without search due to the large number of possible actions. The possible content is weighted based on author-chosen story heuristics, and the scene the player experiences is based on the characters they choose to interact with, and how they are interacting. Chapter 5 discusses the capabilities of GrailGM in far greater detail.

In these ways, the Grail Framework works at three of the levels of storytelling to provide interesting and meaningful choices to the player moving the CRPG genre one step closer to real player immersion within a playable story.

CHAPTER 3

LEVELS OF STORYTELLING

Computer role-playing games are consistently referred to as story games, but it is important to know what is meant by story. At first glance, "story" refers to the overarching narrative that the designers create and the player experiences. However, this glosses over a number of smaller story arcs the player may encounter throughout the game, when talking to NPCs or fulfilling quests. Even including this level of detail, it still misses the stories the player creates through their choices within the game mechanics themselves.

Instead, I posit that story within CRPGs is happening at four distinct levels: *world, game*, *quest*, and *player* levels. When all of these levels work in tandem, they create a cohesive narrative experience. The type of story is often quite different at each of the levels, both in structure and content.

The world level story is the setting and fiction of the world that the game takes place in. This often includes a backstory that has been created by the designer to flesh out the world and give a sense of history. This level of story is most often completely static as it describes things that have already happened which are presented as facts about the game world.

The game level story is the overarching story throughout the game. At this level, the designer has the most input into the story, choosing the genre and tone of the story which will be pervasive throughout the game. The player also tends to have the least amount of

input, with game level story nuggets often used as rewards for the player working through the experience.

At the quest level, the story is made up of a series of smaller arcs found in quests. A player has control over the outcome of these arcs, as well as some control over how the arc progresses, but generally the designer has control over the goal of the story arc, and the reasons behind it.

The player level stories are made up of a sequence of moment-to-moment choices that the player makes, with the choices being constrained by the mechanics of the game. While there are a large number of choice sequences, the player stories are often those that the player feels are interesting or meaningful enough to be worthy of retelling. At this level, the player is given the most freedom to choose within the constraints of the mechanics available.

By examining each level of storytelling within CRPGs, it is possible to find different methods to improve agency and playability.

1 Player level

The player level of storytelling within CRPGs is defined by a sequence of moment-to-moment decisions a player makes within the constraints of the game mechanics. In many CRPGs, these actions include movement, talking to non-player characters (NPCs), manipulation of game items (e.g. breaking a barrel, using a key to unlock a door, opening a letter in the characters' inventory), and combat actions (e.g. casting a fireball, swinging a

sword, or defending an attack with a shield). These mechanics define the possible space of player level stories that a player can tell.

While not every sequence of actions creates a compelling player story – very few people would find a story about breaking a barrel and picking up what was inside compelling – the interestingness and meaningfulness of the sequence of the events is primarily from the player's point of view. While the barrel breaking story may not be interesting to many players, if the player had been looking for a particularly rare artifact, and it happened to fall out of the barrel, it is now an interesting and meaningful story to the player.

Chris Crawford describes the moment-to-moment choices of a player as the basis of interactive storytelling (C. Crawford, 2004). However, Crawford uses this to describe a story world in which these choices created the entirety of the game level story. This is not the approach we are suggesting for CRPGs, but rather the player level stories are interesting unto themselves.

Will Wright said in his talk at SXSW in 2007 (Wright, 2007), "Players invariably come up with stories about what they did in games. They're never describing a cut scene. I categorize these as Unintentional, Subversive and Expressive." The categories Wright describes are in a sense, the types of stories that players find meaningful or interesting. The unintentional story is one in which the player encounters a bug and expresses it in story form. The example given in the keynote is of the spontaneous combustion bug in *The Sims* (Maxis & The Sims Studio, 2000).

Subversive stories are ones in which the player attempts to push the boundaries of the game, by finding cheats or exploits. Finally, expressive stories are those in which the player uses the game to express their own story. *The Sims* became a storytelling tool for many expressive stories, perhaps the most famous being the moving tale of a homeless family followed in the blog, Alice and Kev (Burkinshaw, 2009). Machinima is the more general form of using games to create expressive stories.

However, not every player story fits cleanly into one of these categories. For instance, our barrel breaking story is neither an exploit nor a bug, nor is it a truly an expressive one. What makes the barrel breaking story something worth remembering or even retelling? Schank provides the answer when describing story. "[…] stories are especially interesting prior experiences, ones that we learn from" (Schank, 1995). The barrel story is interesting, because it was such a surprising event, and humans learn more from surprising events (van der Spek, van Oostendorp, & Ch. Meyer, 2012). In this case, the player has now learned that sometimes nice and desirable items can come from destroying barrels.

Recounting game stories is something that many players enjoy, as Will Wright states. This is particularly prevalent within the RPG genre (CRPGs, MMORPGs, and table-top RPGs all share this phenomenon) perhaps due to the player having a larger role in the character they are playing. Websites such as Obsidian Portal (Wedemeyer, Felton, & LeNeave, 2007) and forums such as Giant in the Playground (Jelsoft Enterprises, Ltd, 2000) have dedicated areas for players to post their stories from their gaming adventures.

Stories of player character progression or development is a common player level story shared by gamers. These stories involve the player making choices about which ways they should develop their character which has an impact on further game play. Players often have different ways they develop their characters, which leads to differing levels of accomplishment at higher levels. Many players trade these stories to try to learn the best way to maximize the abilities of their characters, which fits with Schank's story definition, in which players learn from their experiences at character development.

One of the limiting factors for CRPGs as compared to table-top RPGs is that CRPGs contain far fewer possible mechanics than table-top RPGs. This in turn limits the types of stories a player can tell with their actions. For instance, in *Dragon Age: Origins*, the first boss battle with an ogre at the top of the tower is surprisingly difficult, and when discussing the game, players begin recounting their methods of beating the boss. One player bought a large number of potions to replenish mana so that she could continually cast a particular spell which made the battle easier. Another player uses the barrels in the terrain to hide behind for the ogre's special moves so that the party took less damage. The reasons these stories are interesting to players is because each player has a unique story of how they defeated (or didn't defeat) the ogre. Many CRPGs use combat as the main method of solving a problem, while a table-top RPG allows anything the player can think of and convince the DM to allow. Obviously, allowing as many options as a table-top RPG is not within the scope of current CRPGs, however there is room to allow for more player stories than combat-based ones.

There is plenty of evidence that there are players who are not interested in combat-based stories. In the casual game market – which is primarily women (Association, 2007) –

adventure games is a growing genre partially because of the strong story elements. These games generally involve stories revolving around mystery (e.g. *Mystery Case Files* series (Big Fish Games, 2005), *Nancy Drew Adventure* series (Her Interactive, 1998)) or grim fairytales (e.g. *Drawn* series (Big Fish Studios, 2009), *Dark Parables* series (Blue Tea Games, 2010)) and entirely lack combat mechanics. Additionally, in MMORPGs, differences in player types (Bartle, 1996; Yee, 2006) and player expectations show that there is a desire for other options (Tychsen, Tosca, & Brolund, 2006). There are players who devote their in-game time to crafting, trading, or selling instead of combat-oriented activities. Because these games support non-combat play, players are able to pursue non-combat player stories. Unfortunately, most single-player CRPGs do not give players these options.

This can cause a lot of frustration for players. In a table-top RPG, it is possible to role-play a priest who refuses to shed blood or harm another living being, which creates the possibility for very interesting player level stories. In a CRPG, the same player would not make it past the first encounter, as mechanics for such behavior are not supported. The first step is then to begin supporting non-combat mechanics.

1.1.1 Social Mechanics

While there are a number of possible player stories that could be supported, many CRPGs involve inter-personal dynamics, and therefore creating social mechanics would positively affect a large number of story spaces. To do this, we first looked at how a player interacts with the other NPCs in the game.

The current state-of-the-art character interaction involves a player either choosing one of a series of lines of dialogues, or choosing a particular mood that they wish to convey within the dialogue. Some of these dialogue options can lead to limited branching points within the overall story, but the majority of the options have no effect on the game level story. This can break a player's sense of agency because the game implies that the player has a meaningful and interesting choice, when really they do not. This also lacks the complexity of a combat system, in which a player is able to strategically think and plan through their options based on the enemy they are fighting, and their character's abilities.

1.2 Visual Novels and Social Sims

Games do exist in which social interaction takes a key role in the game mechanics. In particular social simulations such as *The Sims* (Maxis & The Sims Studio, 2000) have complex models of social state. The player has a number of actions available which change the world and social standing with other characters. However, interactions with other characters are simplified to stat changes, as the dialogue is all spoken in *Simlish* which is nonsense sounds. Because there is no dialogue, the actions that can be taken in *The Sims* are fairly coarse, such as talk, flirt, and entertain. With dialogue, it is possible to create much finer grain actions such as encourage, physical flirt, seduce, etc. While *The Sims* does model the social space between characters, it is really more about modeling the individual characters. Finally, as denoted by the name, *The Sims* is a simulation game, and therefore does not have very strong quest or game level story. Most of the story is created at the player level.

Dating simulations are specialized social simulations in which the player character attempts to improve their romantic standing with a choice of non-player characters. While this

research is interested in looking at more than romance, dating simulation mechanics have been successfully integrated into CRPGs. In particular, the *Harvest Moon* series (Marvelous Interactive, 1996) and the *Shin Megami Tensei: Persona* (Atlus, 1996) series are both CRPG titles which have incorporated dating sim mechanics. In the *Harvest Moon* series, the player can gain favor with different characters by speaking with them and finding gifts that the particular character likes. Once the character favors the player character enough, the characters can get married. At specific times during the game, the player will have dialogue options which will have impact on the relationship between the characters. This system is very similar to the romance system found in the *Dragon Age* series (BioWare Edmonton, 2009), except that the romance options play a larger part in the *Harvest Moon* games.

In *Shin Megami Tensei: Persona* games, the player character creates social links to non-player characters. These links are improved during interaction with the appropriate character. The player is often given dialogue options which have varying effects on the level of improvement on the social links. In turn, the social links strengthen the *personas* which are used for specific skills during combat. This gives the player strong incentive to improve the social links, but the social system feeds into the combat system instead of into itself. For instance, a stronger social link with a character does not change future possible social link opportunities.

Visual novels are another genre which is very focused on the social interactions between the player character and non-player characters. However, these are at the opposite end of the spectrum from social simulations in which the gameplay is primarily following a game level story with a few key places in which the player is given an option that will branch the story

and possibly lead to a different ending. Because of the heavy designer influence on the story, these games are able to deal with difficult topics such as social privacy as in the game *Don't take it personally, babe, it just ain't your story* (Love, 2011) or physical disability as found in *Katawa Shoujo* (Four Leaf Studios, 2012).

Instead of these two extremes, the Grail Framework provides a social mechanics system which resides in the middle – allowing the player to have control at the social mechanics level, but also having impact on a designer-created quest and game level story.

The Grail Framework provides a social mechanics system in which the player is able to choose what type of interaction they would like to have with an NPC. Additionally, characters are described by a series of traits that effect which interactions they are offered, and how they respond to the various interactions. Each interaction also has an effect on the social state of the game world, similar to combat actions affecting the current battle. This means that a player is given interesting and meaningful choices in how they create their character and how they interact with other characters.

By creating a set of social mechanics, the player is offered the ability to create social-based player stories, with truly interesting and meaningful choices. Because of the focus of characters within CRPG stories, this provided a large amount of expansion of the current systems. For instance, with social mechanics, player stories may now include sequences such as a player convincing their romantic interest to leave their current, but damaging, relationship behind, and instead date the player, or healing an old emotional wound to create harmony within a family.

2 Quest Level

The quest level of storytelling within CRPGs is found in the story-arcs encompassed by quests and quest chains. Quest level stories fit between game level and player level in that there is a designer created beginning and end, with the player providing the middle of the story through their actions. Therefore, there is often a more formal story structure than that found at the player level, but less than one found at the game level.

For this level of storytelling, it's useful to look at Ryan's dimensions of narrativity (Ryan, 2001) (shown in Figure 1) to define the nature of story at the quest level. Quests are given by NPCs within a game world (rules 1, 4, 5), and the player fulfills the quest by taking action which changes the game world (rules 2, 3), and is then rewarded for their actions (rule 6). MMORPG quests fail rule 7 ("The occurrence of at least some of the events must be asserted as fact for the story world.") in particular, but CRPG and MMORPGs both tend to fail rule 8 ("The story must communicate something meaningful to the recipient.")

As discussed in the previous chapter, MMORPG quests offer story descriptions, but the actions the player takes in the world towards quest completion are rarely asserted as fact in the game world. Enemies re-spawn and story consequences such as "fixing the inn in Westfall" are never realized in the actual game world.

Spatial Dimension	1. Narrative must be about a world populated by individuated existents.
Temporal dimension	2. This world must be situated in time and undergo significant transformations. 3. The transformations must be caused by non-habitual physical events.
Mental dimension	4. Some of the participants in the events must be intelligent agents who have a mental life and react emotionally to the states of the world. 5. Some of the events must be purposeful actions by these agents, motivated by identifiable goals and plans.
Formal and pragmatic dimension	6. The sequence of events must form a unified causal chain and lead to closure. 7. The occurrence of at least some of the events must be asserted as fact for the story world. 8. The story must communicate something meaningful to the recipient.

Figure 1: Ryan's dimensions of narrativity.

Furthermore, within MMORPGs, quests are offered as quest level stories, but involve no real player choice other than whether they complete the quest or not. The story feels secondary and inconsequential. When *World of Warcraft* was first released, the quest text would fade in, with the “accept” and “cancel” button not available until the text had all appeared. This was done to simulate the quest being told sequentially to the player. However, the quest text was so often seen as unnecessary that one of the first popular mods created for *World of Warcraft* was used to force the quest text to appear immediately. In later releases, Blizzard removed the fade in, and began to include quest summaries that explained the

quest tasks and reward so that players would not need to find the information within the story wrapper. This is a strong indication of a failure at rule 8 – the story is not communicating something meaningful to the player.

In contrast, in table-top RPGs, there is a great deal of latitude in how a player can choose to fulfill the quest goal, as their choices in game mechanics are much wider, and the human DM can easily adapt the story and results to fit the player's actions. The player therefore has choices which are stated as fact in the game world (rule 7) and the story is meaningful to the player (rule 8).

In CRPGs, quests have a stronger story component than found within MMORPGs, with quests often being used to gate the game level story (C. Crawford, 2004). Because the game world caters to only one player, the quests do lead to changes stated as fact in the game world; creatures do not need to re-spawn for other characters, nor does the game world need to remain static for new heroes starting the same quest. However, the quests themselves have few choices within them and often no effect on the game level story other than allowing it to continue. This leads to storylines which are often more meaningful to the story creator than the player, or at best, only meaningful to players as a passive complement to the game, rather than an active part of play.

There are a few exceptions to this trend. For instance, in the Arl of Redcliffe quest line in *Dragon Age: Origins*, the player is given the chance to save a Blood Mage, an enemy of the state. If the player chooses to save the mage, they are given an extra option at the end of the quest line. The options have a strong effect on how Alistair, one of the party members,

feels about the player's character, possibly leading to Alistair leaving the party if the player chooses an option which makes Alistair upset enough.

Players find this type of choice compelling because it is an interesting choice (empathy vs. justice) with a meaningful impact on the story (an option for how the quest line completes, and how Alistair feels about the player character). However, these examples are few and far between. Most quest lines in CRPGs have combat-based tasks (including *Dragon Age: Origins)*, with no other option for completion. This leads to a case where the designer control of the quest level is overly strong, and the player ends up feeling stifled.

As a counterpart, there are also examples within the CRPG genre in which the player is given a large number of quests to choose from. This is common in open-world or sandbox style CRPGs. At first glance this may be seen as a good thing. However, while the player has choices, they are not interesting (the player does not know which is better than the other) nor meaningful (choosing one over the other does not have an impact on the world) and therefore it often ends up just overwhelming the player and not improving agency.

2.1.1 Quests

To address the quest level story, it is important to look at quests and the quest system within CRPGs.

Jeff Howard describes a quest as "a goal-oriented search for something of value" (Howard, 2008). The quest for the Holy Grail in Arthurian stories is a familiar example of a legendary quest. This style of quest is often used within table-top RPGs by DMs to give a general direction for player actions. In a table-top RPG, the quest does not need to be followed, or

the players can choose to solve the quest in a multitude of ways. For instance, a quest to overthrow a tyrannical king can be solved by combat (kill the king), subterfuge (convince the king's followers to dethrone him), diplomacy (work with nearby kingdoms to remove the king from power), or any other way the players can imagine – and convince the DM to allow.

Taking the lead from table-top RPGs, computer role-playing games also use quests as a staple in gameplay (Howard, 2008) as we touched upon in Chapter 2. Quests are often used to direct the player through the game's story, or to give meaning to the player's actions. On the surface, this is very similar to the quest structure in table-top role-playing games, where the DM often provides at least a main quest that the players use to direct their actions. However, in a CRPG, there is often only one way to fulfill a given quest, with a combat-based solution being the most prevalent. To that point, Aarseth (Aarseth, 2005, p. 503) states that quests in CRPGs "force players to experience the game world, to go where they have not gone before, and barely can. The quest is the game designer's main control of the players' agenda, forcing them to perform certain actions that might otherwise not have been chosen, thus reducing the possibility space offered by the game rules and the landscape."

This is a very different take on quests from that of Howard. Aarseth further points out that, "Such enforced spaces and quests may be used to convey information that may pass as stories, but these 'stories' are not co-told by the players, only uncovered and observed by them" (Aarseth, 2005, p. 504) Here Aarseth is pointing out that the player is merely a passive participant in the quest level stories in many CRPGs. By giving the player interesting and meaningful choices at the quest level, players are becoming more active participants and increasing the possibility space of typical quests.

In her work exploring MUDs (Multi-User Dungeons, a text-based precursor to MMORPGs), Tronstad (Tronstad, 2001, p. 3) describes the pursuit of a quest in terms more closely related to Howard, “To do a quest is to search for the meaning of it. Having reached this meaning, the quest is solved.” When looking at computer role-playing games, Tosca (Tosca, 2003) responds to this by stating, “But quests in computer games are very often devoid of any search for meaning (take this letter to the merchant!), there is no meaning to be sought, nothing to be known, but something to be done.” Likewise, in Karlsen’s (Karlsen, 2008) work comparing the *Discworld MUD* (Bennett et al., 1991) to *World of Warcraft* (Blizzard Entertainment, 2004), he states, “In contrast to this, the point of doing quests in World of Warcraft is to reach the reward. The quests are not so much intellectual challenges as logistical ones." This dichotomy is due to the more expansive set of mechanics available within MUDs. Player and quest level stories are limited by the mechanics available to the player; therefore MUDs have more latitude for quests and do not have the same reliance on combat that many CRPGs and MMORPGs have.

There are games that contain quests which are exceptions to the critique, especially the stand-out games such as *Planescape: Torment* (Black Isle Studios, 1999) and *Star Wars: Knights of the Old Republic* (BioWare & LucasArts, 2003). Both *Planescape* and *Star Wars*, for example, integrate non-combat solutions for a selection of quests, allowing players to choose whether to take a traditional combat role or to fulfill one of the other supported solutions to complete the quest.

But, given current tools, such multiple-solution quests are burdensome to implement and highly bug-prone. In our interviews with quest designers at Sony Online Entertainment, we

were told that—using their in-house tools—the effort required to implement quests with multiple solutions was not additive, but exponential. Similarly, Wardrip-Fruin has analyzed BioWare's Aurora editor (BioWare, 2002a), showing how it reifies quests as a series of milestones, requiring those with other aims to work against the system's organization, leading ambitious RPGs toward narrative breakdown (Wardrip-Fruin, 2009). Wardrip-Fruin uses an example in *Star Wars: Knights of the Old Republic,* but we see it in many other CRPGs as well. For example, in *Morrowind*, (Bethesda Softworks, 2002) quests exist in which the player has the choice to kill a specific NPC or find another non-violent solution. Regardless of the choice made, the player can later receive a quest which requires them to talk to the NPC which they may or may not have killed. This not only demonstrates the bug-prone nature of this approach, but also shows that the choice the player makes does not affect the underlying storyline.

Quests are used mainly to give thematic meaning to the supported player actions – typically fighting with enemies scattered throughout the world – and to move players towards areas suitable for their level. Quests are popular in many types of RPGs; however online worlds generally have thousands of quests to examine, which is significantly more than most single-player RPGs. While I am going to briefly focus on quests in the extremely popular MMORPG *World of Warcraft* (*WoW*), (Blizzard Entertainment, 2004) the issues we discuss are common across single-player and multi-player CRPGs.

There are a number of quest taxonomies suggested for *WoW*; we combined and adapted the systems available to make the following taxonomy (Karlsen, 2008; Oh & Kim, 2005; Walker, 2007)

1. Kill X number of enemies (where X may be 1 or more, and the enemy may be unique)
2. Kill enemies until X number of a specific item drops (where X may be 1 or more)
3. Collect X number of specific items from the environment (where X may be 1 or more)
4. Deliver an item to a specific NPC.
5. Talk to someone specific.
6. Escort someone.
7. Use a special ability.

As this taxonomy begins to highlight, in *WoW* the majority of the quests and experience points received are related to activities that revolve around killing. To illustrate this further, we examine the second expansion of *World of Warcraft: Wrath of the Lich* King, (Blizzard Entertainment, 2008) which was released in late November 2008. One of the first regions (called zones) that a player can choose to explore is Howling Fjord. In this zone, there are 133 quests available to an Alliance player. Breaking down these quests into the above taxonomy and then weighting the categories based on the experience received for doing each quest illustrates the continued combat-centric game design.

Of the 133 available quests, 55% of the quest experience comes from straight kill-based tasks of the type “Kill X number of enemies,” “Kill a specific named enemy,” or “Get X number of items from killing enemies (drops).” Furthermore, 22% of the experience is received from collection-style quests, most of which require collecting items from areas that are infested with enemies. This is problematic because the game implies that the player has

choices based on race, class, and play style, but in fact the way the quests are designed, the player's only real options are within combat mechanics.

This is compounded by the fact that the only ways to gain experience in *World of Warcraft* are to complete quests, kill enemies, or discover new areas. The experience received from discovering new areas only accounts for a minute portion of the overall experience received by a player. Therefore, the major ways to progress through the game are through combat.

The combat-centric nature of quests is in part because quests must support the mechanics available to the player. However, there is also an issue that players do not have any interesting or meaningful choices for the majority of quests within CRPGs. Adding new mechanics as suggested in section 1 solves part of the problem, but unless quest structures incorporate player choice, they are merely dictating the designer's preconceived notions for how a quest should be solved.

2.1.2 Task-Based vs. Goal-Based Quests

A striking feature of the style of quest described above is how different they are from the definition provided by Howard. In his description, the key concept is that of a "goal-oriented search." In contrast, most contemporary CRPG quests present the player with a checklist of actions to take to complete the quest. The goal is to complete the list; there is no player choice involved other than whether they choose to complete the quest or not.

In such quests, which we refer to as *task-based* quests, the list of tasks provides the player with the collection of non-optional actions that must each be completed in order to complete the quest. In contrast, a *goal-based* quest presents an end point, or goal, for the

player to achieve. The player chooses, within the constraints of the game world and mechanics, how to reach the given goal.

An example of a task-based quest that could be found within many RPGs is the quest to save a farm from wolves. A farmer will give a player a task of killing the wolves to save the farm. The player must kill the wolves in order to receive the reward for completing the quest, regardless of class or role-playing preferences. In contrast, a goal-based quest would simply explain to the player that there are wolves killing the local livestock. The player would then be allowed options for completing the quest. They could still kill the wolves if they choose, but other options would be available such as creating better fencing or helping restore the local deer population so the wolves no longer have to hunt livestock. In this way, the player is able to make an interesting choice (no choice is clearly better in an absolute sense than the others) perhaps based on game-defined class or race, or on personal role-playing preference. With the inclusion of these choices having a meaningful effect on the game (e.g. different quest rewards, game world evolution, or even affecting future quest and solution availability) the quest level story becomes playable.

While there are some choices within even a task-based quest (e.g. how a player might choose to battle the wolves), they are being limited to a strict subset of game mechanics available to them. In *World of Warcraft,* players receive quests that might involve collecting environmental objects to create a potion which put into a water source saves a dying population, or the player might be told to kill some number of NPCs or monsters. However, the choice is always at the designer level, not the player level. The game obviously supports both types of play (as there are quests for both), but the player does not get to exercise

their own creativity or preference when completing the task.

The existence of player choice in how quests are completed is the key difference between goal-based and task-based quests. Early adventure games such as *The Secret of Monkey Island* (Lucasfilm Games, 1990) and *King's Quest* (Sierra Entertainment, 2008) games often presented the puzzles within the game as a quest for the player to fulfill: "Become a pirate," "Save the mayor," "Save the land." While on the surface these seem like goal-based quests, there was in fact no player choice involved in quest completion. This could easily lead to frustration where the player would be searching for the solution the designers had chosen for each quest, even though there were other options that made sense to the player but were not supported.

For example, in *Monkey Island* there exists a deceptively simple goal of getting past some deadly piranha poodles. At this point in the game, the player has become a sword master, but they are not allowed to fight the poodles, they must go find the exact item required to pass the dogs. The player has a chunk of meat, but this alone is not exactly what is required. Using the meat with grog (potent alcohol) does not work, but instead the player must eventually realize that they need to use the meat with a small flower they (hopefully) found while walking through the woods to drug the meat. Until the player stumbles upon this solution, they cannot progress further in the game.

These frustrations are due to these quests being presented as goal-based quests, when, in fact, they are task-based quests with opaque specifications. A true goal-based quest allows for interesting player choice, where there are multiple ways to fulfill the quest, and one

solution is not obviously better than the others.

Presenting a task-based quest as goal-based is still present in many CRPGs today. For instance, within *Dragon Age: Origins* or *Oblivion*, the player is given large goal-based quests, such as obtain the assistance of the Arl of Redcliffe in *Dragon Age*, or recover the amulet of kings in *Oblivion*. This is masked much more effectively, however, because the player is presented with the goal-based quest, then given smaller task-based sub-quests. However, the player still lacks the choices that the game implies that they should have.

Creating a true goal-based quest system requires a GM (human or computer) that can understand the world well enough to accept whether or not a player's proposed solution is reasonable. While this is a worthy goal, for our work, we instead had the designer think about different ways a quest might be fulfilled, and designate specific outcomes for each quest goal state. This gives players some choices within the quest, and by having outcomes based on how the quest was fulfilled, the choices become interesting and meaningful.

3 Game level Stories

The game level of story encompasses the overall, main story arc throughout the game. The vast majority of role-playing games follow the structure of Campbell's monomyth or "Hero's Journey" (Campbell, 2008) in which the player is put in the role of the hero. The game begins with the player character having a typical day, when an event happens which calls them to adventure. Through the adventure, the player character ventures into the unknown, facing larger and more complex challenges, until the player character finally faces the biggest challenge of them all and, if victorious, the player character receives a great gift.

In MMORPGs, the game level story is often so abstracted that the player has no real interaction with it at all. Story progression happens during patches and expansions, through no work on the player's part. This is in part because the narrative is shared among millions of players, but also due to the technical limitations of MMORPGs being distributed across servers. Even if the players on one server could work together to progress the story, players on other servers may choose different options, requiring the development team to create separate content for each server. This is non-tractable, and other than *A Tale in the Desert,* which uses a single server and resets the game world every year, and *Eve Online* which is only on one server and relies mainly on player level story, no MMORPG has found a solution to this.

At the other end of the spectrum, table-top RPGs offer the most meaningful and interesting choices to the player in regard to the game story. With the creativity of a human game master, it is possible to integrate the players' choices into the overall narrative of the game in meaningful ways. Good GMs can make even seemingly small decisions re-appear in grand ways within the story, giving the player a feeling of importance within the story.

CRPGs have a more defined game level story than MMORPGs as they have a built-in beginning and end, and usually include a few points in which the player has some choices that will affect the story. In some cases, these choices may change the ending of the story, or future options within the story. For example, in *Dragon Age: Origins*, a female human character of noble birth (decided by the player at character creation time) who chooses to start a romance with the NPC party member Alistair, and takes steps to harden Alistair (which is done through dialogue), has access to a story ending which involves marrying

Alistair and becoming the queen. This is a great example of how player action can affect the game level story, but it is also a rare example. Creating multiple branches for player choices can often be error prone, and since only a subset of players will see it, many game companies feel that it is not worth the development time.

While creating a system that supports a few branching points within a generally linear narrative guarantees a strong and cohesive story, it often gives the player the impression that their choices rarely matter. It can also frustrate players that seemingly arbitrary choices may so strongly affect the future of the game (such as a choice of race changing a possible game ending) because these choices are not sign posted in any way. The lack of consistency ends up leaving a player unsure as to which choices matter and which do not.

3.1.1 Dynamic Story

To be able to adapt stories at the game level, the overall story needs to be able to incorporate some level of adaptability, and give feedback to the player that their choices are impacting the story.

To enable adaptability through dynamic restructuring of the storyline, it must first be broken into components which can be rearranged. We use the notion of plot points — self-contained units of story which signify important events along the storyline, that when experienced sequentially create a narrative – borrowed from previous drama management research (Sullivan, Chen, & Mateas, 2009). Different sequences of plot points define different player trajectories through the game level story space. By using plot points, the

system can choose a subset of the possible set of events, as well as re-order them to create many different possible narratives.

As an example, in *Mismanor* the player may encounter the plot points *Thomas is worried for his father* and *Thomas is protecting his father.* While this is a mildly interesting vignette, it becomes more interesting if the player encounters the plot point *The Colonel is disappointed in Thomas*. Now we have a more interesting dynamic in which the relationship is not automatically reciprocated. Finally, should the player uncover the plot point *Violet is blackmailing Thomas,* the player finds out that the Colonel is disappointed in Thomas because of incorrect information Violet has told the Colonel, but Thomas cannot correct her because Violet is threatening to harm the Colonel if he does. On the other hand, if the player learns that Violet is blackmailing Thomas first, the other plot points feel like extraneous information, as the player can easily extrapolate the knowledge given the blackmailing plot point.

Choosing plot points requires taking many considerations into account. From the author's standpoint, they would like to tell a story that follows a set of desired heuristics such as tension and revelation. The player, on the other hand, wants to be able to affect the story with their actions and choices within the game.

The Grail Framework uses these considerations to choose a potential ordering to the plot points, as well as which plot points the player will encounter. A small subset of plot points are active (or possible to come next) at any one time. This subset is chosen based on authorial considerations such as story cohesion and tension, as well as taking into account

the player's past actions by looking at the social state of the world. For instance, if the player has a strong friendship with one of the characters, they are more likely to have plot points concerning that character be active.

Which plot point is triggered from the subset of active plot points is in turn based on the player's actions. The plot points are revealed as dialogue exchanges between the player and a non-player character, which are mixed-in to the dialogue associated with social moves. Therefore, the plot point that is triggered is based partially on which social move the player chooses as well as who the player is interacting with. This gives the player interesting and meaningful choices at the game story level, while still allowing some authorial control to the overall structure of the story.

4 World Level Stories

The world level of story describes the setting of the game world – the backstory or past events that are asserted as facts, as well as the world structure. Games that exist in a series often share a world level story, using the common setting as a unifying feature between the games. Some games such as the *Mass Effect* series (BioWare, 2007) use game-level choices in the early games as world level story in subsequent games.

World level stories for many CRPGs are often large and complex but static, and therefore offered as optional content. This is done using in-game books that players can choose to read or extra dialogue possibilities when talking to NPCs. Some games offer the world level stories as real-world novels, available to the player if they desire, but not as a requirement for understanding the game level story.

Even though world level stories are asserted as facts in the game, it is possible and interesting to think of scenarios in which this level of story could also be playable, such as a *Memento*-style (Nolan, 2000) game in which the backstory itself is unreliable and scenes are experienced in reverse order. However, this was not the focus of this dissertation, therefore the majority of the work presented here instead focuses on methods for making the game, quest, and player level stories playable.

5 Level Interconnectivity

The overall narrative experience is shaped by all four levels of storytelling. When the levels work together, each influencing the other, the player feels connected to the story and that their actions play a part in shaping that story.

It is when the levels of storytelling do not work together that common problems within CRPGs are found. When the actions available to a player do not match the type of story the author wishes to tell, the designer must use cut-scenes to force the desired action onto the player character. For instance, if a player is only allowed combat-centric actions within the game, and the designer wishes to create a love scene, there is no way for the player to act out their role within that scene, and a cut-scene must be used. Sometimes designers will force the combat mechanics to fit the mold, but it ends up feeling very contrived, such as the quest in *Dragon Age 2* (BioWare, 2011b) to kindle the romance between the two NPCs Aveline and Donnic. In this quest, the player character is asked to help get guardsman Donnic's attention by increasing the amount of time the two characters can spend together in guard duty.

Similarly, when the game mechanics are not tied to the quest level stories in quests, the story must be explained with triggered scripted events, wordy quest descriptions or dialogue. While some of the scripted events may be interrupted (giving the player an interesting choice), if they cannot, they are merely game-rendered cut scenes. Removing control from the player gives them less reason to care about the story, and instead merely focus on the actions required to complete the quest.

Finally, when there is disconnect between player and quest level stories, quests are created which feel heavily scripted. These quests often come in the form of delivering an item or talking to an NPC on another NPCs behalf. Unless there are choices available in the item delivery or the NPC dialogue, the quests are considered boring and inconsequential.

6 Conclusion

Story is the key component of CRPGs, and part of what defines the genre. However, the term story describes four different levels of narrative within the game experience. The player level story is the story created by the player based on their moment-to-moment decisions constrained by the available game mechanics. Quest level story is the story arc made by the player completing a series of actions which fulfill a specified goal such as in a quest. Game level story is the overall narrative that shapes the game experience, and serves to give meaning to the player's actions. Finally, the world level story is the setting of the game, presented as facts within the world.

When all four levels work in tandem to create interesting and meaningful choices for the player, then playable stories will truly have been achieved.

CHAPTER 4

CIF-RPG AT THE PLAYER LEVEL

As introduced in chapter 3, the player level of storytelling in CRPGs is defined by a series of actions the player makes moment-to-moment, within the constraint of the game mechanics. Because the game mechanics define the possible space of stories, we look to increase the types of stories available to players beyond the limited game mechanics currently available in CRPGs. In particular, we are interested in increasing the number of options at the social level when interacting with other NPCs, increasing the complexity of the social actions available, and improving the interplay between social actions.

Because we are interested in focusing the game complexity on character relationships instead of combat mechanics, we chose to use the Comme il Faut (CiF) system (McCoy et al., 2010), previously used in *Prom Week* (McCoy, Treanor, Samuel, Mateas, & Wardrip-Fruin, 2011) (a social interaction simulation game set in the week leading up to a high school prom), to leverage the first-class models of multi-character social interactions. From this system, we created CiF-RPG (Sullivan et al., 2011) which supports an explicit model of a player character, and treats items and knowledge as first-class objects similar to characters.

CiF includes first-class models of social interaction – modeling relationships, traits, statuses, social history (in the social facts database, or SFDB), and culture, along with a library of social interactions or moves. The rules about how these models affect social interactions are represented as micro-theories. The micro-theories represent social knowledge outside of

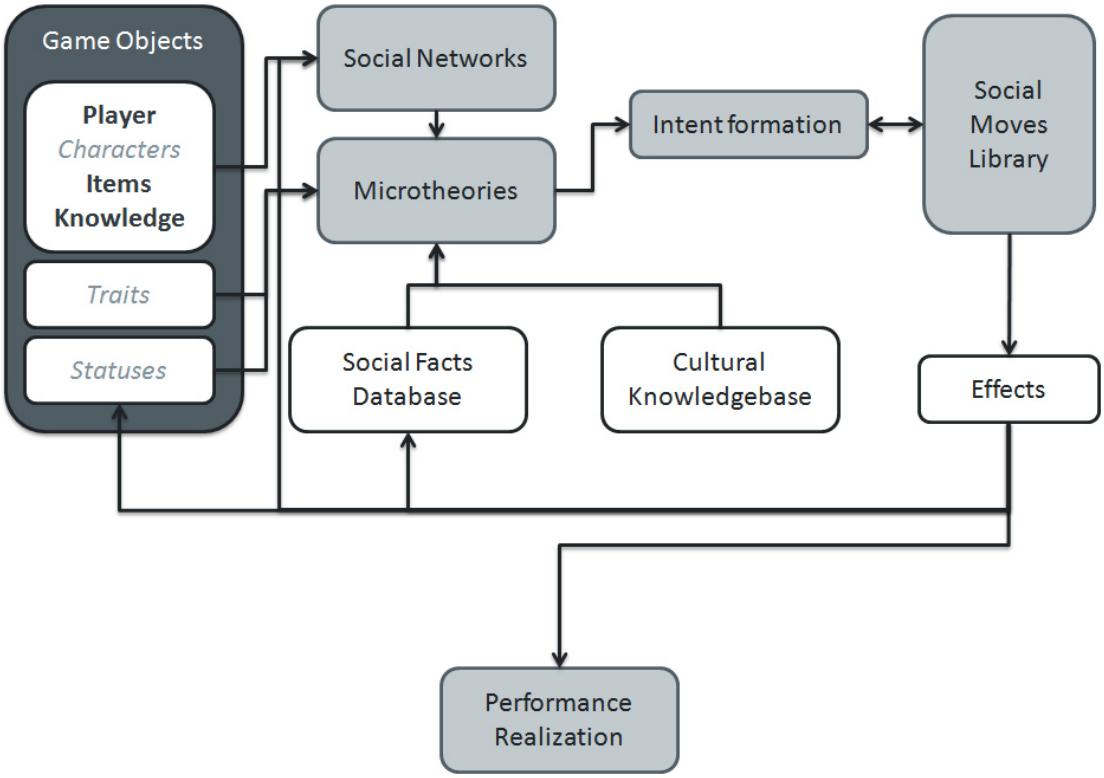

Figure 2: An overview of the CiF-RPG architecture, and how it differs from CiF. .Structures in dark grey and Game objects in bold were added to CiF-RPG. Structures in grey and game objects that are italicized were modified in CiF-RPG. Structures in white were not modified significantly, although content was created for them.

the context of specific social moves, supporting their reuse. The micro-theories are used to modify the saliency of each social action—adding a positive or negative weight to whether a character is likely to want to engage in a specific social action.

1 Social State Representation

For our use, one of the most important aspects of CiF is the social state representation. Each game object is described by a set of traits and statuses, while several relationship networks track how certain types of game objects feel towards one another. The Social Facts

Game Object	
Name	Violet
ID	0
Type	TYPE_CHARACTER
Traits	*female, charismatic, independent, observant, vengeful, dishonest, snide, domineering*
Statuses	*manipulating (player), is dating (James)*
Examine description	"You often catch Violet looking past you, as if something else is on her mind."
Discuss lines	Neutral: "Violet takes after her mother." Like: "Violet has an independent streak which lets her get quite a lot done." Love: "Violet notices every little thing. She's quite observant." Hate: "You need to watch out for Violet. She's a wily one."

Figure 3: An example of a game object -- the character Violet.

Database (SFDB) is used to store notable events that have occurred due to player action and the game's backstory, and the Cultural Knowledgebase (CKB) describes character opinion of cultural objects or concepts, and what the popular opinion is of those objects. Figure 2 shows the CiF-RPG architecture, denoting additions and modifications to CiF that were required for the creation of CiF-RPG. Each of the systems is described in more detail below.

1.1 Game Objects

The base version of CiF only deals with characters, so it was necessary to abstract game objects as we wanted to support items and knowledge as well as the characters. Game

objects include the name of the object, network ID, type of object (character, item, or knowledge), traits, statuses, description upon examination, and lines of dialogue when describing that object. An example can be seen in Figure 3. In most cases, CiF-RPG works with the top level game objects, without needing to consider what type of object it is.

The name of the object is the name that will appear when referencing it in dialogue, as well as the name that is shown when clicking on the object. The one exception to this is that CiF-RPG always uses the name “Player” for its internal representation, but it is substituted for the player’s chosen name when displayed during the game. This was done because the player character is set up partially before the user specifies customization details such as a name.

The ID given to the object refers to the networkID. This ID is used to reference the network values for all the social networks between each of the characters. Each game object is assigned a unique ID.

Type refers to whether the game object is a character, an item, or knowledge. The type is automatically assigned when a game object is initialized, as a base game object is never allowed to be created. Rather it must be a character, item, or knowledge.

Traits and statuses are explained in more detail in sections 1.2 and 1.3 respectively. They describe the object’s make-up and current state. The examine description and discuss lines are both used for dialogue mix-ins during social moves. When executing the examine social move, the examine line is used, so that every object can have a unique description. The discuss lines are similar; except that they are used during discuss social moves. Each discuss

Character	
gender	“female”
inventory	“Violet’s Diary”, “white wine”
itemFromList	“Violet”, “Baroness”
memory	“Baroness has a drinking problem”, “James is afraid of getting kicked by a horse”
knowledgeFromList	“Baroness”, “James”
Prospective memory	Caching for faster intent formation
locutions	Sweetie: “Love” Pejorative: “idiot” Positiveadj: “lovely” Shocked: “Oh my.” Greeting: “Why hello”

Figure 4: The internal structure of the character Violet.

line is associated with a particular feeling. These feelings are used in a general way to denote the emotional valence of the object doing the describing towards the object they are describing. The emotional valence is calculated by averaging the romance and friendship networks and comparing them to threshold values.

1.1.1 CHARACTERS

Characters are game objects that a player can interact with, and can in return interact with the player. The character object is used for both non-player characters as well as the player, as CiF-RPG does not differentiate between the player and NPCs when reasoning about the social state and available social moves. Since characters extend game object, they inherit all the information shown in Figure 4.

However, characters also have unique features that CiF and CiF-RPG track. Unlike other game objects, characters have a gender associated with them. This is used during dialogue, in which the dialogue can use the gender to choose pronouns or other gender-specific turns of phrase. For instance, the following line of dialogue:

“%gender(i, He, She) slowly pats out the wrinkles in %gender(i, his vest, her skirt)% while trying to calm down.”

This line looks at the gender of the character initiating (or starting) the social move, denoted by the ‘i' as the first variable, and then chooses the first set of text if the initiator gender is male, or the second set of text if the initiator is female. This gives dialogue writers the ability to use gender-specific dialogue which can be used with any character.

The inventory tracks what items the character has in their possession. Whenever an item is added to the inventory, the item and character automatically receive statuses which denote that the item is being held by that character and that the character is holding the item. The itemFromList is paired with the inventory, and tracks who the character received the item

from. If a character started with that item, and did not receive it from anyone else, their name is stored in the itemFromList. In retrospect, this could have likely been better accomplished using the Social Facts Database as there is more robust support for SFDB within CiF and CiF-RPG, and in the future we will move to using that instead.

Memory and knowledgeFromList are very similar to inventory, but instead of tracking items, it tracks the knowledge the character possesses and who they learned the knowledge from. When gaining a piece of knowledge, the character and knowledge automatically receive the statuses which note that the knowledge is known by that particular character and that the character knows that knowledge.

Using statuses for knowledge and items was done to easily leverage the support already provided in CiF. It is easy to test whether a character is holding an item or knows something as a precondition or influence rule for a social move, quest or plot point, and by adding or removing the status items can be moved in or out of an inventory, or into memory. For our purposes, knowledge cannot be lost, but it is easily added to the system if that was desired.

1.1.2 ITEMS

Items and knowledge build on the representation of characters – in a sense CiF-RPG treats them as if they were a type of character with limitations on how they can be interacted with. While this may initially seem counter-intuitive, representing items and knowledge as character-like entities allowed them to inherit all the reasoning mechanism of micro-theories, influence rules and so forth for determining how they participate in social actions.

Item	
Name	White Wine
ID	10
Type	TYPE_ITEM
Traits	*item, examinable, holdable, usable, alcoholic*
Statuses	*drinkable*
Examine description	"The white wine has a delicate aroma."
Discuss lines	Neutral: "The white wine is a very good vintage."

Figure 5: The internal structure of the item White wine.

Items are game objects that can be interacted with similar to game characters, with a set of specialized traits and statuses that can only apply to items. Items can appear on screen like characters, but can also be held (handled as a directed status). Items cannot autonomously choose reactions in our story, although we left the system flexible to having magical items that are able to react to the player, such as a cursed dagger which talks to the player.

Items extend the base game object, but do not add any unique features, as seen in Figure 5.

While items have a network ID, for our uses, we did not track social networks with items, as it did not feel appropriate for our story. However, CiF-RPG supports it even though it was not used. This could be used to track emotional attachment to items, such as a favorite doll or a despised dagger.

Items can be described by traits and statuses that are unique for items. For instance, the trait *holdable* means that the item can be added to a character's inventory. Without that trait, that item cannot be picked up. The *alcoholic* trait is used to determine the effect of using that item, in this particular case it would give the imbiber the status *tipsy* or *drunk* depending on their current state of inebriation.

The *usable* trait is used in conjunction with the statuses. The statuses describe in what ways that item can be used. For instance, a door could be *openable,* which means that once used, it would be open and have the status *closeable. Drinkable* denotes that if used, the character will drink the item, and afterwards it will be empty and will lose the *drinkable* status.

Because we do not track social network values with items, there is only one discuss line available. If networks were tracked, it would make sense to supply more than just a neutral discuss line so that dialogue about that item could reflect a character's feelings about that object.

1.1.3 KNOWLEDGE

Knowledge is the most restricted game object, in that it can only be referenced in social moves by the player and other characters, and cannot be interacted with directly. Knowledge, like items, has a set of specific traits and statuses which are used to describe the circumstances of a particular piece of knowledge. A full knowledge object including inherited features is shown in Figure 6.

Knowledge	
Name	Violet fears being abandoned
ID	unused
Type	TYPE_KNOWLEDGE
Traits	*knowledge, emotional, embarrassing, secret*
Statuses	*true*
Examine description	unused
Discuss lines	Neutral: "I've heard that some of Violet's issues come from her fear of abandonment."
Mentioned Locations	*none*
Mentioned Characters	"Violet"

Figure 6: An example of the internal structure of the knowledge "Violet fears being abandoned." Grey features are inherited, green (and dark bordered) features are those that are unique to knowledge.

CiF and CiF-RPG store events as facts into the Social Facts Database (SFDB) which will be described in Section 1.5. However, a restriction of the original CiF is that events that are in the SFDB are known by every character. In CiF-RPG, we wanted a way to capture the notion of knowledge that is not global to all characters, but can be learned by various methods.

Similar to items, we do not track social network values between characters and knowledge. However, this could be used to track a character's feelings towards a piece of knowledge, or an interest in hearing more about it.

There are specific traits and statuses that are reserved for use with knowledge. For instance, an *emotional* vs. *factual* piece of knowledge refers to the knowledge being about an emotional state. Similarly, a *secret* trait refers to something that at least one character would like to keep quiet. This is useful when deciding which knowledge a particular character might wish to discuss. For instance, a *shy* character with a low *trust* towards the player would be less willing to discuss a secret than a *gabby* character with high *buddy*.

Currently, the only two statuses for knowledge are *true* and *false.* These are stored in case a piece of knowledge becomes false over time (for instance, if Violet no longer fears being abandoned), or to possibly be used to mark information for use by a *dishonest* character.

Because knowledge cannot be interacted with directly, there is no use for the examine description. Also, similar to items, because we do not track social network values between characters and knowledge, there is only one discuss line available.

Finally, the two pieces of information that are unique to knowledge are tags for any locations or characters that are mentioned. Currently, this is only being used by plot points (a specialized type of knowledge described in more detail in Chapter 5, Section 2.1), but could be generalized to be used for social moves. For instance, a *vain* character might want to talk more about themselves and would prefer knowledge that was character tagged with their own name.

1.2 Traits

One of the ways that game objects are described within CiF-RPG is through traits. Traits are descriptors of static characteristics of an object. For instance, a character might be

charismatic, sentimental, unforgiving, or *defensive*, which all describe characteristics which do not change over time. Traits are different from the typical *stats* used in RPGs, such as *strength* or *wisdom,* in that they are binary; either a character has them or they do not, and they do not change in strength over time. Our prototype game, *Mismanor*, is a short enough experience that progression was not needed, but a system such as this would handle progression by allowing players to add traits and statuses. More on this can be found in the character creation section in Chapter 8.

Traits are broken into two categories, positive and negative, based on the cultural implications of each trait. This is used both for character creation as well as for our player models (more information can be found in Chapter 7, Section 6) used for testing. Traits are often domain specific, with available traits being decided based on the type of story the author would like to enable.

Traits play a large part in influencing what social moves are available for a character to initiate, as well as how a character will respond. For example, a *sentimental* character is more likely to *Reminisce,* while a *disapproving* character is more likely to *Find Faults.* For items and knowledge, traits are used to determine which particular social move they may take part in. Social moves are described in more detail in Section 2.

1.3 Statuses

Unlike traits, statuses are transitory descriptors, used to represent temporary character states. Statuses can describe general states such as *sad* or *tipsy*, or they can be directed at a specific character such as *angry at* or *in love with*. Statuses can be designated to wear off

after a certain amount of time, or be allowed to continue indefinitely until some action within the game removes them.

Character-based statuses are also categorized by whether they are positive or negative statuses, based on what type of emotional state they describe. For instance, *sad* is considered negative, while *excited* is considered positive. There are a few statuses such as *tipsy* or *drunk* which are not clearly good or bad so they are not categorized. Furthermore, directed statuses were also categorized, such that *grateful towards* is a positive directed status, while *is enemies with* is a negative directed stat.

These categories are very useful when creating influence rules (see Section 2.3) and micro-theories (Section 3) in that it can easily be denoted, for instance, that a character in a generally positive frame of mind is more likely to *Flirt* than a character in a generally negative emotional space. Additionally, being able to test whether a character has positive (*feels good about*) or negative (*feels bad about*) directed statuses involving them is equally helpful. For example, if a character *feels good about* the character initiating a *Flirt* move, they are more likely to respond to the flirt positively.

1.4 Social networks

Social networks are the way in which CiF-RPG tracks the current status of the interpersonal feelings between characters. While statuses represent a character's general emotional state, social networks represent how two characters relate to each other. For instance, while a character may be *angry at* another character, they might still have a strong romantic attachment, or strong feelings of friendship. CiF-RPG maintains multiple dynamic social

spectrums between characters, which are bi-directional, but not necessarily reciprocal. For instance, in the game *Mismanor* (described in detail in Chapter 8), the Colonel has a high family bond towards his daughter, Violet, but Violet does not have similar feelings towards the Colonel. Therefore, these values can move independently of each other, allowing for more complex relationships between characters.

Since inter-personal feelings are multi-dimensional, CiF-RPG supports a number of different social networks. These networks are stored as a number on a scale of 0-100, with social actions often changing the value along this scale. As an example, *Mismanor* uses four social networks: *trust, buddy, romance,* and *family bond.* These were chosen based on the setting and tone of the game.

Because social networks are the main way inter-personal feelings are tracked, they play a large part in CiF-RPG. Most social moves are based on the intention of increasing or decreasing a character's feelings along a particular social network. For instance, *Flirt* has the intent to increase romance, while *Petty Argument* has the intention of decreasing feelings of friendship. Current network values also play a part in weighting what social actions are available to a character as well as how that character will respond to different actions.

Additionally, how characters feel about the player makes a difference in what plot points, quests or even game endings they may experience. This is covered in the chapter on GrailGM, Chapter 5.

1.5 Social Facts Database

The Social Facts Database, or SFDB, tracks every social interaction over the course of the game. This allows the system to reference past events to influence future events. Events can also be labeled for more general reference. For instance, if the player backstabs the Colonel, this would be labeled as a negative action. In future interactions with the Colonel, the SFDB can be searched for past negative actions which would have an influence on which social actions the Colonel would be willing to engage in with the player.

Each entry into the SFDB can also have a line of dialogue associated with it such that it can be brought up in future dialogue exchanges within social moves. Using the previous example of backstabbing the Colonel, if the player attempts to flirt with the Colonel in a future social action, the Colonel may reject the attempt and refer back to the backstab action using the associated dialogue string.

The SFDB is also used to provide back story for the game. By seeding the SFDB with events before the game starts, the characters within the game can make references to these actions in the same way as they reference actions made by the character. This provides a unique way to not only have back story be brought up through playing the experience, but to also have the back story have an effect on the way the game progresses.

1.6 Cultural Knowledgebase

The Cultural Knowledgebase or CKB is where facts about the popular culture associated with the game world are stored. For instance, in *Mismanor*, roses are considered *romantic* by the

popular opinion, which can be stored within the CKB. Information making up the cultural backdrop of the game is provided by the authors at design time.

In addition to the popular opinion, the CKB stores each character's actual opinions of the object; each character may either like, dislike, or have no opinion on the object. For instance, Violet likes roses so when engaging in a romantic game, roses could be specifically referenced within the dialogue.

This provides a way for interesting variation within the dialogue as well as allowing ways for character's quirks to be revealed through the course of the game.

2 Social Actions

Social actions, or social moves, are the mechanics that are available to the player and NPCs to interact with one another. In combat mechanics, a player is given choices of which combat actions they would like to use (e.g. cast a fireball, cast a heal spell, flank an opponent, backstab from stealth). The actions are based on the type of character they are playing and the set up of the combat. For instance, stealth may not be available if there are no shadows, or fireball will not work on fire-based creatures. Social actions are similar, but within the domain of character interaction. The makeup of a social move is shown in Figure 7 and Figure 8.

Social Move	
Name	Compliment
Role Types	Initiator: Character Responder: Character
Intent	Buddy up (responder towards initiator)
Preconditions	none
Initiator Influence Rules	+1: initiator is outgoing (trait) +1: initiator is grateful towards responder (status) -2: initiator doesn't like responder much (buddy < 30)
Resp. Influence Rules	-1: responder is humble (trait) -1: responder is embarrassed (status) +1: responder trusts the initiator (trust > 50)
Instantiations	"You look %gender(r, handsome, lovely)% tonight." Accept: "Thank you. You are too kind." Reject: "I fear you may need your eye sight checked."
Effects	Accept: Responder likes the initiator more (buddy +10) Reject: Initiator is embarrassed (status: embarrassed) & Responder trusts the initiator less (trust - 5)

Figure 7: An example of a 2-person social move, Compliment. The influence rules, instantiations, and effects are samples from a larger number of entries.

The social moves created for a game are tailored for that specific experience. Just as a World War II CRPG would likely find magic-based combat actions inappropriate, the type of social interactions are heavily influenced by the design of the game. A murder mystery game within a cloister would not need romance moves, whereas a Jane Austen adaptation would require them.

Social Move	
Name	Give Romantic Gift
Role Types	Initiator: Character Responder: Character Other: Item
Intent	Romance up (responder towards initiator)
Preconditions	Initiator is holding an item (status)
Initiator Influence Rules	-1: initiator is shy (trait) +2: initiator loves responder (status) -1: initiator is angry at responder (status)
Resp. Influence Rules	-2: responder is unforgiving (trait) & initiator did something mean recently (SFDB) +2: responder likes item (ckb)
Instantiations	"I want you to have %o%." Accept: "Oh this is %positiveAdj%, thank you!" Reject: "I'm afraid I cannot accept this from you."
Effects	Accept: Responder loves the initiator more (romance +10) Reject: Responder pities initiator (status: pities) & Initiator loves responder less(romance - 5)

Figure 8: An example of the social move, Give Romantic Gift, which requires three game objects: two characters and an item. The influence rules, instantiations, and effects are samples from a larger number of entries.

Social actions can have two or three participants, referred to as the initiator (who started the social action), the responder (who they are interacting with), and if there are three participants, the other (not directly participating in the social action, but referenced).

When a social move is initiated, the responder can either accept or reject the social move. In combat this would be akin to hitting or missing. Choosing a move and how a move is accepted or rejected are described in section 3.

As mentioned above, currently, only characters can initiate social moves, although the system could easily support item or knowledge initiators if it was desired. A responder can be either a character or an item, and the other can be a character, item, or knowledge.

A role-type must be assigned during the creation of the social move. This means that there cannot be a generic *Discuss* social action, but instead there is a *Discuss Item, Discuss Knowledge,* and *Discuss Character.* This was done because each of these social actions has different influence rules and effects and should be considered uniquely.

Because physical items and knowledge are treated as fundamentally the same as other game objects, social actions can be created that use items or knowledge. These actions rely on the traits and statuses of objects to describe the possible interactions the player may take. As discussed above, an object with the status *drinkable* allows the player to drink the object. If the object also has the trait *alcoholic*, upon drinking, the player will gain the status *tipsy*. Similarly, knowledge with the trait *secret* is less likely to be discussed unless the trust network between the characters is high. This gives the designers the ability to quickly make objects and describe them with traits and statuses without having to create actions to interact with each object. While the micro-theory rules for physical object and knowledge interactions will generally be simpler than that for social interactions, the full power of CiF-RPG is available to support making physical object and knowledge interactions as context and history dependent as desired.

2.1 Intent

The *intent* of a social move is the outcome the initiator desires if the social move is accepted. In general, social moves can have intent to increase or decrease social network values or add or remove statuses. For instance, the *Ask Out* social move has the intention of adding the *Is Dating* status, while the *Mediate* social action has the intention of increasing the responder's family bond network value with the other.

In combat mechanics, a fireball would have the intent to cause damage, a heal would have the intent to remove damage, a buff increases stats, and a debuff decreases stats. The intent is a way to generalize the type of action that particular mechanic is associated with.

2.2 Preconditions

Preconditions are requirements for a social move to be available. In general, preconditions are used rarely, as they may cut off interesting dramatic possibilities. For instance, if the *Ask Out* social move had a prerequisite that the initiator and responder were not currently dating someone, this would make it impossible for characters to cheat. While this may be desirable in some domains, it removes some possibly interesting player stories in which one of the characters may get caught cheating.

Instead, we choose to use preconditions to exclude situations in which the social move would not make sense at all. For instance, the *Break Up* social action does require the initiator and responder to be dating, as breaking up makes no sense if the characters are not dating to begin with.

2.3 Influence Rules

Every social action has a set of initiator influence rules and responder influence rules. These rule sets correlate to rules that change the likelihood of a particular character initiating that particular social move, and the likelihood of the responder accepting that particular social action, respectively.

For the initiator influence rule set, each rule contains one or more predicates that describe a possible characteristic of a character, along with a weight for that rule. For instance, a character with the trait of *shy* might have a -2 weight towards initiating the *Boast* social move, and a character with a romance network value greater than 50 towards the responder will have a +1 towards the *Flirt* social move.

The responder influence rule set is similar to the initiator influence rule set, except that instead the weights are used to calculate whether the responder accepts or rejects that particular move from that character. If the final score is 0 or more, the move is accepted; less than zero and the move is rejected.

Because there are a number of similarities between influence rules for different social moves, CiF and CiF-RPG both support *micro-theories*. These are explained in more detail in Section 3, however, they are used to generalize influence rules so that they need not be entered for each social action.

This leaves the influence rule sets for describing specifics and exceptions that are not captured by the micro-theories. For instance, a *shy* character may generally have a leaning towards buddy up social moves (captured within a micro-theory), but that same character

would be very unlikely to initiate a *Boast* social move (captured within the initiator influence rule set).

2.4 Instantiations

The instantiations for a social move are the possible dialogue exchanges between the initiator and responder. The exchanges can be spoken by any character, and therefore there is a set of tags to denote places in which CiF-RPG or GrailGM should replace with different content.

The use of %gender% is discussed earlier in the chapter, in which the author may specify parts of the dialogue based on the gender of the initiator, responder, or other. Each character also has a list of locutions to personalize the dialogue to that character. For instance, as shown in Figure 4, Violet uses the term “lovely” as a positive adjective. Another character might prefer “amazing” or “fantastic”; these locutions allow for some of the character’s personality to come through in the dialogue. In Figure 8, the accept instantiation for accepting the romantic gift includes the responder saying “Oh this is %positiveAdj%, thank you!” In Violet’s case, if she were to speak this line it would be shown as “Oh this is lovely, thank you!”

To use a character’s name in the dialogue, %i%, %r%, and %o% refer to the initiator, responder, and other, respectively. These will be replaced with the name of the character, item or knowledge taking that role.

Other mix-ins include functionality to randomly choose from a selection of dialogue, inserting the examine description or discuss lines for a game object, and referring to

information within the CKB or SFDB. All of these are used keep the dialogue fresh and contain character-specific nuances.

2.5 Effects

Finally, the effects of a social move describe different outcomes for that social move. Each effect is either an accept or reject effect, and is paired with an instantiation.

Each effect has a description of a social state that is the *condition* of the effect, and the *change* to the social state if that state is met. For instance, for the *Court* social move, one accept effect has the *condition* that the two characters are not currently dating, and their romance network is higher than 65. The *change* for that effect is that the characters are dating, and the romance network is increased by 10. The effect also has a specific instantiation which is used to convey that the characters are now dating. Every social action must have a default accept or reject – an effect in which there is no social state to match, just the outcome of that move.

When choosing an effect, CiF and CiF-RPG choose the effect with the most specific *condition* rules, unless that effect has been used too recently. This is to keep the same instantiations from being shown in succession.

3 Micro-theories

As mentioned above, micro-theories are used to encode general preferences for initiating, accepting, or rejecting social moves based on traits, statuses, or social network value. A micro-theory consists of a definition (what the micro-theory is defining) and a set of influence rules which are weights associated with the different types of social move intents.

Micro-theories allow the designers to create general rules of thumb for each trait and status without having to add an influence rule to each specific social move.

For instance, the micro-theory describing the status *cheerful* has rules that apply to any character that currently has the *cheerful* status. The influence rules describe that a cheerful person is more likely to initiate and accept social moves that increase friendship, trust, romance, or family bond, and less likely to initiate or accept social moves that decrease these networks. Additionally, social moves that remove negative statuses or add positive statuses are positively weighted, while the opposites have a negative weighting.

Micro-theory influence rules are used for calculating whether a character initiates a particular social move, as well as whether a character accepts or rejects a particular move. While this is not always desired, for instance in the case of a *domineering* character being likely to initiate a social move to add the status *is dating* but less likely to accept such a move, it was generalized to save time and effort on the part of the authors.

3.1 Choosing a social move

To decide on a set of social moves to present the player, or which social action an NPC will initiate, CiF-RPG calculates the total weight for each social move based on micro-theories and initiator influence rule sets. The initiator's traits, statuses, network values, entries into the SFDB, and preferences in the CKB all come into play.

Using the cheerful micro-theory described above, and the *Compliment* social move shown in Figure 7, if the initiator has the trait *outgoing*, the status *cheerful* and a buddy network higher than 30 with the responder, they have a positive weight to initiate the *Compliment*

move. In reality, hundreds of rules are taken into consideration to calculate a final weight towards a particular social move.

When presenting moves to the player, the top 4 social moves are presented for the player to choose from. However, if a move has been used too recently – within the last three actions – it will not be available. This was initially done to keep the NPCs from continually choosing the same move, but also helps to simulate a "cool down" and keep the player from just using one social move.

When an NPC is interacting with the player, the highest weighted social move is chosen. Again, if a move has been used within the last three actions, it is not available to be used.

3.2 Accepting or Rejecting a move

Once a move has been initiated, the responding character either accepts or rejects the move. This is similar to a hit versus a miss in combat, and simulates real-world interactions in which a social maneuver is not always taken in the way it was intended. For instance, there are many jokes about how a compliment can be taken the wrong way, in which the person receiving the compliment ends up hurt instead of feeling better.

To calculate whether a move is accepted or rejected, the micro-theories are used but from the responder's point of view, along with the responder influence rules. For instance, the *cheerful* micro-theory describes that an initiator is more likely to initiate social moves with intents to increase social networks. This also means that a *cheerful* responder is more likely to accept social moves with intents to increase social networks.

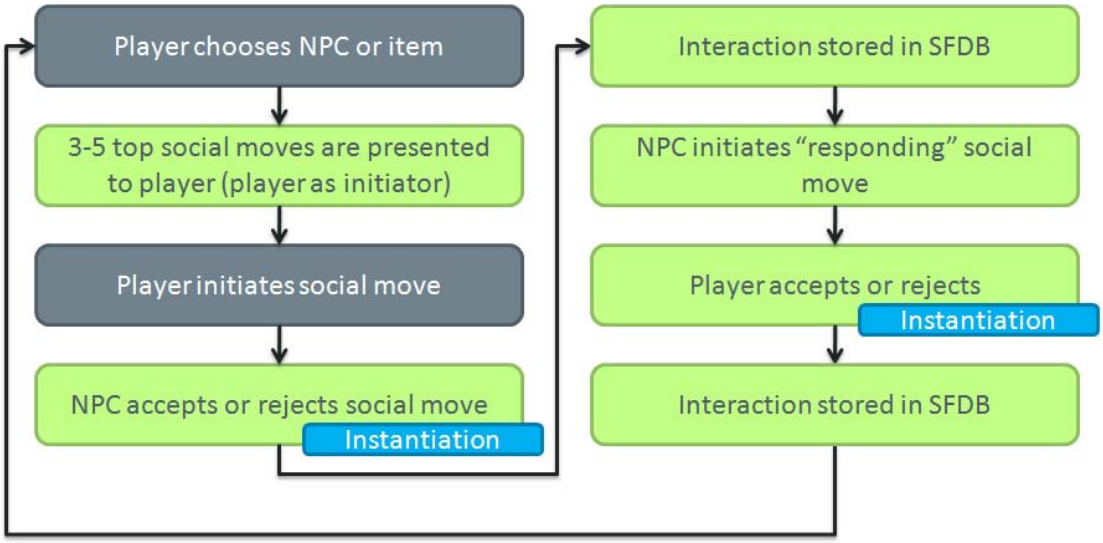

Figure 9: The game loop in CiF-RPG. Dark grey boxes denote player input, light green boxes denote CiF-RPG calculations. Small blue boxes are places in which the system outputs dialogue exchanges.

The weights of the influence rules and micro-theories are summed, and if the weight is zero or higher, the move is accepted. A lower weight means the move is rejected. We considered allowing the player to choose whether they accepted or rejected a social move, but decided that for the traits and statuses to be more meaningful, it would be calculated and decided based on their character's social state.

4 Player-System Interaction

The game loop in CiF-RPG (shown in Figure 9) involves a turn-taking approach to social interactions. Social actions describe a discrete interaction between two characters, although they can involve a third character, item, or knowledge. One character proposes the action, and the other character accepts or rejects it based on that character's traits, statuses, and the social network values between the two characters. In CiF-RPG, one of these characters is always the player.

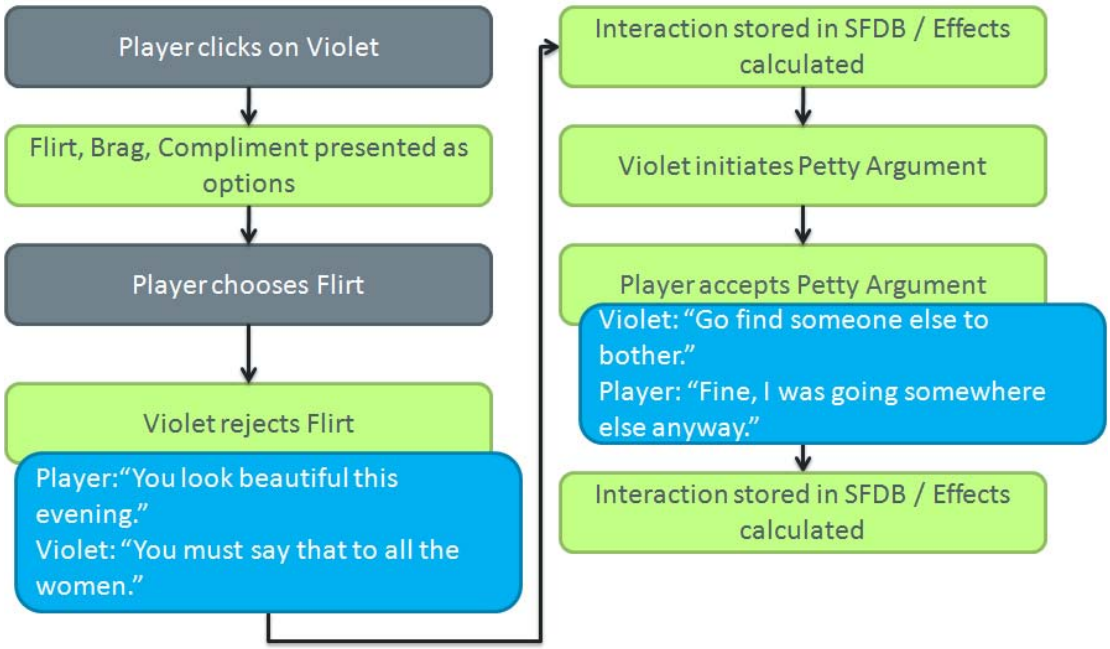

Figure 10: An example in the game loop in which the player attempts to flirt with Violet, is rejected, then accepts Violet's petty argument social move.

For instance, the player may choose to try a flirt action between their character and an NPC. Hundreds of rules are evaluated to test whether the NPC accepts or rejects the move. For instance, if the romance between the characters is high, the NPC is not currently dating someone else, the player character is confident, and the NPC is not angry at the player, the NPC likely has a positive weight towards accepting the flirt action, which has the effect of raising the romance levels between them, as well as possibly removing the *heartbroken* status if the NPC has it.

However, if the NPC is shy, is currently in love with another character, and is resentful towards the player character, they will likely reject the player's advances (seen in Figure 10),

leaving the player with the *embarrassed* status for 3 moves and causing the player's romance towards that character to drop.

The original design of CiF was to be used within a social simulation, where a player could choose any two characters to interact. However, within a CRPG, a player is generally controlling only their own avatar, and interacts with game characters through that avatar. While this is not quite the case in party-based CRPGs such as *NeverWinter Nights* (BioWare, 2002b) and *Baldur's Gate 2* (BioWare, 2000), in social exchanges, it is often assumed that the player's character is the one that is directing the social interchange.

In order to properly model the social actions, the CiF-RPG system needs an internal representation of the player. We chose to model the player similarly to the model of the non-player characters within CiF, such that CiF-RPG itself does not know the difference. This gives us the ability to leverage the capabilities of CiF and keep consistency across the characters. This also allows us to tailor the set of possible social actions presented to a player based on the way they chose to create their character as well as the current standing between the player character and the character they are interacting with.

5 CiF-RPG and Player Level Stories

With CiF-RPG, the player now has access to a range of social mechanics. This changes the types of player level stories that a player can create. For instance, instead of describing the strategy employed for fighting and beating a boss creature, in *Mismanor,* a game in which the player is destined to be the sacrifice of a cult ritual, the player can create and relate a story in which they are able to convince Violet that she truly loves James, that her feelings

are more important than her father's approval, and she gives up her cult aspirations and elopes with James. Or perhaps the player chooses instead to lure Violet away from James, thereby convincing her to save the player and sacrifice her brother instead.

By supporting social mechanics for the player level story, we open up a new range of possibilities within the quest and game level story space. Quests can be created which rely on social mechanics to reach a completion state which gives rise to a new range of quest types. Similarly, because social actions are available to the player, game stories can more strongly incorporate social dynamics without relying on cut scenes, but rather relying on interesting and meaningful choices for the player in the story space.

CHAPTER 5

GRAILGM AT THE QUEST AND GAME LEVEL

To support our desire for playable stories at the quest and game level, we created the GrailGM system, a run-time game master which uses the current social state and the player character's traits, statuses and network standings to dynamically select quests and story plot points.

1 QUESTS

GrailGM supports quests with multiple goal states (see Chapter 5, section 1.5 for more details), allowing the player some choice in which direction they wish to take the quest. By choosing quests based on player action, and giving the player a choice in completion state, the player is able to shape the story as they go, having a discernible impact on the narrative.

To support this kind of dynamism, it was necessary to create a flexible quest structure. In modern CRPGs, it is common to use flags to track quest completion and story state. While flags are a simple mechanism, they require designers to carefully think through all the possibilities of the situation and are subject to state explosion. As the number of options grows, it becomes more difficult to track different game states, and there is more room for designer error. For instance, as discussed earlier (see 0 by Wardrip-Fruin (Wardrip-Fruin, 2009), even in the award-winning RPG *Star Wars: Knights of the Old Republic* (BioWare & LucasArts, 2003), it is possible to complete quests and meet characters in orders not

Figure 11: A quest flag bug seen in the MMORPG La Tale, in which the game does not recognize that the player has the items required to complete the quest.

foreseen by the designer, which can lead to inappropriate story moments or quests becoming impossible to complete.

Quest flags are also brittle. They are designer-created labels of state, but do not include any built-in knowledge of that particular state. Therefore, when something unexpected happens, such as the player completing quests or meeting characters out of the expected order, there is a breakdown of the system. A somewhat common example of this is seen in Figure 11 in which a player received a quest in the game *La Tale* (Actoz Soft, 2006) to retrieve 3 silver furs and 3 rough furs (Xagiri, 2008). The items are already in the player's

inventory, but the system does not recognize the quest as being completed, because the player did not receive the items while on the quest. The player is unable to complete the quest, as they cannot get the items and they are unable to progress.

Due to the brittle nature of quest flags and our desire for playable quests, we chose to instead use predicate logic to represent intent, pre-requisites, goal states, and completion states within our quest structure as seen in Figure 12. This allows the designer to describe the desired states without needing to track the various ways to get to that state.

1.1 Name, Quest Giver, and Quest Finisher

A quest is internally represented by the *name* of the quest, the *quest giver* and *quest finisher*, the *intent* of the quest, *preconditions, goal states, completion states, locations mentioned,* and *characters mentioned* as shown in Figure 12.

The *name* of the quest is used mainly internally, although it is also used when displaying active quests to the player in the quest journal.

The *quest giver* and *quest finisher* are the characters that give the quest and accept the quest when it is finished. Often this is the same character, therefore if a quest finisher is not specified, GrailGM designates the quest giver as the quest finisher. However, allowing these to be different characters gives the designer the option of creating delivery style quests, in which the quest giver asks the player to travel to the other character to deliver an item, or information.

Quest	
Name	Convince Violet and James to break up
Quest Giver	Colonel
Quest Finisher	Colonel
Intent	Violet and James lose the status *is dating*
Preconditions	Violet and James are dating Violet and James are not planning to elope
Goal States	Violet and James are not dating Violet and James are not dating, Violet dating player Violet and James are not dating, James dating player
Completion States	Violet and James are not dating Violet and James are not dating, Violet dating player Violet and James are not dating, James dating player Violet and James are planning to elope
Characters mentioned	Violet, James
Locations mentioned	none

Figure 12: A sample quest in GrailGM, in which the Colonel would like the player to convince his daughter, Violet, to break up with the stable boy, James.

The original intention was to have the quest giver and quest finisher use predicate logic so that the quests could be templated. For instance, the quest giver for the quest shown in Figure 12 could be any character that had a higher social network value with one character than the other in a relationship. However, in practice, we found that the authors preferred to design quests for specific people, as it was easier to design this way.

1.2 Intent

The *intent* of a quest is used as both a designation of the quest type, as well as capturing the intent or motivation the NPC would have for giving the quest – similar to the categorization of social actions. One of the ways in which we reason about quests is based on the quest type. For instance, *delivery* quests require movement and item manipulation actions, and *kill* quests require combat actions. Because CiF-RPG supports new types of player actions (particularly social actions) not typically found in RPGs, it was necessary to create new types of quests.

The intent of the quest correlates with the change in the world state if the quest is completed according to the default completion state. The new quest types are: *Relationship Up/Relationship Down* (e.g., improve Friendship or decrease Romance), and *Status Gain/Status Lose* (e.g., gain Dating, lose AngryAt). The system reasons about item and knowledge possession through using statuses, so the status quest types include quests that involve items and knowledge as well.

Using our predicate system, we can label each quest with a set of intents, which represent the motivation for an NPC to give that particular quest. For instance, the quest to break up James and Violet has the intent to remove the *dating* status between Violet and James. Each quest is associated with one or more intents. This allows GrailGM to reason about the quests based on NPC motivation, which works well with the capabilities of the CiF-RPG framework.

1.3 Preconditions

The preconditions of a quest describe the starting state of the world that must be matched for the quest to be given to the player. Generally, the preconditions are kept fairly simple, and are merely used to make sure that the player has not already met one of the completion states to give GrailGM the most flexibility. For instance, in our example quest in Figure 12, the preconditions check that Violet and James are still dating, as not only would it make no sense if the quest were received, one of the first three goal states would be immediately reached.

While it would still make sense for the player to receive the quest from the Colonel if Violet and James are planning to elope (perhaps even more sense), the preconditions check that Violet and James are not planning to elope, so that the fourth completion state is not immediately reached. In future iterations, we hope to be able to intelligently invalidate completion states which have already had their states met, so that quests may still be used.

We generally do not include preconditions that include the quest giver's feelings about giving the quest, as this is captured through intent formation and volition calculation when deciding on possible quests. We discuss this in more detail in Section 1.7.

1.4 Mentioned Locations and Mentioned Characters

The author is able to tag each quest with locations and characters that are mentioned in the quest. This is used for correlating plot point and quest selection which is described in more detail in Sections 1.7 and 2.3.

Goal State	
Name	Violet and James are broken up, Player dating James
ID	2
Preconditions	Colonel doesn't like the player much (low buddy)
Dialogue	"Please convince Violet and that stable boy to break up. You're more fitting for him, perhaps you can convince him to date you instead."

Figure 13: A sample goal state for the quest to break up Violet and James.

For this particular quest, because the quest involved Violet and James, they are both listed in mentioned characters. However, no locations are mentioned, so none are listed.

1.5 Goal States

Each quest is associated with one or more goal states. A goal state is an explicit goal that the player can be given by the quest giver. This is different from a completion state, explained in Section 1.6, which describes a state in which the quest is considered complete. For the example quest in Figure 12, there are 3 possible quest goal states that would be possible to receive from the quest giver. They are: break up Violet and James (default), break up Violet and James and date Violet, and break up Violet and James and date James. The last is shown in Figure 13.

A goal state is defined by a *name, ID, preconditions,* and *dialogue.* The *name* and *ID* are used for internal representation, and showing the player which goal they were given.

Completion State	
Name	Violet and James are planning to elope
ID	3
State	status(eloping with, Violet, James)
Scene	Goal state match: This completion state is never requested, do not need to create a scene for state match Goal state mismatch: The Colonel is angry at the player

Figure 14: An example of a completion state for the quest "Convince Violet and James to break up."

The *precondition* of the goal state is in addition to any preconditions of receiving the quest itself. Therefore there is always a default goal state, in which there are no additional preconditions. For this example, if the Colonel has a particularly low opinion of the player, he will choose this goal instead of the default goal. The most specific goal is always chosen, with specificity calculated based on the number of predicates.

The *dialogue* is the dialogue exchange between the quest giver and the player for giving this particular quest goal. In the example shown in Figure 13, we show a representation of the dialogue for clarity, but in practice it would be a dialogue exchange between the player and quest giver.

1.6 Completion States

A completion state is a state defined by the author that the player can reach, and in which the quest is considered completed. Having only one completion state is the most similar to a traditional quest, but because we look merely at the state of the world as opposed to checking a set of flags, the player has more freedom in how they choose to reach that goal. It is possible for a designer to create a very specific state which will mimic a task-based quest, but by designing for a state instead of a set of actions, it is more likely that the designer will create a more goal-based quest.

By not prescribing a set of actions for the player to take, and instead testing for a completion state, we also avoid the issue found in many adventure games, in which there is a hidden set of actions the player must take. Any actions the player chooses to take that reaches the goal will match a completion state, not just those that were pre-determined by a designer.

For every goal state, there must be a matching completion state, such that GrailGM knows when a more specific goal state has been met. However, it is possible (and desired) to create other completion states. In particular, as seen in Figure 14, it becomes more interesting to capture an antagonistic approach to the desired goal states. For instance, if instead of convincing Violet and James to break up, we make their romance network values so strong that they decide to elope; this should cause the quest giver to respond in some way. This gives the player more interesting and meaningful choices at the quest level, as their choice to work against the quest is met with believable consequences.

Once a completion state has been met, the quest is considered completed. If more than one completion state has been met, the most specific completion state is chosen, based on the number of predicates describing that state. This was done to prevent players from meeting one completion state and then later meeting another. This has a side effect in which it is much harder to meet some of the other quest goals, as seen in our example quest. There are social moves available (*Seduce, Ultimatum)* in which the player may simultaneously break up two characters, and date one of them, but it is a much more difficult move. This is a design issue that still requires refinement.

A completion state is defined by a *name, ID, state,* and *scene*. The name and ID are again used for internal representation.

The *state* is the description of the state in which the quest would be considered finished. For the example quest in Figure 12, the completion states are the three that match the goal states – Break up Violet and James, Break up Violet and James and date Violet, Break up Violet and James and date James – and the antagonistic completion state of convincing Violet and James to elope. We use the term antagonistic here, not to imply a value judgment on the player, but rather to denote that they are working against the stated goal of the quest, for whatever reason.

1.6.1 Scenes

The *scene* of a completion state is the list of outcomes and dialogue exchanges based on which goal state and completion state pair the player encountered. A scene is comprised of a *paired state ID*, *social change,* and *dialogue* as seen in Figure 15.

The *paired state ID* is the ID of a goal state. This ID signifies a goal state and completion state pairing. For instance, the scene example in Figure 15 matches the goal state ID in Figure 13, and this scene is part of the completion state shown in Figure 14. Therefore, this particular scene corresponds to if the player was given the goal to break up Violet and James, and date James, but completed the quest by reaching the state of convincing Violet and James to elope.

The *social change* of the scene is the outcome if that scene is matched. In this case, if the Colonel asked the player to break up Violet and James, but the player instead convinced Violet and James to elope, the Colonel is very upset about this. The outcome of this is that the Colonel gains the *angry at* status towards the player, and the Colonel's trust towards the player drops significantly, by 20 points (or falls to zero if the Colonel's trust towards the player is already lower than 20).

Scene	
Paired State ID	2
Social Change	Colonel gains *angry at* status towards player Colonel's trust towards player decreases (-20)
Dialogue	"I asked you to break them up, and instead you've managed to convince them to elope! I'm not sure what you're playing at, but get out of my sight."

Figure 15: A scene for the completion state "Convince Violet and James to elope" within the quest "Convince Violet and James to Break Up".

The *dialogue* is the exchange between the quest giver and the player, which expresses the outcome in dialogue form. Again, we have shortened the dialogue for representational purposes in Figure 15, the actual exchange would involve both the quest giver and the player.

While this may initially seem similar to a flag-based approach, it differs in a couple of key ways. The first is that quest completion is based on a specified state; we do not need to maintain and set flags for each possible combination of events which could lead to the desired state. This allows the designer to easily modify the desired states without worrying about coded flags. Additionally, we use default scenes for desired completion and undesired completion. This allows the author to create new start and completion scenes without needing to author more dialogue, but specific dialogue is supported.

This system allows the player latitude in how they choose to fulfill a quest, as well as give the system the ability to have that choice affect the game world and the story.

Finally, the effects of a scene are also stored in the Social Facts Database (SFDB) which, as described earlier, is used to reference social history. Future social actions and outcomes of actions can be modified or reference the outcomes of the quests.

1.7 Dynamic Quest Selection

In *Mismanor* – a social CRPG created using CiF-RPG and GrailGM – the player may only be on one quest at a time. This design decision was made as the intent for our game is for a shorter, more focused experience. It also allows the player to focus on the story they are creating without being sidetracked by too many quest objectives.

GrailGM is tuned such that there is a strong preference for the player to always be on a quest. Therefore, whenever a player has finished a quest, a new set of possible quests are chosen. GrailGM calculates the top three possible quests after every state change. The recalculation is done because we calculate the intent of a quest giver to give a particular quest, and this intent may change due to the state change. When a player interacts with a quest giver for a possible quest, they are offered the quest. Currently, the player always accepts a quest, although in future versions of the system, the player will be able to refuse the quest.

We initially considered making the act of giving a quest a social action supported by CiF-RPG. However, because of the desired shortness of our experience, we wanted to be able to have a finer control over when quests were made available. Therefore, we gave GrailGM control of quest giving. GrailGM is able to override a social move initiated by an NPC and instead have the NPC offer a quest. In a longer experience, it would make sense to move this functionality back to CiF-RPG to support micro-theories and influence rules about when a character might want to assign a quest.

However, micro-theories and influence rules still play a part in choosing which quests are available for GrailGM to choose from.

1.7.1 Choosing a quest

The GrailGM system chooses up to three possible quests for a player to be able to receive, although the player can only be on one quest at a time. The GrailGM chooses which quests

to make available based on, in order: prerequisites, author preference and character intent, and matching tags with possible plot points.

The first pass through the quest pool first looks at the hard constraints of preconditions for a quest. For every quest that has the preconditions met, it is added into a set of possible quests. The possible quests are then ordered based on author preferences encoded into heuristics and the weight of the intent for the quest giver to give that particular quest.

The author heuristics and intents are all given a normalized weight indicating how much of a part that heuristic should play. The heuristics currently supported within GrailGM are *quest mixing* and *quest concentration*.

For *quest mixing*, when looking at the history of quests, this heuristic prefers quests with different types than previously used quests. We use a geometric progression with a common ratio of ½ to score each quest in history. We do this to model decay over history while still preferring quest types that have never been used before. For every quest in the history that matches the type of the quest currently being considered, the score of that quest based on the geometric progression is added to a running total. We negate this score and multiply it by the weight assigned to quest mixing to reach a final quest mixing score.

Quest concentration is the opposite of quest mixing. Instead of preferring quests with different types, this heuristic prefers quests that match previously used quests. The calculation is the same, except that the final score is not negated at the end. Therefore, if quest mixing and quest concentration are given the same score, they cancel each other out.

Both of these quest heuristics are supported within GrailGM to give authors ways to fine tune the different heuristics and allow for different preferences in quest choice.

The author also chooses a weight assigned to how important a role the quest *intent* should play. To calculate the intent, we leverage CiF-RPG and calculate the volition of the quest giver to assign that quest. The volition is calculated by assigning the quest giver the initiator role, and other characters or items involved in the quest are assigned as the responder or other if needed. For instance, in our quest, Break up Violet and James, the Colonel is assigned as the initiator, Violet is assigned as the responder, and James is assigned as the other. This helps us calculate a weight of the intent for the Colonel's desire to break up Violet and James. The volition score is then multiplied by the intent weight assigned by the author.

Once the heuristics and intent have been calculated, the three scores for quest mixing, quest concentration, and intent are summed and the total score is assigned to each quest. The quests are then ordered based on total score, highest to lowest. The top three quests are chosen as the possible active quests.

If there are any ties between quests, they are broken based on the mentioned locations and characters assigned to the quest. GrailGM looks at the active plot points, and the quest with the most matches of characters and location with the active plot points is given a higher score. This is done to more strongly tie the game level and quest level stories, by making it more likely that a plot point is discovered that correlates with the current active quest.

The possible active quests are kept updated after every social state change, until the player interacts with a quest giver of one of the possible active quests. At this point, the quest is given to the player, and that quest becomes the current active quest. At this point, GrailGM no longer updates the possible active quests, but instead checks for quest completion.

1.8 GrailGM and playable quest level stories

GrailGM supports playable quest level stories in two ways. The first is that the system gives the player *interesting* choices in how they choose to complete quests by both offering quest goal states instead of prescribing a set of actions to take, as well as by supporting multiple completion states which gives the player options in how they might wish to fulfill (or not fulfill) a particular quest.

The system also offers the player *meaningful* choices in two ways. Each completion state for a quest has an outcome which impacts the social state of the game world that is consistent with how the quest giver would feel given the player's actions. The second is that which quests are even possible for the player to receive is based on the social state which the player is continually able to change.

In these ways, GrailGM provides the player with a more playable quest level story experience.

2 GrailGM game level Story Management

We are interested in not only giving the user playable quest level story choices through dynamic quests, but also allowing the player's actions to have an effect on the game level story line as well. To accomplish this, we needed to add a form of story management to

GrailGM. We used the knowledge that we gained from working with drama management (discussed in Chapter 7), and incorporated some of these ideas into GrailGM's story management system. To facilitate game level story management, we followed a scene-based content selection method in which we designate plot points which denote the important events within our game level story. GrailGM then uses heuristics to choose a small subset of plot points the player has the chance to uncover, and which one is discovered is based on the social moves the player chooses.

Plot Point	
Name	Colonel doesn't know about some rooms in the manor
Traits	*knowledge, cult line, colonel's line, plot point*
Mentioned Locations	*none*
Mentioned Characters	Colonel
Preconditions	*none*
Instantiations	Colonel (love): "I am embarrassed to admit it, but I feel I can trust you. It seems as if parts of the manor have started changing. There are doors where I don't remember them, and some seem to have gone missing. Do you think I have gone daft?" Violet (hate): "You're obviously suffering from some sort of mental issue. Next you'll be thinking the manor has moved some of its rooms around, like my dear old father thinks."

Figure 16: A sample plot point used in Mismanor, in which the player learns that the Colonel doesn't know about some of the rooms in the manor. Plot points extend Knowledge Game objects, and ID, Type, Statuses, Examine and Discuss Lines have been left out for clarity.

2.1 Plot Points

Plot points in GrailGM extend the Knowledge game object in CiF-RPG. Because we are interested in being able to dynamically shape the story using author-level heuristics and player choice, the major story elements are not (for example) placed in fixed locations in physical space or simply triggered by the player accomplishing in-game objectives. Instead, possible plot points are chosen by GrailGM (discussed in Section 2.3) and revealed during other social moves as dialogue mix-ins (discussed in Section 2.2.1).

2.2 Plot Point Structure

The plot point structure is shown in Figure 16, although some of the inherited structure has been left out for clarity. The name of the plot point is used primarily by the authors as an internal representation to distinguish different plot points. The other inherited variables (ID, type, statuses, examine lines, discuss lines) are all treated the same as the knowledge base class by CiF-RPG (which does not reason about plot points), and are unused by GrailGM.

The features of plot points that are used by GrailGM are *traits, mentioned locations, mentioned characters, preconditions,* and *instantiations.*

A plot point's *traits* are the same traits as used in CiF-RPG. Traits are used to describe plot point features that do not change over time. By default, every plot point has the traits denoting that it is a knowledge object, and that it is a plot point object. We also use the traits to track which storylines the plot point is a part of, such as the example plot point which is a part of the cult main storyline and the Colonel's personal storyline. This is used

during dynamic plot point selection. It is possible for a plot point to be part of multiple storylines (such as our example), and therefore both storylines are represented.

The *mentioned locations* and *mentioned characters* are discussed in Section 1.4 when describing quests, and they are used in the same way. Mentioned characters allows the designer to tag any characters that are mentioned in the plot point. For the plot point shown in Figure 16, the Colonel is the only character that is mentioned. Mentioned locations likewise lets the designer denote any locations that are mentioned when the plot point is revealed. These tags are used by GrailGM to help weight which quests and plot points should be added to the possible active list.

The *preconditions* for a plot point refer to hard constraints in the story. For instance, before the player can learn that there is a cult ritual happening in the manor that evening, they must learn that there is a cult. The fewer preconditions placed on the story, the more room GrailGM has for recombining plot points. However, preconditions are necessary for retaining a coherent story.

2.2.1 INSTANTIATIONS

Plot point *instantiations* is the way that the plot point is revealed to the player. Each instantiation has a *speaker, mood,* and *dialogue.* A plot point is revealed to the player through a dialogue exchange. The dialogue exchange is mixed-in to the dialogue exchange resulting from a social move. Speaker and mood are both used to determine whether it is possible to mix-in the instantiation dialogue with the social move dialogue.

Plot points can be revealed by one or more characters, but how it is revealed is different based on the character revealing the plot point, as well as the mood of the current dialogue. Therefore, each possible plot point dialogue exchange is associated with a *speaker*, or character that is participating in the dialogue exchange with the player. This allows the designers to retain control over how the plot points are revealed and who can reveal them.

The *mood* of the instantiation denotes the tone of the dialogue exchange. When creating plot points, our writers chose the moods *love, like, neutral,* and *hate* to designate the tone of the dialogue. This merely describes the mood of the dialogue at that current moment, not the feelings between the characters themselves.

When creating instantiations for social moves, the writers are able to denote plot point mix-in points with $love$, $like$, $neutral$, and $hate$. When a particular instantiation is chosen, if it contains any of these tags, GrailGM will search the pool of active plot points for any that are a possible match. A match is found if there is a mood and speaker that matches the mix-in tag and the character that the player is interacting with.

For instance, if the player is engaged in a *Petty Argument* with Violet, and the plot point in Figure 14 is in the active list, it would likely be matched since the $hate$ mix-in would likely be used in an instantiation associated with the *Petty Argument* social move. We do not require the authors to create each combination of game character and mood; if there is no matching plot point mix-in for the current interaction context, we silently move past the mix-in point and leave the plot-point available for a future social interaction.

This system allows the author to have some control over which characters reveal which plot points, as well as what ways it can be revealed. Not every character need be able to reveal every plot point. For instance, the Colonel does not know about the cult, so it should not be possible to learn that plot point through him. The system provides enough flexibility for the player to still find game level story content but have it tailored to who they choose to interact with, and how they choose to interact with them.

2.3 Dynamic Plot Point Selection

GrailGM uses a tiered constraint system to choose which plot points are available for the player to uncover. At any given time, up to three plot points are possible for the player to reveal.

GrailGM initially looks through all available plot points, and tests that their preconditions have been met. These plot points are put into a pool of possible plot points. The possible plot points are then scored based on three built-in heuristics, *story cohesion, story mixing,* and *story concentration.* Each heuristic is given a normalized weighting by the author to designate how strong of an influence it should have on plot point choice. With *story mixing* and *story cohesion*, the author supplies both a minimum and maximum for these heuristics, as they change over time.

The *story cohesion* heuristic scores plot points based on the number of story preconditions that have been met to get to that plot point. This means that it prefers plot points that have a large number of preconditions or those that are deep within a story tree. Each

precondition contributes one point towards the story cohesion score, which is then multiplied by the author-chosen weight for story cohesion.

The *story mixing* and *story concentration* heuristics work similarly to the quest mixing and quest concentration heuristics used by GrailGM in dynamic quest selection. Each looks at the history of revealed plot points, and assigns a score that decays using a geometric progression with a common ration of ½. Then all plot points within the same storyline (tracked by the plot point traits) are summed for a final score. For story mixing, this score is negated and then multiplied by the story mixing weight, while story concentration is not negated.

Over the course of the game, it was desired by our writers that the game would initially have a stronger weighting for story mixing, to give the player a chance to see parts of each storyline. However, as the game progresses, and plot points are revealed, story concentration would grow in strength, to lead the player towards an ending. Therefore, story mixing starts at the author-defined max, and declines to the minimum, while the story concentration starts at the minimum, and rises to the maximum.

The final weighted scores are summed together and assigned to each plot point. GrailGM then orders them and chooses the highest three weighted plot points to be active, or possible to be revealed by the player. If there are any ties, GrailGM looks at the active quest, if any, and scores the number of matching characters and locations that are mentioned in both the active quest and the plot point. The plot point with the highest score is chosen. If there is still a tie, a plot point is chosen at random.

Game Ending	
ID	4
Ending State	Violet has high romance towards the player (>75) Violet has low familyBond towards Thomas (< 30) Player has low buddy towards Thomas (< 50)
Ending Text	"Realizing her love for you and her distaste for her brother, Violet chooses to sacrifice Thomas instead of the you. Unfortunately, the sacrifice fails, and you and Violet are haunted by the death of Thomas."

Figure 17: One of the five pre-scripted endings for Mismanor.

When a player reveals a plot point, GrailGM re-calculates the list of three active plot points, as the recently revealed plot point will change the list of possible plot points, as well as the weights assigned to them. When the player reveals an ending plot point (denoted by having the *ending* trait) the game is over and a game ending is chosen.

2.4 Game Endings

To truly have meaningful choices within the game level story, there needs to be multiple paths which can lead to multiple endings, chosen based on the player's actions within the game. Therefore, GrailGM allows the author to describe an arbitrary number of game endings. A game ending is made up of an *ID, ending state,* and *ending text.*

The *ending state* is specified by a set of predicates which specify the required state to get that ending. For the game ending shown in Figure 17, the ending state requires Violet to be romantically close to the player, and both Violet and the player to not be close to her brother, Thomas.

When an ending state is matched, the player sees the *ending text* that matches that state. This is a textual description of how the game ended. In this particular example, because Violet loves the player so strongly, and has such low regard to her brother, she chooses to sacrifice her brother instead of the player in the cult ritual.

2.4.1 Generated Game Endings

It is not realistic to assume that a designer will want to cover every particular ending possible for the game. Therefore, if no specified ending is found, the game will generate an ending based on the social standing of the player with the other characters in the game. To do this, GrailGM looks at the different network values for each non-player character towards the player. The character that has the most networks that are 75 or higher is used in the generated ending. The high-level networks are also referenced in the ending.

If there is more than one character with high enough network values, GrailGM will use the character with the most network values over 75. In the case of ties, all network values are looked at, and the character with the highest total across all networks is used for an ending. This is done to find the character with the strongest ties towards the player's character, as they are the character most likely to help the player.

As an example, if at the end of the game, Liz has a romance value of 80, a buddy value of 45, a trust value of 65, and a familybond value of 30 towards the player, Liz would be considered for an ending due to the romance value being over 75. If at the same time, James has a romance value of 35, a buddy value of 85, a trust value of 30, and a family bond of 25 towards the player, James would also be considered for an ending. However, Liz has a

higher total value across all networks, so the ending would be created involving Liz. In this case, the ending will state “Because of Liz’s romantic feelings towards you, she chooses to help sneak you out of the manor, and save you from being sacrificed.”

If the player has no high relationships with any of the characters, and does not match any of the pre-defined endings, the player will receive the “bad” ending in which they do not escape from the cult, and is instead sacrificed. In this case, the player receives the ending, “Sadly having made no alliances in the manor you are sacrificed to raise Violet's mother. Unfortunately since you die, you don't even get to see if the sacrifice worked.”

3 Conclusion

To create playable story at the game and quest level, the player needs to have interesting and meaningful choices within quests and the overall story arc. GrailGM is an AI framework which supports both dynamic quests and plot points, providing a structure which allows player choice to have a meaningful impact on the way the stories develop.

By having one system that supports the story at both the quest and game level, the system is able to keep the two levels tightly coupled. This helps keep a strong coherence throughout the different levels of the story, while maximizing the meaningfulness of the choices offered to the player.

CHAPTER 6

TOOLS

1 INTRODUCTION

Creating systems that support dynamic storytelling is only part of the solution. When developing a new AI-based infrastructure for dynamic content, it is also necessary to create the tool chain to enable authors to create the new kind of content required. This turns out to be a non-trivial task.

There appears to be a two-fold problem. The first issue is being able to generate enough ideas for the content requirements of a dynamic content selection system such that the system has enough content to make flexible choices in response to the player, rather than being forced into a decision merely due to a lack of choices. The second problem is that the content required by the system needs to be authored to support re-sequencing. Instead of creating dialogue and stories that can only be used in fixed contexts, the designer needs to create content that can be re-ordered, and lines of dialogue that can potentially be spoken by multiple characters.

To begin addressing these issues, we created two tools. The first, Brainstormer, was created to aid the designer in the idea phase; to be able to brainstorm different quests, and different possibilities for a given quest. The second, the Social Mechanics Design Tool, was created for content authoring in CiF-RPG and GrailGM.

2 GRAIL FRAMEWORK BRAINSTORMER

While GrailGM supports playable quests, to truly achieve this playability, the designer needs to be able to create a large number of possible goal states and completion states for each quest. It is easy for designers to fall into a pattern of using the same types of quest ideas and possible ways of completing the quest. To alleviate the difficulties in considering multiple interesting goal and completion states for each quest, we created the Grail Framework brainstormer tool.

There are other tools to help designers brainstorm game design such as Grow-A-Game (Belman, Nissenbaum, Flanagan, & Diamond, 2011) and Deck of Lenses (Schell, 2008), however there are none to our knowledge that are aimed at the quest domain. When focusing on quest design in particular, most tools are proprietary and therefore not discussed openly. The exception to this is ScriptEase (McNaughton et al., 2004), a designer tool created to work with the *Neverwinter Nights* (BioWare, 2002b) Aurora toolset (BioWare, 2002a). ScriptEase follows a pattern-based approach to authoring, with many of the common designing tasks available as a pre-scripted selectable component in the tool. Many of the standard quests regularly found in CRPGs are available as a pattern in the quest library. These patterns are extendable so that a designer is not restricted to just the quests available in the library. Once created, these quests are playable within a *NeverWinter Nights* module; however, the quests are statically placed and do not change based on the player's actions. In contrast, Grail Framework Brainstormer is not a tool for tailoring specific quests, but rather for helping the designer think about the possibilities of quest design.

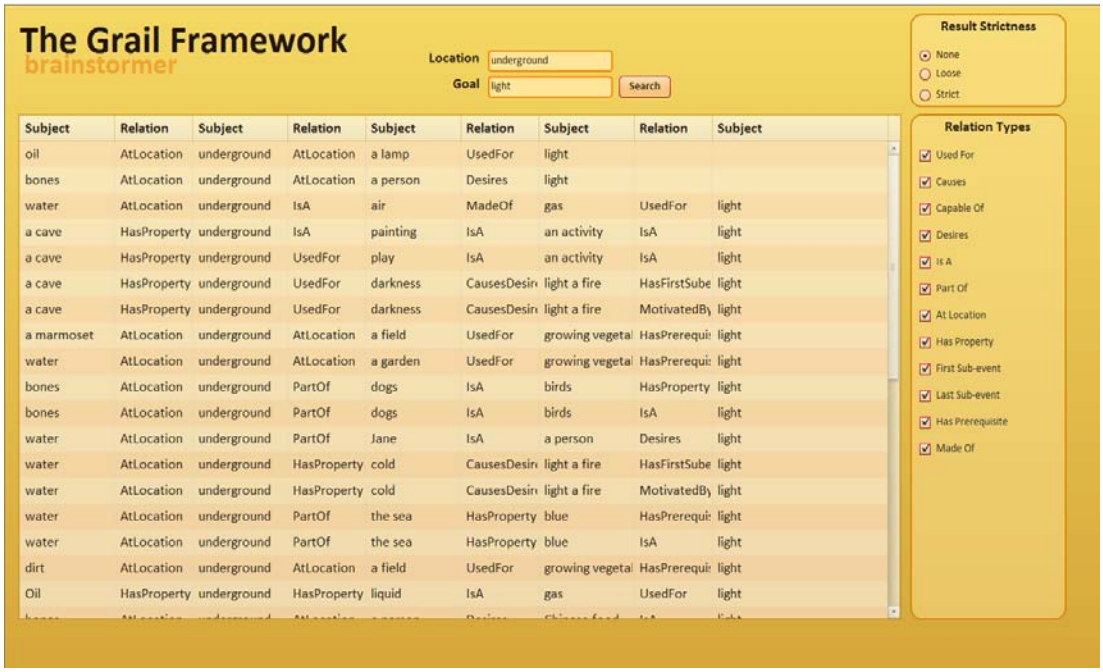

Figure 18: The Grail Framework brainstormer tool. This allows designers the ability to leverage a common-sense database as a method of brainstorming.

When designing quests, a designer must first identify an issue that the quest giver would want solved. Then one or more solutions to the issue must be found. These solutions need to be something the player is able to fulfill using the supported game mechanics.

The Brainstormer tool allows the designer to find relationships between quest concepts. As an example, the designer may choose to create a quest that is located in a church, forest, cave, alley, or city. The designer can be as specific or general as they would like. When the designer enters one of these locations into Brainstormer, a list of concepts associated with the location specified is created. If the designer decides they would like to create a quest for a cave, Brainstormer will generate a list of concepts including *underground*, *dark*, and *mine* among others.

Given these concepts, the designer may decide that they would like to make a quest involving a darkened underground room with the objective of finding light so that a player may cross the room. Certain ideas may initially come to mind, such as finding a torch or building a fire. However, the brainstorming tool will return a table displaying all paths with 5 or fewer concepts between *underground* and *light,* showing ideas that also make sense but may not come to a designer's mind.

The following are some examples returned by Brainstormer for the situation above:

Underground ←(*At Location*)– **Rock** –(*Part Of*)→ **Moon** –(*Capable Of*)→ **Light**

Underground ←(*At Location*)– **Fungus** –(*Capable Of*)→ **Bioluminescence** –(*Capable Of*)→ **Light**

Underground ←(*At Location*)– **Coal** –(*Used For*)→ **Burn** –(*Capable Of*)→ **Light**

In unassisted quest authoring, the designer has to imagine properties of a location that the player might change (such as dark), and imagine ways to change them. Using a tool such as Brainstormer, designers discover interesting and unusual quest possibilities, lowering the burden of creating interesting and meaningful choices for the player when presented with a quest.

The Grail Framework Brainstormer tool finds concepts by leveraging the crowd-sourced common-sense database ConceptNet 3 (Havasi, Speer, & Alonso, 2007) to find links between quest-related ideas. One of the benefits of using ConceptNet is that the database is able to easily supply relationships between objects that are unusual or surprising in some way as it does not have a personal bias or preference for a specific outcome. ConceptNet 3

structures its data by storing concepts as nodes which are connected to other nodes based on their relationships. There are currently twenty discrete relationship types, such as *PartOf*, *ConceptuallyRelatedTo*, *UsedFor*, *CapableOf*, and *LocationOf*.

ConceptNet 3 is mined from a crowd-sourced corpus of natural language sentences about the world, and consequently contains a fair amount of noise. Through experimentation we determined that it is necessary to limit which link types we use, eliminating the most abstract link types, such as *ConceptuallyRelatedTo*, in order to not swamp generally useful quest suggestions in a sea of bizarre results. The abstract link types ended up having the most amount of noise due to the less-defined nature of the relationship.

We chose to use ConceptNet3 in particular because each concept / relation / concept triple is scored by other users, unlike previous versions of the system. This score is stored in the database and available for use in filtering responses returned by the system. We also limit the length of the paths returned by ConceptNet to 5 nodes. This was chosen after experimental runs showed a higher number of noisy results when paths became too long. Requests to the database also began to noticeably slow down when paths became 6 nodes or longer.

2.1.1 GUI

In our interviews with Sony Online Entertainment quest designers, it became clear that the user interface was an overlooked portion of quest design tools. The in-house design tools required the designers to enter information multiple times, as well as remember numerical tags for database information. Other teams required the quest designers to enter the same

variable name in multiple text files. This caused a number of entry issues, as well as frustration. While we are not primarily working on the human-computer interaction elements of the quest-building problem, it is still important that the designers be able to successfully use what we create. We therefore felt it was necessary to have a basic graphical user interface (GUI).

To interact with the system, a designer supplies the GUI tool with a quest concept, and optionally, a possible quest objective. The system returns a list of linked paths through the knowledge space that connect the starting concept and quest objective specified. This gives the designer possible ways to reason about the relationships between these objects. If a designer does not specify a goal or objective, Brainstormer will show all nodes directly connected to the concept node chosen. In this way, if the designer has a concept in mind, the tool can help the author think of quest goals.

For Brainstormer, we feel that it is important for the user to be able to constrain the possible space, as the number of responses can be quite large. Using the UI, the designer is therefore able to manipulate the quest paths found by requiring or excluding nodes, constraining the types of links between the nodes and requiring results to be above a certain reliability score. This allows the author to narrow down the field of results in a way that suits their needs.

2.1.2 Nodes

Each concept is stored as a node within ConceptNet3. Nodes are represented as “Subjects” within our GUI. It is important to give the designer the ability to restrict the results in

various ways. For some designers, results returned might be funny, while for others, it would be inappropriate. An example of this is the following solution returned for the Starting Subject: Underground, and the Goal Subject: Light.

> **Underground** ←(*At Location*)– **Oil** –(*Used For*)→ **Eating** –(*Creates*)→ **Gas** –(*Used For*)→ **Light**

For these cases, the designer is able to exclude the *eating* node from any further responses if they would like to avoid these types of solutions. Similarly, the designer is also able to require nodes. For instance, if the designer wished to see more solutions using the idea of *oil*, they could require it in further solutions, and only paths including the node *oil* would be shown.

2.1.3 Relation Types

ConceptNet3 links nodes based on the type of relation between them. These are unidirectional links, so while *oil* has the link *at location* to the node *underground*, there is not necessarily a link from *underground* to *oil*. For this reason, it is important to allow the tool to traverse links in both directions to find the relations between nodes. Otherwise, the results returned are unnecessarily limited by the system.

To increase the authorial power of the tool, we have given the designer control over which relation types are used in the results. In this way, the designer is able to narrow down the results returned. For instance, some designers may wish to enforce the temporal relation of subjects, and will want the results to include relation types such as *has subevent*, *has*

prerequisite, etc., while another designer may wish to exclude those types of relationships from their results.

These constraints are not always necessary, and often the default relationship links return usable quests. For instance, choosing church as the original concept and heal as the quest objective generates the following non-obvious idea as one of its solutions:

Church –(*LocationOf*)→ **Music** –(*CapableOf*)→ **Heal**

2.1.4 Example

To illustrate how a designer might use the Brainstormer tool, we have created the following example. In our fantasy setting, there is a wooded town named "Forest Song." The designer therefore has the starting state seen in Figure 19.

Town:	Forest Song
Problem:	
Reason:	
Solutions:	

Figure 19: An example starting state for a quest designer.

By entering Forest as the starting subject, and song as the goal subject, Brainstormer will return the concepts shown in Figure 20. Looking through the results found, the designer might spark off the idea that the town of Forest Song was named after the bird song so common in the area. Therefore an interesting quest could revolve around the fact that the birds have quit singing, and the inhabitants are understandably concerned by this.

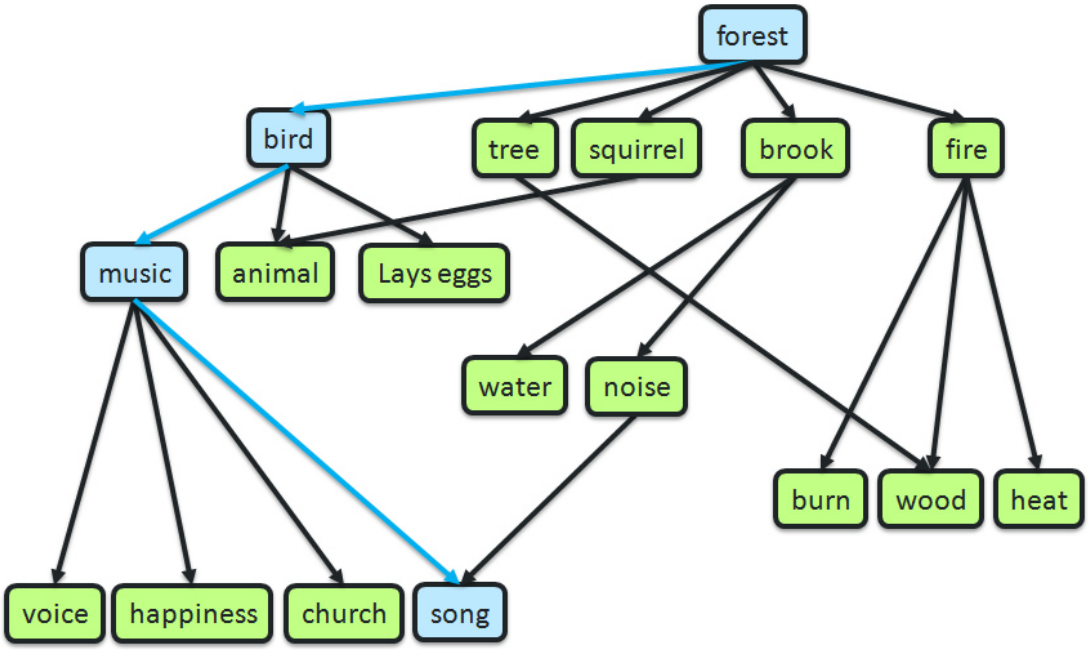

Figure 20: Subset of the tree of concepts returned by the brainstormer tool for starting subject "Forest" and goal "Song".

Town:	Forest Song
Problem:	The birds have quit singing.
Reason:	
Solutions:	

Figure 21: After using the Brainstormer tool, the designer decides the issue to be solved for this quest is that the "Birds have quit singing."

The designer therefore chooses to have the issue for the quest revolve around the fact that the birds of Forest Song have quit singing, as seen in Figure 21.

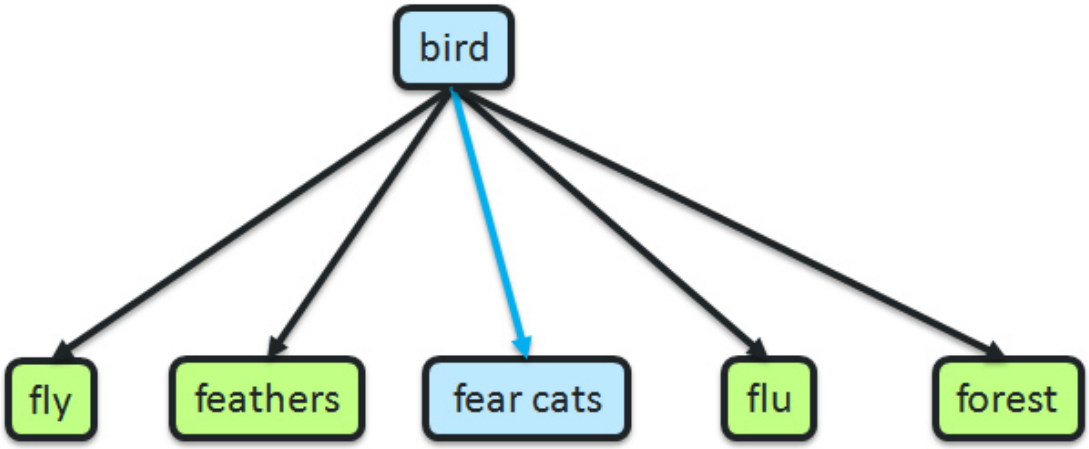

Figure 22: Entering the concept of "bird", the Brainstormer tool returns all concepts connected to bird. The results have been pruned for space.

At this point, in order to determine some possible solutions to the quest, the designer needs to specify the reason the birds are no longer singing. By putting just the concept of bird into Brainstormer, the tool will return all concepts connected to that node, filtered based on author-specified preferences. Based on the results shown in Figure 22, the designer in this case chooses to focus on a bird's natural fear of cats.

Town:	Forest Song
Problem:	Birds aren't singing.
Reason:	Wild cats are scaring the birds.
Solutions:	

Figure 23: Looking at the results for bird, the designer chooses to focus on a bird's natural fear of cats as the reason for the birds no longer singing.

Town:	Forest Song
Problem:	Birds aren't singing.
Reason:	Wild cats are scaring the birds.
Solutions:	[combat] Kill X number of wild cats.

Figure 24: Typical combat solution for the problem, kill the wild cats scaring that are scaring the birds.

Now that the designer has decided why the birds are no longer singing (Figure 23), a goal for the player to achieve has also been decided. The goal of the quest is for the player to get the birds to sing again by dealing with the wild cats.

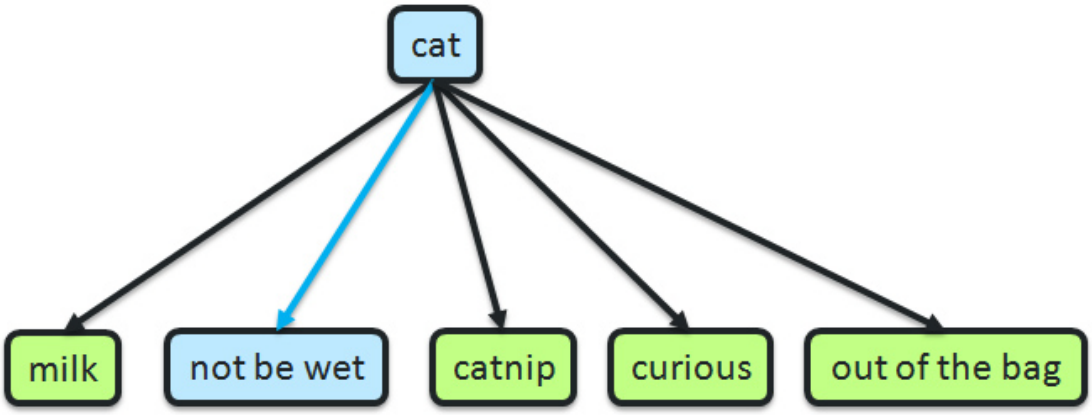

Figure 25: Using the results from the Brainstormer tool, the designer chooses a solution based on cats' general desire to not be wet.

The next step is for the designer to find solutions for this quest. In a typical role-playing game, the designer would designate a kill quest (Figure 24), in which case the player character is asked to kill a certain number of wild cats to rebalance the population. This

would be fine for a player character that is attack based, or for a player who prefers combat interactions.

To find solutions for other play styles, the designer inputs cats into the Brainstormer system, to help brainstorm ideas for non-combat quest solutions. Figure 25 shows the pruned subset of nodes that the designer found interesting. For the first quest solution, the designer focuses on the fact that cats do not like to be wet. In this case, the author designs a solution in which the player would be able to stealth past the cats and create moats around the nesting trees of the birds. This would allow the birds to stay in Forest Song without fear of the cats catching them.

Town:	Forest Song
Problem:	Birds aren't singing.
Reason:	Wild cats are scaring the birds.
Solutions:	[combat] Kill X number of wild cats. [stealth] Stealth past cats, create moats around trees.

Figure 26: Using the idea that cats don't like to be wet, the designer creates a possible quest solution of creating moats around the birds' nesting trees.

Looking at the nodes again, the designer decides to look at the fact that cats desire catnip (Figure 27). Spell-casting characters might be able to create a catnip potion to befriend the cats. Once befriended, the villagers can begin feeding the cats. This will keep the cats from needing to feed on the birds, and allow their song to fill the forest again.

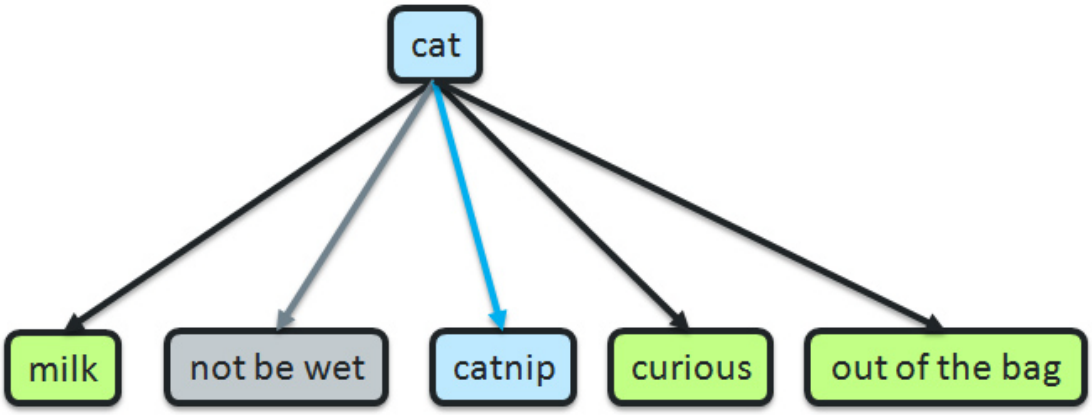

Figure 27: For an alternative solution, the designer chooses to work with the idea that cats love catnip.

Town:	Forest Song
Problem:	Birds aren't singing.
Reason:	Wild cats are scaring the birds.
Solutions:	[combat] Kill X number of wild cats. [stealth] Stealth past cats, create moats around trees. [potion] Create a catnip poultice to befriend the cats.

Figure 28: Using the concept that cats love catnip, the designer creates a solution in which the player could create a catnip poultice to befriend the cats.

Finally, the last solution the designer uses is a bit tongue-in-cheek and uses the concept of "letting the cat out of the bag." The player can use milk and cats' natural curiosity to tempt the cats into bags. Once bagged, the cats can be moved to a new location with plenty of mice for the cats to eat (Figure 29). This solution is one that any style of player can complete.

Town:	Forest Song
Problem:	Birds aren't singing.
Reason:	Wild cats are scaring the birds.
Solutions:	[combat] Kill X number of wild cats. [stealth] Stealth past cats, create moats around trees. [potion] Create a catnip poultice to befriend the cats. [all] Use milk to tempt cats into bags – relocate.

Figure 29: Using the concepts of "letting the cat out of the bag", that cats like milk, and that cats are curious, the designer creates a final solution to tempt the cats into bags with milk and relocate them.

In this example, it is important to note that the designer is still a key part of the design process. Brainstormer is a tool to help the designer think of possibilities that use common sense, but the designer is necessary in putting it all together. Additionally, Brainstormer is not domain specific, so it is up to the designer to decide if or when quests should be class or player specific.

2.1.5 Future of Brainstormer

In order for the designer to realistically be able to create the number and variety of solutions required for playable quests, we created Brainstormer to aid the designer in quest creation. Brainstormer is built upon a common-sense knowledgebase created by thousands of people, therefore the knowledgebase does not have the biases or preferences that just one person would. Consequently, using this system allows the designer to see past their own quest preferences and routines, finding new and innovative solutions for each quest.

This keeps the player engaged as they are offered interesting choices at each quest, instead of the same single solution that is common in many modern-day CRPGs.

While the Brainstormer system is a helpful for brainstorming, there are two ways in which it could be made much stronger. The tool currently displays information as a list of text, which can be very overwhelming and hard to read. It is unfortunately not intuitive how to interact with the information presented. By converting the system to use a graph-based interface with concepts represented as boxes, and relationships represented by edges, it would be a much more intuitive way to deal with the information presented. By making nodes and relationships clickable objects, it would be easier for a user to include or exclude those particular nodes or relationships. This allows for the system to be understood more readily and gives the user a more intuitive way to constrain the design space.

The second way the Brainstormer could be improved is by making the relationships between concepts just as important as the concepts themselves. Currently, most of the importance is placed on the concepts, as seen in our example where the relationships were used merely to help constrain the space. However, by being able to focus on relationships between concepts, it allows for more abstract brainstorming. For instance, if a designer wanted to create a quest about love, it would be very helpful to be able to find all nodes that share the relationship “Loves”, and then be able to constrain the nodes in a useful manner. This would require a creative way to constrain nodes, but could make for a much more powerful tool.

While ConceptNet works fairly well for real life domains, many computer role-playing games are set in science fiction or fantasy settings. ConceptNet breaks down on this as there is

little to no information for these domains. Working with different corpora which are domain-specific would greatly alleviate this issue. Using a system such as Microsoft's MindNet (Vanderwende, Kacmarcik, Suzuki, & Menezes, 2005), which can automatically generate a knowledgebase from text, along with canonical domain-specific books such as *The Lord of the Rings* trilogy (Tolkien, 1954), could allow for quick generation of domain-specific corpora. This would enable the Brainstormer tool to more easily support common CRPG settings.

3 Social Mechanics Design Tool

The Brainstormer tool was created to help with abstract quest design. The social mechanics Design Tool (SMDT) on the other hand, was created to help authors and designers generate concrete, usable information for the Grail Framework.

SMDT was created based on the design tool used by the *Prom Week* team for use with CiF (more discussion of CiF can be found in Chapter 4). SMDT was rebuilt from the ground up to include support for CiF-RPG and to address the requirements for creating a social CRPG. Additionally, SMDT was created with a desire to be more designer-friendly and with a focus on human readability as opposed to programmer notations. The SMDT has four subsections: Microtheory Creation, Social Moves Creation, Plot Point Creation, and Quest Creation.

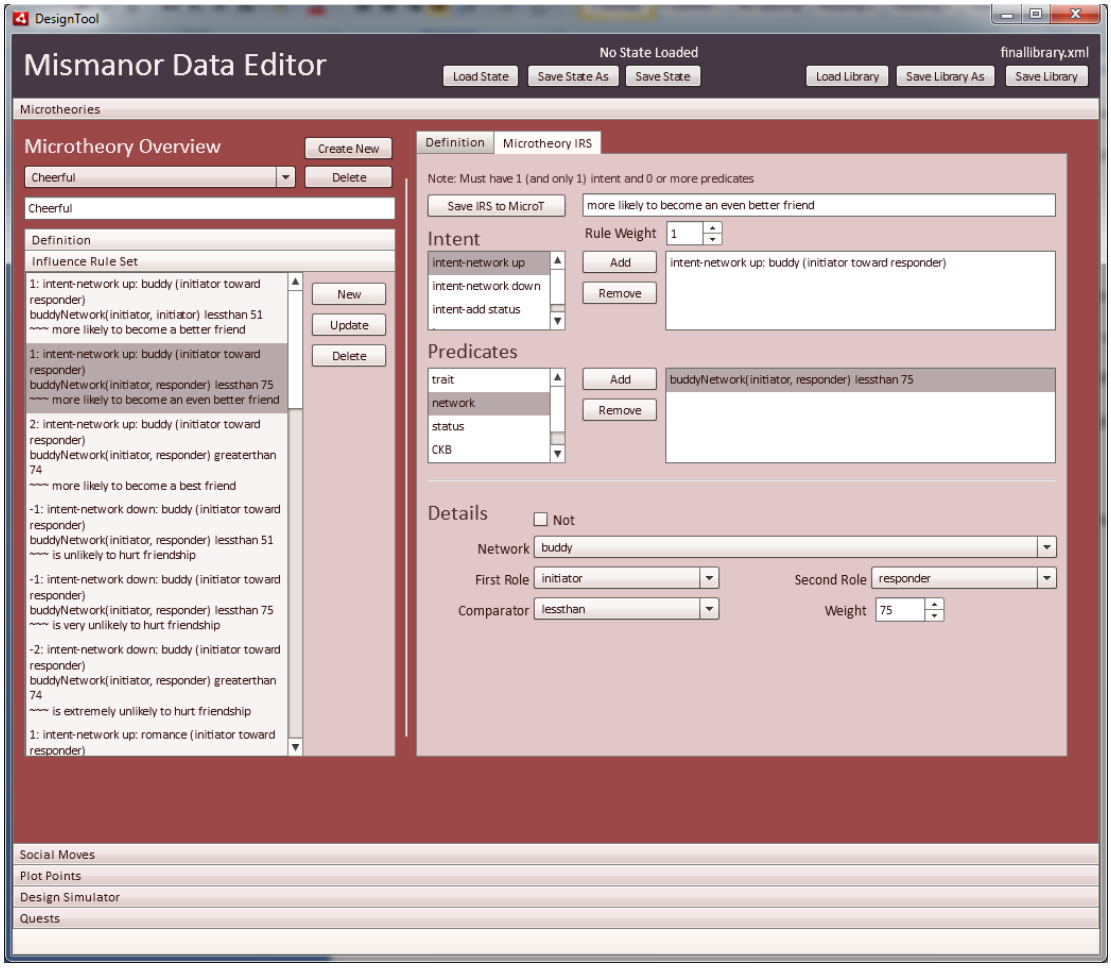

Figure 30: The microtheory editor portion of the SMDT.

3.1.1 Microtheory Creation

Microtheory creation is the subset of the tool used to create the micro-theories describing the traits, statuses, and relationships within the game world (Figure 30). A micro-theory consists of a definition (what the micro-theory is defining) and a set of influence rules (intention weights associated with different rules about the thing the micro-theory is defining).

The SMDT gives the authors the ability to provide a single definition, and any number of influence rules and weights associated with that micro-theory. In addition, the author can designate an author note used to help textually denote what they were trying to describe within the predicate logic. This was useful for not only the authors when they returned to a micro-theory later (similar to comments within code), but also for the programmers who would sometimes need to fine tune the authored micro-theories.

Figure 31: The screen for creating effects for a social move within the SMDT.

3.1.2 Social Moves Creation

The SMDT's most complex section is devoted to social move creation. There are 7 sections within the tool: Role Types, Intent, Pre-Conditions, Initiator Influence Rule Set, Responder Influence Rule Set, Effects, and Dialogue.

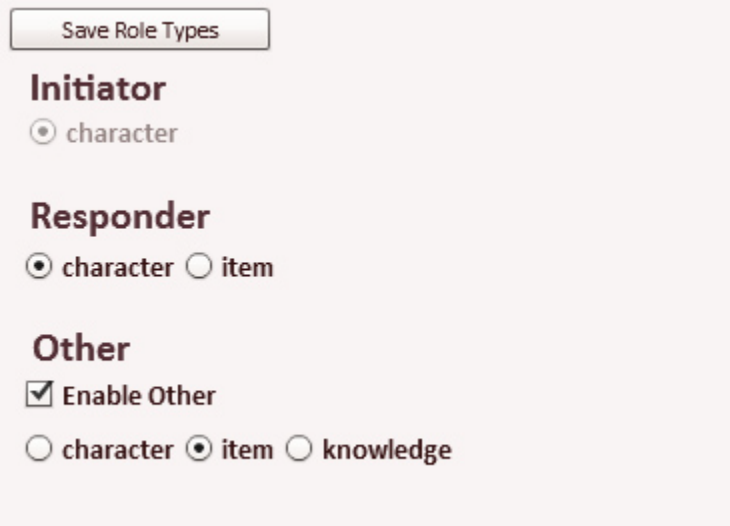

Figure 32: The interface for selecting roles for a social move

Role types is the section in which the designer is able to assign the type of game object for each role, as seen in Figure 32. Most social moves default to a character as an initiator and responder. However, moves such as *Drink* and *Pick Up Item* have a responder of type Item. Games that can include three objects specify an *other* that can be a character, an item, or knowledge object. These are all assigned in the Role Types section.

The *intent* section (seen in Figure 33) is used to designate the initiator's goal for the social move. The possible intents are: relationship up, relationship down, status gain, and status lose. The designer is able to specify the intent type, and the specifics of the relationship or

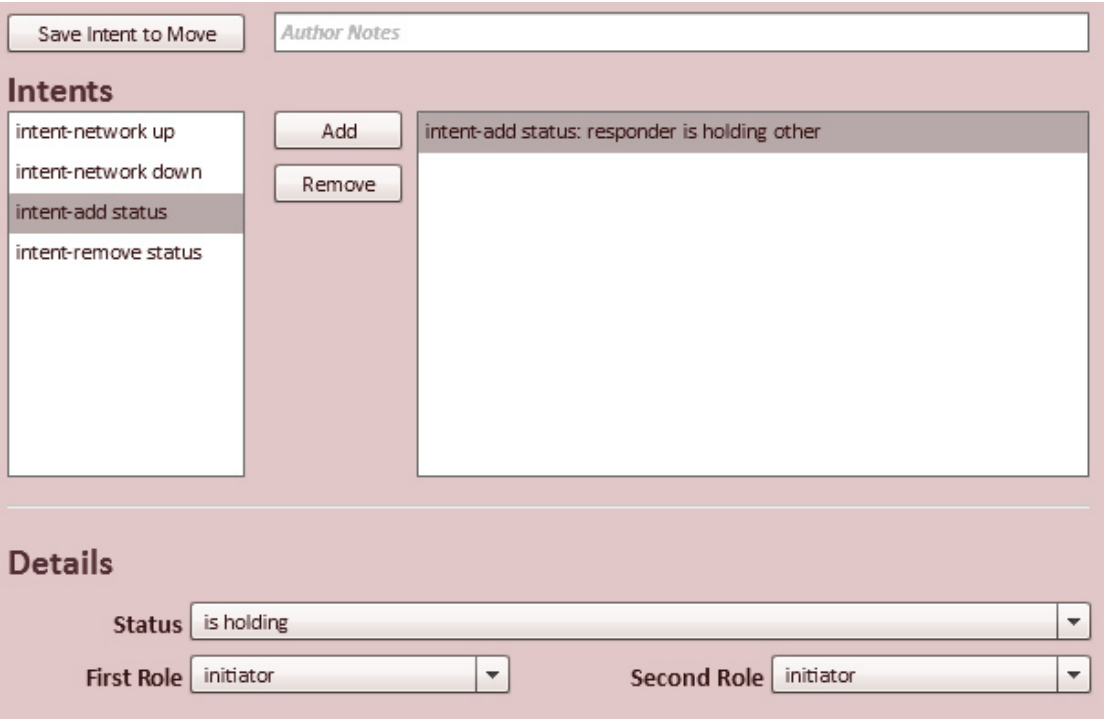

Figure 33: SMDT interface for choosing an intent for a social move.

status that the initiator is trying to modify. Item and knowledge gain and loss is handed through statuses, so these types of social moves are rolled into status gain and status lose.

Pre-conditions is the section where the designer may specify the predicates using the predicate editor (seen in Figure 34) that must be true for the social move to be available. This is not used very frequently as we prefer the influence rule sets to have the power to make a social move more likely to happen. However, for moves such as Break-Up, it is necessary to have a pre-condition that the initiator and responder have the status "is dating" towards each other.

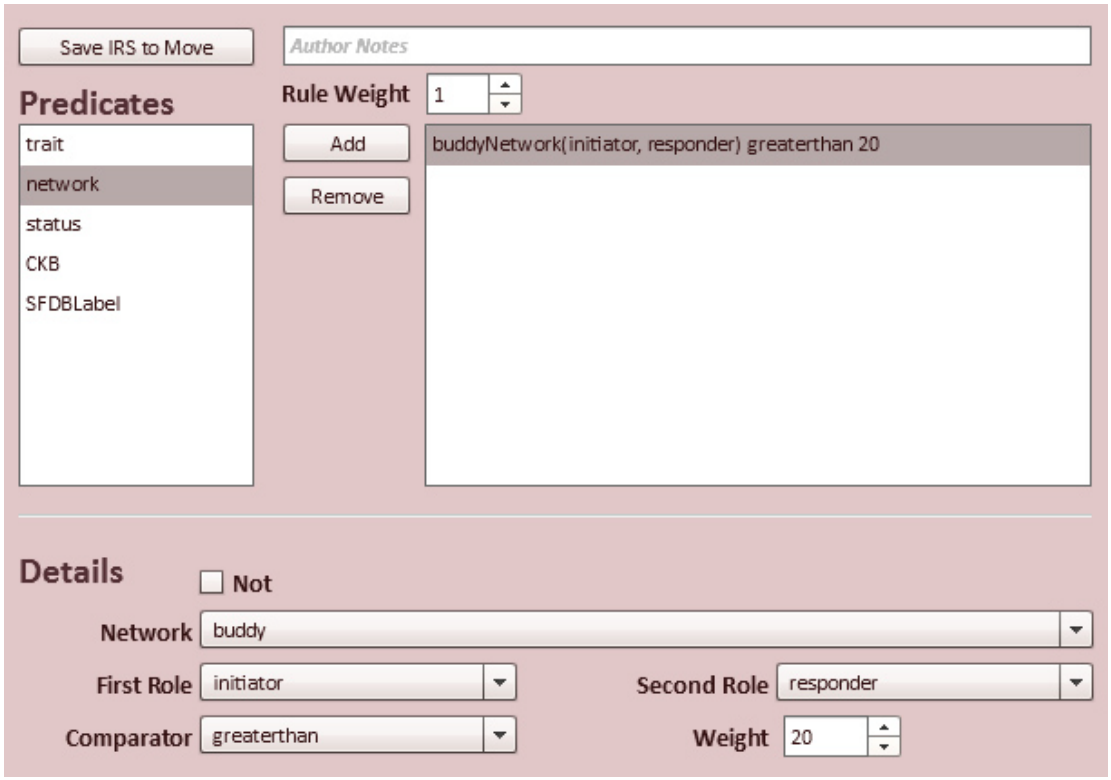

Figure 34: The predicate editor is used for creating preconditions, initiator influence rules and responder influence rules.

The *initiator and responder influence rule sets* sections are where the designer may add lists of rules by using the predicate editor (seen in Figure 34). The rules specify move-specific weights to modify whether a move is likely to be initiated (initiator influence rule set) or accepted (responder influence rule set). The designer is able to specify a set of predicates that make up each rule, as well as a positive or negative weight that is associated with that rule.

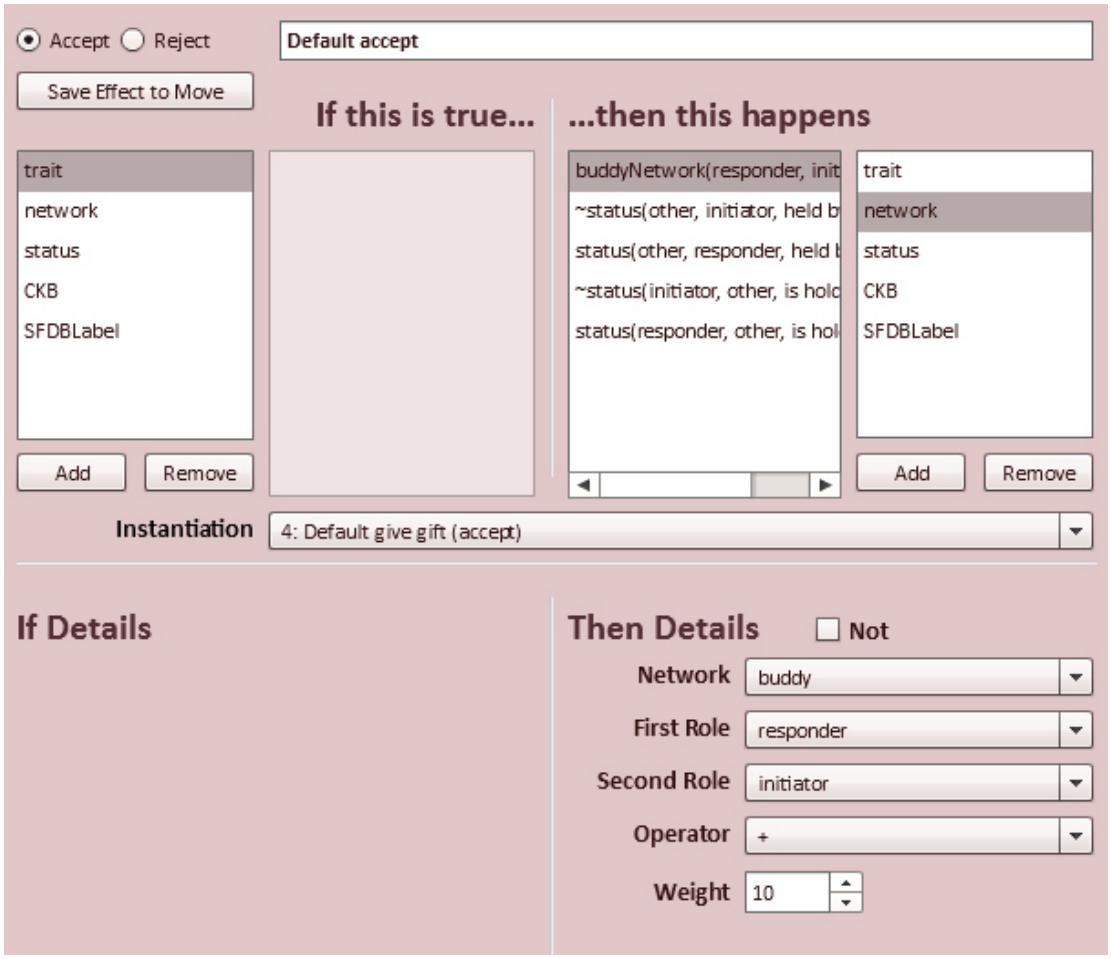

Figure 35: The effect editor in SMDT allows the designer to create the outcome of a social move.

Effects allows the author to list the possible outcomes of a social move (shown in Figure 31 and Figure 35). The designer first designates whether this effect is associated with either an accept or a reject. The section is split into two sections, one that describes the pre-conditions of an effect, the other that describes the outcome. Each sub-section uses predicates to describe both the pre-conditions (evaluation) and the outcome (valuation). Additionally, the designer associates a dialogue exchange to go with the effect.

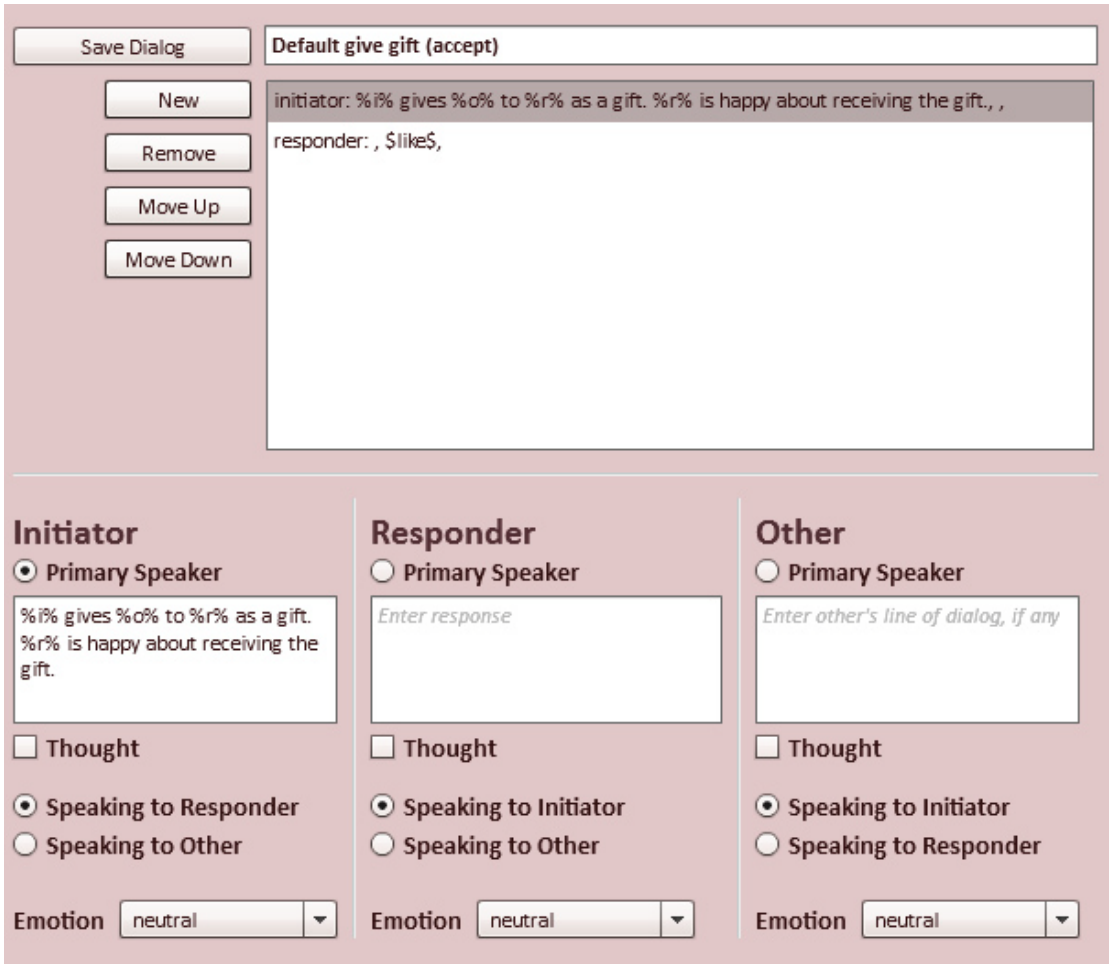

Figure 36: The dialogue editor in SMDT allows the designer to create specific dialogue exchanges associated with a social move. The dialogue input includes support for mark-up language.

Finally, the *dialogue* section (seen in Figure 36) is where the designer writes out the dialogue exchanges that are possible within the social move. They are able to use a mark-up language to designate things to be filled in. For instance, the designer is able to use things such as “Perhaps you should try a new %gender(i, pair of pants, skirt)% to accomplish that look.” The text encased in percent signs means that the gender of the initiator (designated by “i" in the first argument) will use the second argument “pair of pants” if they are a male,

or “skirt” if they are a female. The markup language gives the author the ability to write dynamic dialogue that can be spoken by more than one character. Finally, any text that is enclosed in dollar signs is used by GrailGM for plot-point mix-in. For instance, a dialogue line of “$love$” can be replaced by plot point dialogue that has been associated with the love mood. This is described in more detail in the following section.

3.1.3 Plot Point Creation

Plot Points are created in the Plot Points subsection of the SMDT. The plot points are created by designating the overview data about the plot point, the dialogue associated with the plot point, as well as tagged information.

The overview (seen in Figure 37) is where the designer can designate other plot points that are preconditions on this plot point (enforcing the story DAG) as well as enumerate the traits associated with the plot point. For instance, the plot point in which the player finds out that Violet is a member of the cult has a precondition of the player finding out that Violet has scars on her wrist. In addition, this plot point has the traits: knowledge, factual, Violet’s line, cult line, secret, and plot point. The knowledge and plot point traits are required, as this designates that this piece of knowledge is a plot point and that this is a knowledge type game object. Violet’s line and cult line are used to track which lines the player has followed, and is used in GrailGM for dynamic plot point selection. Factual and secret are used within social moves, to denote a type of plot point that might be revealed or talked about.

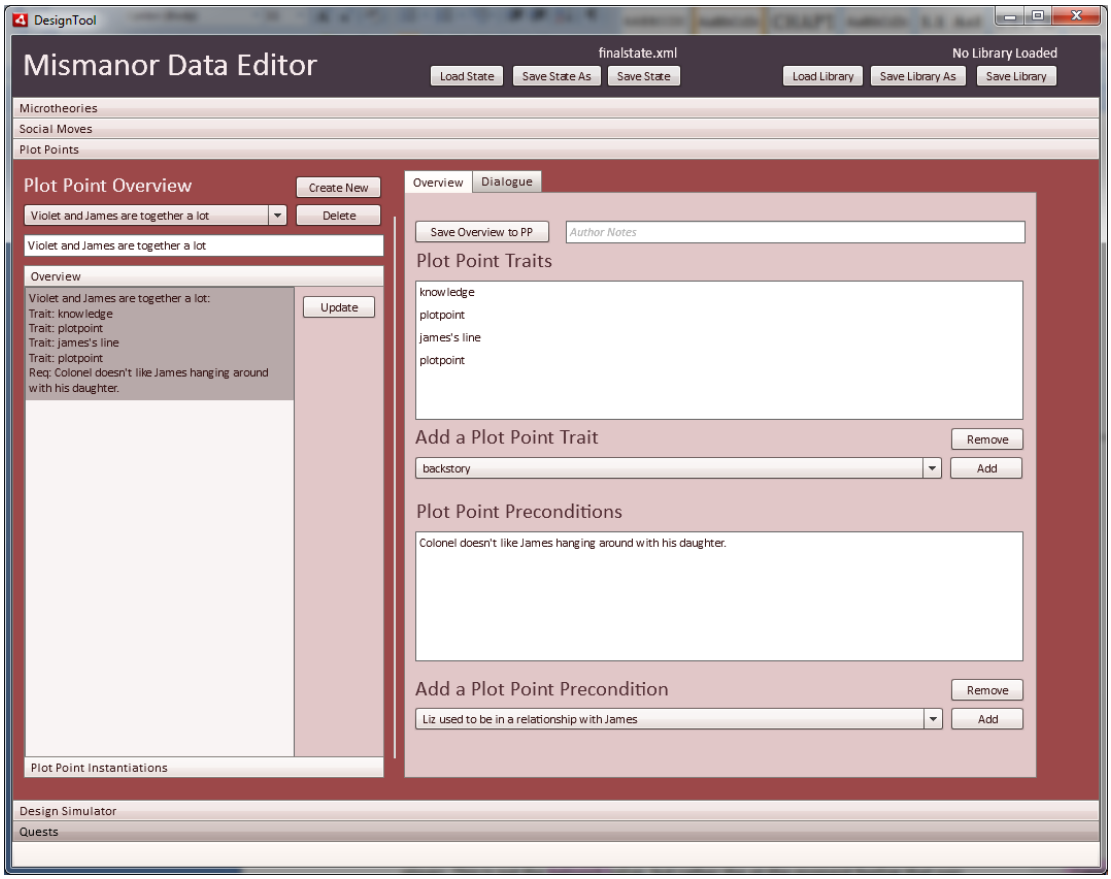

Figure 37: The plot point creation portion of the SFDB.

The dialogue interface allows the designer to designate a speaker, the dialogue exchange, and the mood associated with it. While creating plot points, the designers felt that the mood was more appropriate to denote the emotional state between the speaker and the player. This is not the network value, but rather the at-the-moment feeling that was captured by the social move dialogue. The chosen key words to encode general moods are *hate*, *neutral*, *like*, and *love*. This captures the at-the-moment feeling of the dialogue of the social moves as opposed to the more stable networks and statuses shared by the characters. For instance, while a character may be in love with the player, that does not preclude them

from being able to have an angry dialogue exchange, in which case the “hate” mood would be far more appropriate than the “love” mood.

Finally, the designer can use the *tags* section of the plot point creation tool to designate any characters or locations associated with the plot point. This is used by GrailGM when choosing a new quest or plot point, to maximize cohesion between the quests and plot points.

3.1.4 Quest Creation

The final section of the SMDT is the quest creation section. This section is made up of Intent, Pre-conditions, Characters, Goals, Completion States, Instantiations, and Tags.

Intent is very similar to the social moves editor portion of the SMDT (see Figure 38). Here the designer designates what the intent of the quest is from the point of view of the quest giver. For instance, a quest from the Colonel to convince Violet and James to break up has the intent of lose status *is dating*. The type of quest is extrapolated from the intent; in this example, the quest type is STATUS_LOSE.

Pre-conditions are also similar to the social moves editor portion. Here the designer can specify predicates that make up the rules for the pre-conditions on the quest being available. To maximize the dynamic nature of the quest selection process, pre-conditions are kept minimal.

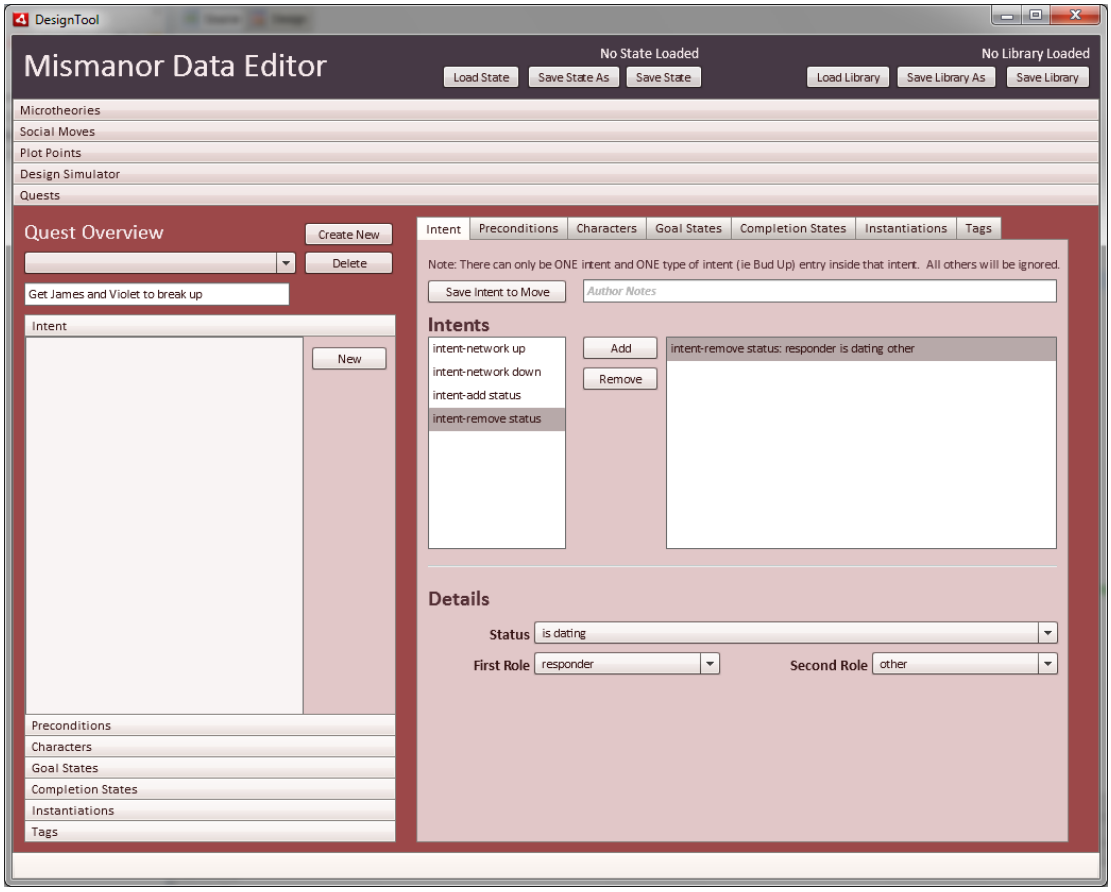

Figure 38: Creating an intent for a new quest in the SMDT.

The *Characters* section is where the designer specifies a quest giver and a quest completer. The designer can be specific and name a particular character, or define a set of predicates that describe a possible quest giver. For instance, the author might specify that a quest can be given by anyone who has the *honest* trait. If nothing is specified, any character may give the quest (except the player). Additionally, the designer may use the same specification methods to designate a quest completer (the character that the player returns to for quest completion). If the designer does not specify anything, the default quest completer is the same as the quest giver.

The *goals* section is where the designer is able to specify the possible goal states that can be given to the player by the quest giver. For instance, the quest to break up Violet and James has three possible goal states: break up Violet and James, break up Violet and James and date Violet, break up Violet and James and date James. The goal is defined by predicates which describe the goal state. The goal may also have pre-conditions that are in addition to the quest pre-conditions. There is always one default goal without any pre-conditions. These are also specified by predicates. Finally, a dialogue exchange is specified by the designer which is the exchange that will be used when the player receives the quest. This is chosen from the list of instantiations the designer created in the instantiation subsection.

The *completion states* section is used by the designer to create possible outcomes of the quest. The completion state is described as a set of predicates. Then the designer specifies an effect and dialogue exchange that is paired with each goal state. The effect is listed as a set of predicates for valuation, while the dialogue exchange is chosen from a list of instantiations created in the instantiation subsection. If no instantiation is chosen, the instantiation that has been marked as the default will be used. There are two default instantiations, one for when the goal and completion state match, and one for when they do not.

The *instantiation* section is where the designer may create the dialogue exchanges. The dialogue may use the same mark-up language as the social moves to allow for multiple characters to play the parts within the dialogue. Additionally, dialogue exchanges may be marked as the “default desired completion” and “default undesired completion.” These are

used when instantiations are not specified, as mentioned above. This allows a designer to add new completion and goal states without having to create new dialogue.

Finally, the last section, *Tags*, is used by the designer to demarcate any character or location mentioned within the quest. The tags are used by the GrailGM system to maximize cohesion between active quests and plot points.

4 Case Study of the Social Mechanics Design Tool

The Social Mechanics Design Tool (SMDT) was used to create the content for *Mismanor*. We had five undergraduate writers use the tool over the course of 6 months. All of the users were comfortable with using computers, although their technical expertise varied. Three of the users had a computer science background, while the other two were primarily creative writers. The main use of the social mechanics design tool (SMDT) was to create concrete domain-specific information for the game world. Using the SMDT was definitely a vast improvement over authors hand-coding XML, and some progress was noted over previous iterations of the design tool. Overall the tool was successful, as we describe below, and we have also found some areas in which to strengthen the tool for future use.

4.1 Use of the Social Mechanics Design Tool

With the use of the Social Mechanics Design Tool, undergraduate authors with no quest authoring or predicate logic experience were able to successfully create a large amount of content for *Mismanor*. With the tool, the authors created over 20,000 lines of XML which would have been incredibly difficult to write by hand, if plausible at all.

4.1.1 Accuracy and Completeness

One of the most obvious benefits of using a tool instead of hand coding is that the tool could guarantee accurate and complete XML output. By this we mean that not only was the XML properly formatted, but each object created with the tool had all the necessary elements. Each section of the SMDT was broken into tabs so the designer knew exactly what needed to be created.

We also included some error checking that would check that all the information was filled out. If something was incomplete, a popup window would appear when the designer saved the file. This could easily be extended to include more checking for common issues in the design process.

4.1.2 Tool Design

Breaking the design tool into task-based sections seemed to work well for most of the designers. This let them easily find where they needed to go in the tool based on the tasks they had been assigned. We found that our writers would gain affinity for a particular task, and in turn gain mastery over that section of the design tool. For instance, we had two authors who became experts at creating social moves, and a third author who became very good at creating micro-theories. Because of the re-use of components across design tool sections, the designers could still quickly learn sections of the tool outside of their expertise, but the task-based nature fit well with the way the students used the tool.

While working with the SMDT, the designers needed to gain some knowledge of predicate logic. Because the tool broke down the types and parts of the predicate, and gave the designer a list to choose from instead of requiring them to type in their own predicates, it

led to much quicker mastery. The designers were able to start working with predicate logic on day one after a short tutorial on how predicates worked within the design tool.

With that said, even with the improved UI, the authors with computer programming experience were able to master the use of predicate logic faster than those without programming experience. However, by the end of the first month, all the students were fairly comfortable using predicates within the tool, and able to create complex rules. For instance, within the *jealous* micro-theory, there is an influence rule made up of a series of predicates which describe that a jealous character is more likely to initiate a social move with the intent to lower friendship with someone if that someone is dating another character that the first character has romantic feelings towards. This is one example of many in which the designers were able to use predicate logic to describe fairly complex social situations.

4.2 Improvements

While the project was overall a success, we did identify some areas that would benefit from some iteration and improvement. Based on the experiences of the writers using the tool, we have created a list of improvements, detailed below.

4.2.1 Designer-Friendly Re-tooling

Currently in the SMDT, the creation of information is broken down based on task. We found in practice, however, that designers were confused by using the same tool to create information for all four levels of story. By making each level of story have its own tool, or a

more visually separate section of the tool, this could alleviate the confusion that the designers felt.

At the world level, designers were creating information for the SFDB and CKB in the tool. Predicate logic was used to designate events that had happened in the past, as well as cultural norms that described the cultural setting of the world.

At the game level, the designers were creating plot points within the design tool. This involved the use of predicate logic to designate when plot points should be available, as well as which characters were best suited for revealing the plot point.

At the quest level, authors created quests within the design tool as well. Again, predicate logic was used to dictate when a quest was available and which characters could give the quest. However, predicate logic was also used to specify multiple entry points to the quest (and the timing and suitability of each one) as well as the possible outcomes and effects of each quest.

Finally, at the player level, micro-theories and social moves were also authored in the same design tool. Once again, predicate logic was used to specify when a character was more or less likely to choose a social move as well as accept or reject that move. In addition, predicates describe the effects of each move.

Unfortunately, SMDT did not present the predicate logic in the most designer friendly manner. The predicates were used as pre-conditions, influence rule weights, and effects, but presented with very little variation. For instance, consider the following sequence:

buddyNetwork(initiator, responder) greaterThan 50 (precondition)

3 buddyNetwork(initiator, responder) greaterThan 50 (influence rule weight)

buddyNetwork(initiator,responder) greaterThan 50 (condition of an effect)

buddyNetwork(initiator, responder) +10 (change of the effect)

The first describes a precondition for a social move, in which the social move is only available if the buddy network between the initiator of the move towards the responder of the move is higher than 50.

The second is used in an influence rule for a social move, or for a micro-theory. In this case, if the initiator has a buddyNetwork over 50 with the responder, that rule adds +3 to the total score for the intent of the micro-theory or towards initiating the social move.

The third and fourth lines are both sides of an effect, the condition and change. The third line tests if the buddyNetwork between the initiator towards the responder is greater than 50, and the fourth line causes the buddyNetwork to increase by 10 if that effect is matched.

Often the designers would get confused as to which way the predicates were being used, and this led to errors within the authoring such as unnecessary or false pre-conditions, or logical errors. By making it more obvious to the designer what role the predicate is currently playing, it would alleviate many of these issues. This could be done with color-coding, or human-friendly comments.

Not only was the proper use of different predicates sometimes confusing, but the display of the predicates was also not necessarily intuitive. With the use of simple parsing, this could be changed to be shown in a more readable manner. For instance, to specify that the initiator of a social action *Ask Out* needs to not be dating the responder, the SMDT would display this as:

Precondition: ~status(initiator, responder, is dating, -1)

The -1 is the duration of the status, and is only used during valuation. That it is being shown at all is only part of the problem. Even the more technical users of the tool would have errors in their content because it was easy to miss the ~ or by reversing the initiator and responder among other issues. It would be a nice improvement if the design tool displayed this in human-friendly terms such as "Ask Out requires that the *initiator* does not have the status *is dating* towards the *responder*" or even "Ask Out requires that the initiator is not dating the responder". This would alleviate some of the errors that were caused by the designer not being able to quickly understand what they were specifying.

4.2.2 Robust Editing Support

Creating content requires the creation of a lot of information, and at times the designers are tasked with re-entering the same information multiple times. For instance, often the writers would want to reuse initiator influence rules for responder rule sets as well. This is alleviated to some degree with micro-theories; the designers are able to designate behaviors associated with specific traits, statuses, and relationship levels without having to add it to every social action, quest or plot point.

However, designers are still required to enter the same information a number of times in different places. Because the GUI involves drop downs and selection boxes instead of text entry (to minimize game-breaking typographical errors), entering information requires 3-5 mouse clicks. The easiest way to help with this is to allow copy and pasting of specified predicates.

A more robust way to expand SMDT beyond copy & paste would be to allow the designer to choose an active trait, status or network setting. Then they could scroll through the micro-theories, social moves, plot points and quests, and add the active element to a precondition, influence rule set, or effect. This would have the added benefit of supporting different methods of designing. Instead of designing everything for one micro-theory, social action, quest, or plot point at a time, a designer could think about each trait, status, or relationship individually, and decide how that concept would affect the various story elements.

Adding this functionality would add cohesiveness through the story design, as well. For instance, it was fairly common that different authors would have specific traits and statuses that they often favored for designing, and would be well specified throughout their story elements, while others were unaccounted for. Micro-theories, again, alleviated this to some degree, but there was still an issue of heavier weightings due to uneven trait and status design.

4.2.3 In-Tool Testing

The biggest area for improvement in the SMDT is allowing for testing within the tool itself, by including a test harness for the designers. Given the dynamic nature of the game world, it

was difficult for the designers to test the content they added within the game itself. There are a number of levels in which in-tool testing could be added.

At the player story level, the micro-theories work with the social move influence rules to weight whether a character is likely to initiate that social action, or to accept or reject it. Because there may be hundreds of rules pertaining to a specific situation and weighing an option, it can be very difficult to assign weights to the rules accurately. For instance, the following three influence rules:

+1 – romanceNetwork(initiator, responder, greaterThan, 25)

+2 – romanceNetwork(initiator, responder, greaterThan, 50)

+3 – romanceNetwork(initiator, responder, greaterThan, 75)

are commonly found within influence rules. They imply that the stronger the initiator's romantic feelings for the responder, the more likely they are to play that particular social move. While these rules do in fact do this, it is actually weighing the rules much more strongly than the designers intended. If the initiator has a romance of 76 towards the responder, it will trigger all three rules, giving a weighing of 6 towards that social move, instead of the 3 that was likely intended. It is more accurate to give each rule a +1 weighting instead.

Being able to simulate the game world by specifying a type of player as well as a social context, and showing the weights of the influence rules for the available social actions would allow the designers to more easily fine tune the weights of each influence rule. Additionally, it would help them more easily identify outliers which could signify bugs or

repeated predicates within the influence rules. For instance, if one social action has a total score that is much higher than the next scored social action, it is likely that tuning or bug-fixing is required.

Finally, for the player level story creation, it would be very useful for the designer to be able to simulate the various effects of each social move. An effect is made up of a pre-condition (with default accept and reject effects with no pre-conditions) and the effect of that social action being accepted or rejected, as well as the dialogue exchange that is spoken by the initiator and responder. The written dialogue uses a mark-up language to allow for dynamic text replacement within the dialogue exchanges. This is three sets of variables that all need to be testable, and it is difficult to be able to easily simulate all these conditions within the game.

As previously stated, making the predicates more easily readable by humans would alleviate some of the common bugs in authoring. However, being able to both set up a social context and test which effect would be triggered as well as iterating through all of the dialogue given randomly chosen characters would also be helpful. This would find logic issues within the rules, as well as mark-up errors or content issues within the dialogue. For instance, early in the design process, authors would commonly write dialogue that only made sense if spoken by one particular character. By showing the dialogue being spoken by two random characters, it would highlight this issue.

At the quest level, the designer specifies the quest information within the tool. Similar to player level, being able to test within the tool would give the author the ability to test some

of the more bug-prone areas of quest creation. In our experience, creation of the possible starting states and ending states, and the associated dialogue were the most common places for bugs. Because the dialogue is different based on the pairing of the starting and ending state, it was common for the dialogue to be mis-associated. Making this more obvious in the user interface would help alleviate this, but being able to test the setup based on social context would bring to light any predicate-level errors as well as dialogue problems.

Finally, at the game level, plot points are created as a dialogue exchange associated with a mood (so they can be appropriately mixed-in with social moves) and a speaker. For this level of story, it would be very helpful for the designer to be able to test the mix-ins, and make sure the mood was appropriately set. For instance, having a dialogue exchange that sounds angry, but a mood set to *love* would lead to a plot-point mix-in of a very loving social move exchange, followed by an angry sounding plot point dialogue. Something so abruptly incorrect was not very common; more common was an exchange that didn't fit in a more subtle way. By allowing easy testing, this could be fixed by modifying the dialogue of the plot point, the mood of the plot point, or even changing the way the mix-in was handled at the social move level.

At all levels, a huge benefit could be gained by having a way to indicate overly constraining pre-conditions, such that they make that particular story element impossible to reach. This could be done at a logical level—the pre-condition always evaluates to false, such as romanceNetwork(initiator, responder, lessThan, 20) && romanceNetwork(initiator,

responder, greaterThan, 25)—and also using an AI agent which attempts to reach all possibilities.

5 Conclusion

Creating information for such a dynamic world requires specialized tools, as well as a large amount of content to take advantage of the dynamic capabilities. As such, the design tools are an important, and often overlooked, part of the process.

In order for the designer to realistically be able to create the number and variety of solutions required for playable quests, we created the Grail Framework Brainstormer tool to aid the designer in quest creation. Brainstormer is built upon a common-sense knowledgebase created by thousands of people, therefore the knowledgebase does not have the biases or preferences that just one person might have. Consequently, using this system allows the designer to see past their own quest preferences and routines, finding new and innovative solutions for each quest.

The Social Mechanics Design Tool was created to help designers create game-specific content for the social RPG, *Mismanor*. The tool gives the designers the ability to create social moves, quests, plot points, and micro-theories. We used the tool over the course of six months to create the content for *Mismanor* and overall found it to be a successful tool.

CHAPTER 7

EMPATH

When addressing game level story concerns, we first looked at drama management systems. In particular, declarative optimized drama management systems are an interesting option for improving agency and maximizing the meaningfulness of player choice within a game world. To test our hypothesis, we created *EMPath* to test the effectiveness of DODM on allowing for interesting and meaningful choices in the game level story.

Drama management was first proposed in the context of interactive drama (ID), dramatic worlds in which players experience a dynamic story arc and interact with socially capable, personality-rich autonomous characters. In addition to drama management, the creation of such experiences introduces many design and technology challenges such as autonomous characters, dialog management, and natural language understanding and generation.

Because of the complexity related to creating complete, playable IDs, much of the drama management literature has focused on technical proofs of concept that are not integrated into a playable game, making it difficult to assess whether the DM is positively affecting player experiences.

However, drama management can be applied more generally than to IDs; it can be applied to any interactive experience in which the author would like to express preferences for action or activity sequences with specific properties while still allowing player agency and non-linear events. Within existing game genres, drama management can perform tasks such

as sequencing nonlinear missions, encounters, and the order in which players discover information in the game.

In particular, drama management can support a level of non-linearity in existing game genres that would be difficult to impossible to achieve using the traditional approach of scripting and local triggers. Additionally, drama management opens up new game design possibilities that would traditionally be complicated or impossible to conceive of without the capabilities of a Drama Manager (DM). We created *EMPath* to explore the design and technology issues of integrating drama management into a complete, playable experience.

EMPath (Experience Managed Path) is a classic *Zelda*-like adventure game. *EMPath* was designed to be small, so that it can be completed quickly and players can experience a clear sense of progression and completion from beginning to end. It also was designed to be playable without the DM, yet amenable to drama management.

1 Related Work

Search-based drama management (SBDM) was first proposed by Bates (Bates, 1992) and developed by Weyhrauch (Weyhrauch, 1997). Weyhrauch applied SBDM to *Tea for Three*, a simplified version of the Infocom interactive fiction *Deadline*; achieving impressive results in an abstract story space with a simulated user. Lamstein & Mateas (Lamstein & Mateas, 2004) proposed reviving this work.

More recent work has generalized SBDM as DODM, for which search is one optimization method. Other optimization methods have been tried, including reinforcement learning, and an alternative formulation of the DM problem as targeted trajectory distribution

Markov decision processes (TTD-MDPs) (Roberts, Nelson, Isbell, Mateas, & Littman, 2006). Nelson and Mateas (Nelson & Mateas, 2005) have compared the search and TTD-MDP formulations of DODM.

The Mimesis architecture (R. M. Young & Riedl, 2003) constructs story plans and monitors for player actions that might threaten causal links in the story plan. If a threat is a detected, the architecture re-plans or prevents player action. The Interactive Drama Architecture (Magerko, 2005) strives to keep the user on a prewritten storyline by taking corrective action guided by a state-machine model of likely player behavior. In contrast to both approaches, DODM seeks to incorporate player action into a dynamic plot, rather than guide player action onto a pre-written plot.

Content selection models of drama management have proven popular, being used in Façade (Mateas & Stern, 2003), the PaSSAGE system (Thue, Bulitko, Spetch, & Wasylishen, 2007), GADIN (Barber & Kudenko, 2007) and OPIATE (Fairclough, 2004). In Façade, the beat-based drama manager maintains a probabilistic agenda of dramatic beats. Dramatic beats are the smallest unit of change of dramatic value, where dramatic values could be love, trust or tension. Each beat coordinates autonomous characters in carrying out a unit of dramatic action, while supporting player interaction during the beat. Similarly, PaSSAGE contains a library of possible character encounters in a role-playing game, dynamically selecting the next encounter using a function based on the player model.

GADIN creates stories by asking the player how to resolve various dilemmas. The dilemmas are chosen based on how “interesting” the dilemma will be for the player. Like PaSSAGE, the

"interestingness" is rated using a player model. OPIATE extracts subplots using a case-based reasoner to create Proppian fairy tales, re-planning and finding new suitable cases as a response to player interaction. OPIATE is able to pursue goals that may require multiple story events to occur. This leads it to project the future story further than the other systems presented here, which only make local decisions. However, DODM is the only system that makes a decision by projecting future story arcs to completion.

Thespian (Si, Marsella, & Pynadath, 2005) and U-Director (Mott & Lester, 2006) both employ a decision theoretic approach to DM. In Thespian, decision-theoretic goal driven agents select actions to maximize goals specified as reward functions over state. DM is thus decentralized across characters. U-Director employs a centralized DM that selects actions to maximize narrative utility, measured with respect to narrative objectives, story world, and player state. U-Director shares with DODM the ability to trade off among multiple, potentially contradictory narrative features.

DODM has traditionally been evaluated in discrete textual spaces. In contrast, we explored applying DODM to a graphical game world. Ontañón, et al. (Ontañón, Jain, Mehta, & Ram, 2008) take a similar approach, creating a graphical representation of *Anchorhead*, but maintaining the discrete game world and discrete time used in the interactive fiction. In this system, the player interacted with the story using typed commands which were interpreted with an NLU module. The number of actions the player could perform was greatly increased by this, and their system uses a hybrid montecarlo expectimax algorithm to focus the DM searches. This reduces the amount of game tree searches the DM performs, allowing it to respond much faster.

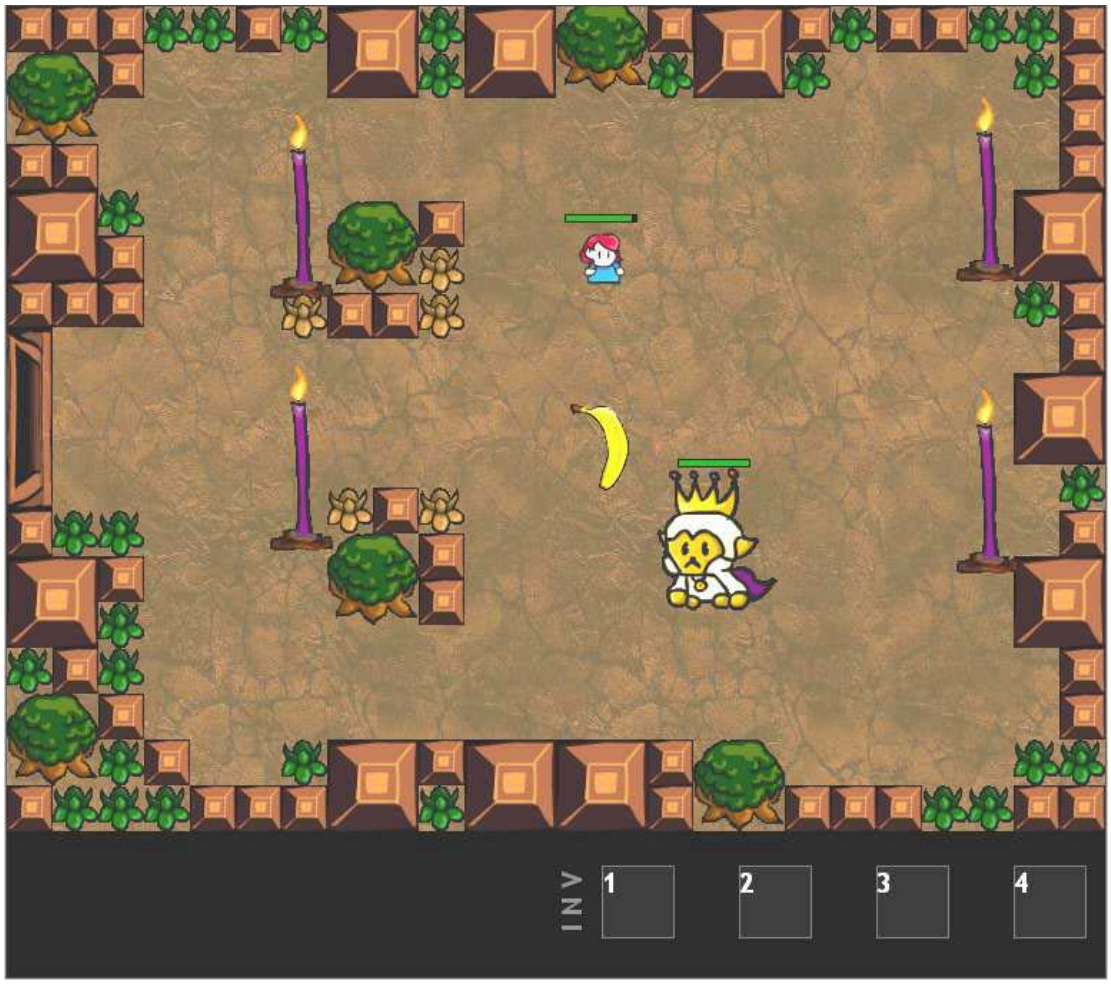

Figure 39: A screenshot of EMPath, showing the player encountering the Monkey King boss.

EMPath is real-time, unlike the turn-based system used by *Anchorhead*, therefore requiring more complicated applications of the abstract drama manager actions to handle the real-time nature of the game.

2 Game Description

The *EMPath* world is a 25 room top-down dungeon populated with enemies, fire traps, a prisoner, and two bosses. The player's goal is to reach the stairs and escape from the

dungeon. However, the stairs are too dark to traverse and blocked by rubble; the player must find the special items required to escape.

There are two quest lines in the game. The first involves finding a candle so that the player can see their way up the stairs. There are notes which contain clues that point to the location and use of the candle. The candle is in the possession of the monkey king, whom the player needs to kill to acquire the candle.

The second quest line revolves around a prisoner, who possesses a magic talisman that can clear the rubble blocking the stairs. A note in the game world hints to the player of the existence of a prisoner imprisoned in a jail cell in the dungeon. When the player finds the prisoner, they are told that the large guard next door (boss) has the key. They player must kill the guard, who will drop the key to the cell. When the player frees the prisoner, they receive the magic talisman.

The game is written such that the drama manager can get the candle and key to the player in other ways, but described above is the default path through the game, and the one that is forced if the drama manager is not enabled.

3 The DODM Framework

To configure DODM for a specific world, the author specifies plot points, DM actions, and an evaluation function. Plot points are important events that can occur in an experience. Different sequences of plot points define different player trajectories through games or story worlds. Examples of plot points include a player gaining story information or acquiring an important object. The plot points are annotated with ordering constraints that capture

the physical limitations of the world, such as events in a locked room not being possible until the player gets the key. Plot points are also annotated with information such as where the plot point happens or what subplot it is part of. The exact set of annotations is flexible and depends on the story.

DM actions are actions the DM can take to intervene in the unfolding experience. Actions can cause specific plot points to happen, provide *hints* that make it more likely a plot point will happen, *deny* a plot point so it cannot happen, or *un-deny* a previously denied plot point.

The evaluation function, given a total sequence of plot points that occurred in the world, returns a "goodness" evaluation for that sequence. This evaluation is a specific, author-specified function that captures story or experience goodness for a specific world. While an author can create custom story features, the DODM framework provides a set of features that are commonly useful in defining evaluation functions.

We used the following four features for the EMPath story evaluation function: *Motivation*, *Thought Flow*, *Manipulation*, and *Story Density*. *Motivation* measures whether events that happen in the world are motivated by previous events. For example, in EMPath, the player can learn from a note that they must acquire a candle before leaving the dungeon; if they encounter this note before finding a boss monster in a room with candles, then the encounter with the boss is motivated.

Similarly, *Thought Flow* measures the degree to which plot points associated with the same topic appear together. In EMPath, this feature prefers stories in which plot points associated with each of the quests are grouped together without interleaving.

Story Density is a measure of how tightly grouped activated plot points occur. In EMPath it is desirable for plot points to happen at a steady pace – not all at once, nor largely spread apart.

Manipulation measures how coercive the DM is in a given story sequence. Manipulation costs are associated with each DM action rather than being associated with plot points. Manipulation counter-balances other features, forcing the DM to take into account manipulation costs when optimizing other features. The author associates plot points or DM actions with specific evaluation features.

3.1.1 Plot Points

In order to create the drama managed version of the game, it is necessary to decouple significant game events at the game story level from their default realization in the non drama-managed game. This is accomplished by defining plot points. As in standard adventure and RPG game design, some of the plot-point sequences are linearized via trigger logic, while others can appear in different orders, including orders that may not be as satisfying. An example of a less satisfying plot point ordering is reading a clue in a note after the player has already accomplished the task. This is a not uncommon experience in contemporary games, which rely on flags and triggers.

Plot Point	Description
info_loc_candle THOUGHT = candle SETS_UP = get_candle	player receives information about the location of the candle
info_use_candle THOUGHT = candle SETS_UP = get_candle	player receives information about the use of the candle
king_dead THOUGHT = candle MOTIVATED_BY = info_loc_candle MOTIVATED_BY = info_use_candle	player kills the monkey king
get_candle THOUGHT = candle MOTIVATED_BY = king_dead	player gets the candle
info_jail THOUGHT = prisoner SETS_UP = get_key	player receives information about a prisoner in jail
info_key_guard THOUGHT = prisoner SETS_UP = get_key	player receives information that the guard holds the key
get_talisman THOUGHT = prisoner PREREQ = get_key	player receives the talisman
get_key THOUGHT = prisoner SETS_UP = get_talisman MOTIVATED_BY = guard_dead	player gets the key to unlock the jail
guard_dead THOUGHT = prisoner MOTIVATED_BY = info_jail MOTIVATED_BY = info_key_guard	player kills the guard
gets_to_stairs MOTIVATED_BY = info_use_wax PREREQ = ^ get_wax give_flute	Player reaches the stairs and ends the game

Figure 40: The 10 plot points in EMPath.

Plot points are the significant events that influence the player's overall experience in the game level story. We defined ten plot points for *EMPath*; they appear in Figure 40. Each plot point is annotated to support the evaluation features, in this case thought flow and motivation. Note that plot points are defined so as to generalize them from their default implementations. For example, instead of a plot point "find note about candle location", there is a plot point "learn candle location" as there are multiple ways to learn about the candle location.

Plot points are organized into a directed acyclic graph (DAG) to capture ordering dependencies that are enforced by the physical logic of the world. In *EMPath*, we had very few dependencies; **get_key** is required before **get_talisman** may happen and **get_candle** and **get_talisman** are prerequisites for exiting the dungeon. Most plot points can appear in any order; however the evaluation function encodes a preference for certain orderings.

3.1.2 DM Actions

For the Drama Manager to be able to influence the game world, it was necessary to create a pool of actions that the DM can perform. These are created by considering each plot point, and designing a set of possible DM actions at the hint, causer, and denier level. For *EMPath*, we created a total of 33 DM actions to help influence the ten possible plot points. Eleven example actions appear in Figure 41.

Designing a set of hints, causers, and deniers for each plot point is similar to a game master in a table-top RPG thinking of ways to adapt the story around the player's actions. At the hint level, the DM actions should raise the probability of a particular plot point happening.

DM Action	Description
note_loc_candle_drop_off_mob HINT = info_loc_candle, STRENGTH = 1.5 MANIPULATIVITY = 0.1	Info_loc_candle note drops off next enemy killed
note_loc_candle_next_room HINT = info_loc_candle, STRENGTH = 5 MANIPULATIVITY = 0.1	Info_loc_candle note shows up in next room
npc_talk_loc_candle CAUSES = info_loc_candle MANIPULATIVITY = 0.6	NPCs walk in chatting about info_loc_candle
reenable_info_loc_candle REENABLES = info_loc_candle MANIPULATIVITY = 0.1	Adds info_loc_candle note back into the game, and re-enables other hints and causers
temp_deny_info_loc_candle TEMP_DENIES = info_loc_candle MANIPULATIVITY = 0.1	Temporarily remove info_loc_candle note from game (other hints and causers disabled)
note_use_candle_drop_off_mob HINT = info_use_candle, STRENGTH = 1.5 MANIPULATIVITY = 0.1	Info_use_candle note drops off next enemy killed
note_use_candle_next_room HINT = info_use_candle, STRENGTH = 5 MANIPULATIVITY = 0.1	Info_use_candle note shows up in next room
npc_talk_use_candle CAUSES = info_use_candle MANIPULATIVITY = 0.6	NPCs walk into room chatting about info_use_candle
reenable_info_use_candle REENABLES = info_use_candle MANIPULATIVITY = 0.1	Adds info_use_candle note back into the game, and re-enables other hints and causers)
temp_deny_info_use_candle TEMP_DENIES = info_use_candle MANIPULATIVITY = 0.1	Temporarily removes info_use_candle note from game (other hints and causers disabled)
boss_attack_player HINT = boss_dead, STRENGTH = 10.0 MANIPULATIVITY = 0.5	Monkey king chases down the player, following them room to room.

Figure 41: Eleven of the 33 possible actions that the Drama Manager can take.

For instance, to raise the probability of the player learning the location of the candle, the DM could have a note containing the information drop as loot from the next enemy the player kills, or have it show up in the next unexplored room the player enters. These are merely hints because the player may not pick up the note, may not kill anything, or may not explore any more rooms. The hints have different levels of strength, depending on how much more likely they are to make the plot point happen.

A causer, on the other hand, makes the plot point happen. For finding the location of the candle, the DM can cause an event such that NPCs enter the room, and the player overhears the information about the candle location. This has a higher manipulation cost, but has a 100% chance of making the plot point happen. Manipulation costs are explained in more detail below.

Finally, a denier makes the plot point impossible to happen. For the most part, deniers remove things from the game, so the player cannot find them. In the case of boss monsters or significant NPCs, for a dungeon-crawler adventure-type game, we chose to use locked doors to keep the player from reaching the boss, and had the wizard NPC be asleep if the DM was denying the player the ability to get information from him.

For each DM action, we also specified a manipulation cost, which is used by the manipulation evaluation feature. More information about manipulation can be found in the following section.

When the DODM is connected to a concrete game world, the world informs the DM when the player has caused a plot point to happen, and which plot point occurred. The DM then

decides whether to take any action, and tells the world to carry out that action. In *EMPath*, for example, the player may be moving between rooms and fighting non-boss monsters. When the player finds a note describing a nearby prisoner, the game tells the DM that a plot point has happened. The DM might then choose the "deny candle location information" action, telling the game world to remove a specific note. Actions are chosen based on which action will lead to the story path with the highest-rated trajectory based on the evaluation features, which are described below.

3.1.3 Evaluation Features

We used the following four features for the *EMPath* story evaluation function: *Motivation*, *Thought Flow*, *Manipulation*, and *Story Density*. These features each measure an author-level consideration on the quest and game level story.

The *Motivation* feature measures whether events that happen in the world are motivated by previous events. Without motivation, plot points feel random because they have no justification for happening. This is a strong factor towards cohesion within the story. An example of a motivated event within *EMPath* is that the player can first learn about the need for the candle (to light the exit) as well as the location of the candle (with the monkey king boss enemy). By learning this information before the encounter of the boss, this makes the encounter a motivated event. If the player finds the boss enemy first, then finds a note about the location of the candle, not only is the boss encounter unmotivated, but the knowledge of the candle location is already known. At best the note will be seen as meaningless, at worst it can feel like a bug within the code, as it is a common bug among flag-based systems.

Thought Flow measures the degree to which plot points associated with the same topic appear together. Thought Flow is helpful in keeping localized cohesion between plot points. This feature highlights the author-level subjectivity of the evaluation features. For instance, some designers prefer higher levels of story interleaving, while others may prefer strong topical cohesion. Because *EMPath* was such a small-scale world, we preferred thought flow because there was not enough content to make interleaving feel successful. In particular, *EMPath* used this feature to prefer stories in which plot points associated with each of the quests are grouped together without interleaving.

Story Density is a measure of how tightly grouped activated plot points occur. In *EMPath* it is desirable for plot points to happen at a steady pace – not clumped together nor with long periods of no story in between. This helps alleviate issues in which a player may end up wandering, lost within the game world, or having all the plot points happen immediately upon entering the game. We discuss story density in more detail in Chapter 7, Section 5.1.1.

Manipulation measures how coercive the DM is in a given story sequence. Manipulation costs are associated with each DM action rather than being associated with plot points. Manipulation counter-balances other features, forcing the DM to take into account manipulation costs when optimizing other features. This cost describes how obvious the action may be perceived to be, or how manipulative the DM must be to carry out that particular action. For instance, moving a note to another place in the dungeon is a low manipulation cost, because finding random items in rooms is a fairly common dungeon-game trope. However, having the monkey king boss appear in a room and begin chasing the

player is a high manipulation cost, because it is more obvious that the game is changing around the player.

4 Game Design Challenges

Accomplishing drama managed levels of dynamism with purely local trigger logic used in many contemporary games would be a Herculean task. For each plot point, it would be coded as local tests that somehow looked at the history of what happened so far, and trading off between the various evaluation features. For instance, exactly when the note should be discovered may involve trade-offs between thought flow and motivation.

The difficulty with trigger logic is that it does not gracefully scale to handling a large amount of interacting state in making decisions, and, more critically, *can only consider the past, not the effect of actions on possible futures*. DODM can take into account the interactions between various decisions, and, by projecting possible futures, it is able to weigh the costs and benefits of making various game world interventions.

Using DODM makes a highly dynamic storyline possible, but does require some changes in how the game is designed. One of the biggest challenges in creating a drama managed game is authoring non-linear story lines. For example, with certain types of DM moves that we used, events can happen in many possible orders and in many possible ways, physical objects may move, and connectivity of map areas may change. This requires creating a story that still works with some re-ordering. By breaking the story into two quest lines with no inter-dependency, we were able to allow these to be discovered and completed in any

order. Inside the quests themselves, we specified a preference for the clues to be experienced before the final quest-resolving action, but it was not necessary.

In addition, the game designer must design how the DM actions affect the detailed game state. The process of making a DM action happen within the game world is called the *refinement* of DM actions. The refinements need to take into account the type of DM action (hint, causer, denier, re-enabler), the plot point that the action is associated with, as well as the fact that this action could possibly happen at any time within the storyline.

The game must be designed dynamically enough to support these different refinements. For example, what happens when the player kills the monkey king if the DM has chosen a different method to deliver the candle to the player? As this is clearly a boss battle, having nothing happen would confuse the player, so the designer needs to provide content to handle this situation.

All sequencing events related to plot points – e.g. whether the player received all the information before finding the candle – are left to the DM. However, in the detailed design of the game world, it is necessary to handle *how* the player encounters plot points. The designer must implement the content in the game that handles how these details interact with other events in the game.

Finally, plot points must be carefully chosen so as to provide the DM with a view of all the events important to the flow of the experience. Without knowledge of all the significant events, the DM will not be able to reason about where to fit each into the experience, nor have actions associated with them.

5 DODM Adjustments

Prior DODM research has focused on abstract game models defined purely in terms of plot points and DM actions. It had been previously believed that everything to do with sequence-level or story-level importance in the game world was captured by the DAG. However, when connecting DODM to a concrete game world, we had to modify DODM to account for aspects of the player's experience influenced by the physical layout of the world. To incorporate the new spatial importance of the world, we created a new evaluation feature, story density, as well as a new player model which accounted for world layout,which is described in more detail in Section 6.1.3

5.1.1 Story Density

Before *EMPath*, all the DODM evaluation features that had been developed had no dependencies on physical details of the game world. Consider the three prior evaluation features that we are reusing for *EMPath*: thought flow, motivation and manipulation. Values for these three features are computed based entirely on static annotations on plot points and DM actions. As we defined the evaluation function for *EMPath*, however, we realized that none of the features in our toolbox adequately capture a notion of *story density*.

The *Story Density* feature relies on an explicit representation of the game world in the drama manager. Unlike the other features, which use an abstract world representation based on plot point orderings, story density requires spatial awareness of the game world. In gameplay, the physical movements of the player become important. We believe it is detrimental to the story for the player to be moving through many rooms without a plot point occurring, as this indicates that the player could be lost or wandering aimlessly.

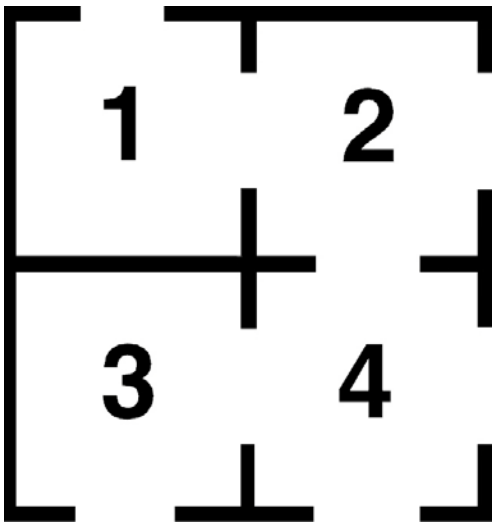

Figure 42: Example layout of rooms within EMPath.

Conversely, if many plot points are occurring in close proximity, the player could be overwhelmed with content. By knowing where plot points can occur in the game world, the drama manager can choose to prefer game experiences that give the story a steady pace. Story Density analyzes the room transitions between plot points and therefore cannot utilize a static annotation on each plot point which is used by the other evaluation features. For Story Density to operate, it needs to predict the distance between plot points to help weight the search through possible future moves.

We calculated the distance between plot points by multiplying the Manhattan distance between rooms by 1.5. The scaling factor is necessary for two reasons. As shown in Figure 42, not all rooms are connected to each other, and thus following the exact Manhattan distance path is not always possible. If the player is in Room 1, the Manhattan distance to Room 3 is 1, but reaching that room actually requires three room transitions. The scaling

factor is also required to take into account that players do not always move the optimal route between plot points.

6 Player Models

The Drama Manager relies heavily on an internal player model to create its predictive game tree. For each plot point that happens within the game, the DM chooses which action to perform (which can also include the null action). The logic for choosing this action is based on creating a look-ahead game tree of possible future plot points and DM responses. Each leaf of the tree is evaluated using the evaluation features specified by the author. The sub-tree that ends with the highest evaluated leaf is chosen as the DM action. Ties are resolved in a non-deterministic fashion.

Future plot point probabilities are provided by the player model, and therefore the player model is the foundation of the Drama Manager's choices. It stands to reason that the more accurate the player model is, the better choices the Drama Manager will be able to make.

6.1.1 Uniform Player Model

This basic player model assumes that each remaining plot point has an equal probability of happening. When hints are active, it adds a weight to the plot point that is being hinted at. This is the most simplistic of the player models, and is currently widely used.

The strength of the uniform player model is that it is the easiest to create and requires the least amount of knowledge of the game world. Because the player model is required to recalculate the probabilities at each node of the game tree, it is called a large number of

times. With this player model, the speed is not an issue, and the Drama Manager can choose its next action in a fraction of a second.

With this simplicity comes a large cost. A uniform player model is not generally an accurate model of what a real player would do in the game world. For instance, the uniform player model does not take the player's position into account, which is unrealistic as it is far more likely that a player will trigger closer plot points than those further away.

By assuming a uniform distribution of probabilities, the Drama Manager may chose an action that will affect a plot point that realistically will not be encountered by the player until much later in the game.

6.1.2 Manhattan Distance Player Model

Due to the limitations of the uniform player model, we first created a player model that utilizes the Manhattan distance to weight the probabilities of each plot point occurring. With this player model, only limited knowledge of the world is known to the drama manager. The coordinates of each plot point is used to calculate the Manhattan distance between plot points. As discussed above in Section 5.1.1, we multiply the Manhattan distance by 1.5 to estimate some allowance for the maze-like nature of the world.

Using this formula created a much more realistic probability mapping of the plot points without adding unnecessary complexity. Since players are much more likely to encounter plot points closer to them, this simplified distance measurement takes this into account. However, there are room configurations where the Manhattan distance is not a good approximation tool. For instance, rooms with locations right next to each other will have a

low Manhattan distance. If these rooms are not actually connected by a doorway, but requires traversing a U-shaped path through a series of rooms, this Manhattan distance is a poor approximation.

In practice, we had a bigger issue with this player model in that the world is small enough that the weighting from the Manhattan distance is not enough to make a discernible difference as seen in Figure 43. Scaling the weights could help account for this, but the inaccuracies of using the Manhattan distance led us towards a different player model.

6.1.3 World Knowledge User Model

Using the Manhattan distance did not give us the spatial accuracy we wanted, so we created a player model which has an internal representation of the world layout, as well as the locations of the plot points. Using this map, it is possible to calculate much more realistic spatial-based probabilities of each plot point happening next.

To calculate the probabilities, we traverse one hundred random walks, noting the first plot point encountered. Afterwards, we tally these numbers and normalize to reach the final probability distribution. When performing a random walk traversal, the player model stops at the location containing the first plot point it encounters 95% of the time, continuing the walk the other 5%. This models the fact that that players will sometimes walk past an interaction associated with a plot point (e.g. ignoring a note, not picking up an important object, etc.), though we observed that this is unlikely.

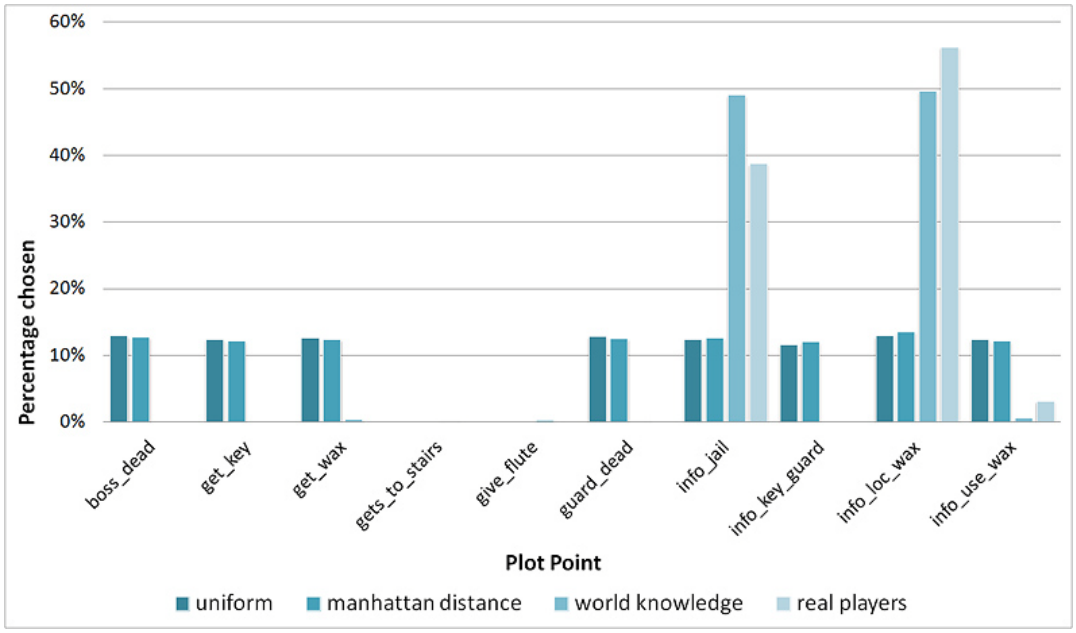

Figure 43: The world knowledge player model most accurately reflected the choices real players made. In this graph, we compare the first plot point encountered by each of the player models, as well as the players themselves. It is clear that the World Knowledge player model had the most accurate results.

We chose to add a small probability of missing a plot point based on the results of our initial player tests. In practice, most users would activate a plot point when it was in the same room as them – by picking up a note or fighting a boss. However, in some cases, players would instead enter a room, then either ignore the plot point device, or back out of the room, because they did not want to activate a particular plot point yet.

The tradeoff of this player model is the added time and complexity of creating the game tree. For every node in the game tree, one hundred random walks are performed which can add up to a large overhead. By optimizing the random walk, we were able to reduce the

decision time of the Drama Manager to approximately 1 second, well within reasonable game time limits.

By comparing the three player models and the actual player traces, we found that the World Knowledge player model is the most accurate of the three as shown in Figure 43. We compared the first plot point chosen by each player model in five thousand runs as well as those chosen by our players from our user tests. We looked primarily at the first plot point choice since all other plot point choices were affected by DM actions. The uniform and Manhattan distance player models are very similar due to the small size of our dungeon. Because the World Knowledge player model followed the actual player choices the most accurately, we used this as our player in our follow-up test runs.

7 User Studies

As an exploratory study of the effect of the drama manager on the player's experience of EMPath, we performed play test with 31 players: 10 players played the game without the DM, 11 players played the game with the DM turned on and an equal weighting of the four evaluation features, and 10 players played the game with the DM turned on but with the manipulation feature ignored by the DM. In this last condition, the DM could be maximally manipulative in optimizing the experience. The purpose of this study was to gather survey data characterizing how the DM affected various aspects of the player's experience. The results of this study can then be used in the future to guide the design of a more formal, quantitative, statistical study.

We hypothesized that players would prefer the flow of the game, find the order of events to make the most sense, and get lost less often with the DM turned on. We hoped this could be accomplished with minimum feelings of manipulation.

7.1.1 Methods

Players initially filled out a survey describing their prior gaming experience. They were then given instructions for the game, including the goal of finding the exit and leaving the dungeon. The users were asked to talk aloud as they played. After finishing the game, an interviewer asked them six questions about the experience. The entire session was videotaped.

7.1.2 Results

As this was one of the first times DODM has been integrated into a real game world, the goal of these preliminary tests was to see if there were any positive effects of the DODM. We are extremely encouraged by the results we found.

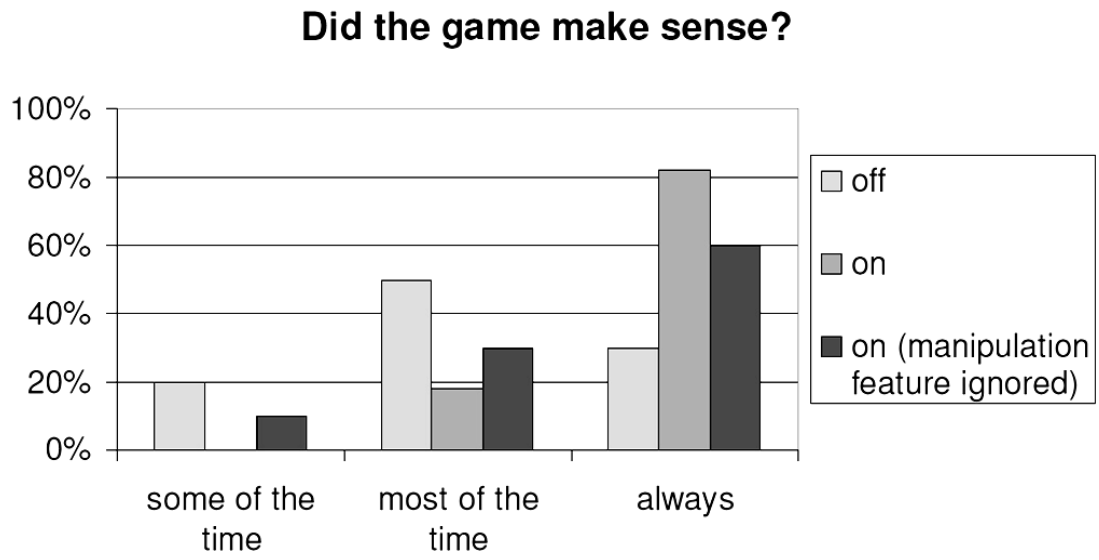

Figure 44: With the Drama Manager on, players found the game made sense more often. None of the participants answered that the game made sense "none of the time."

Question 1 asked if the players felt that the order in which they learned things made sense. In the group that played with no DM, players tended to express confusion about the flow of events in the game. Player 23 complained that “finding things so late in the game, it didn’t make sense”, while for player 27, who happened to discover the prisoner in the jail before learning anything about this sub-quest, “...the guy in the jail was random.” Generally, these issues would be dealt with by limiting the path of the player in some way, such as using locked doors. However, these tactics linearize the experience, removing player choice, and giving every player the same experience. This goes against our desired goal of giving players interesting and meaningful choices.

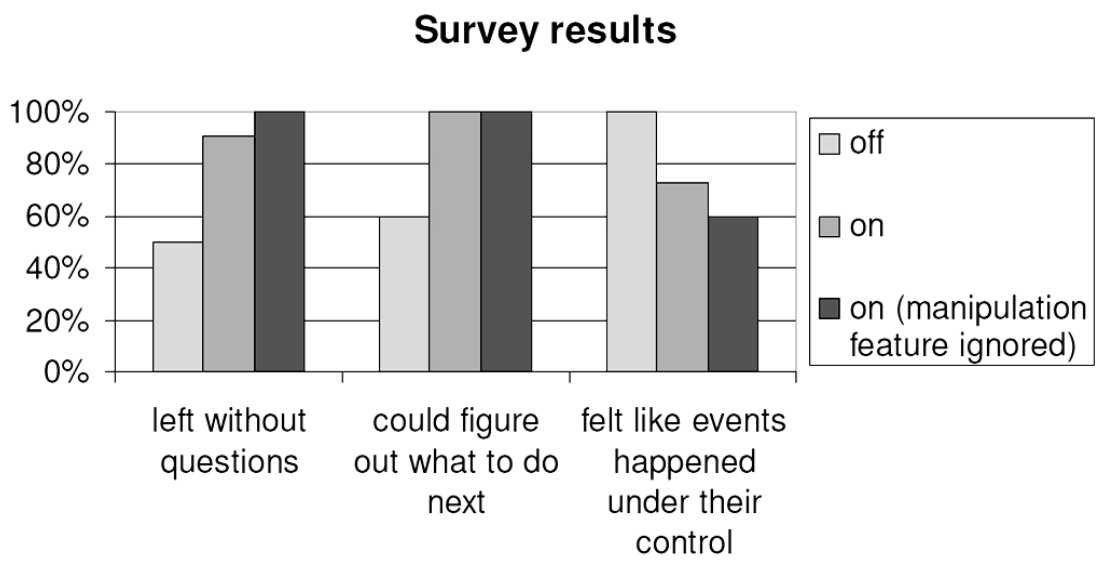

Figure 45: With Drama Management, players had fewer questions about the plot and felt lost less often. A majority of players felt in control of play experience, even with drama management on.

Within the group of players that played the game with the drama manager turned on, players were more likely to report the story made sense. Player 26 mentioned that the order made sense because "you got the notes before doing what they wanted." Likewise, player 16 felt that "the events that happened fit together." Overall, 82% of the participants playing with drama management reported the events always made sense, while only 30% of the control group felt the events always made sense as seen in Figure 44.

Question 2 asked players if they had any questions about items they had received in the game after playing. Our results show that players tend to have fewer questions when the DM is turned on. Player 13, who played without the DM, reported that, "I didn't know why I needed the candle until I reached the stairs," while player 10 who played with the DM turned on commented, "anything I found had information about it." In our tests, 50% of the players in the control group said they still had questions when they had finished playing,

while only 9% of the group with DM turned on had questions, and no one was left with questions in the group where DM was turned on and maximally manipulative as seen in Figure 45. Using traditional methods, we could have chosen to block off the exit until all plot points had happened. However, this can lead to extreme player frustration as they wander the dungeon finding the one thing they happened to miss. For our purposes, we felt that this frustration was not worth the trade off of having all questions answered upon game completion.

Questions 3 and 4 dealt with whether the players felt lost. Question 4 specifically asked the player if they sometimes didn't know what to do next. Without the DM, player 19 said that they got "lost and had a lot of backtracking", while player 40, who played with the DM, reported they had "very little wandering" and that "stuff would come up immediately." With the DM turned off, 40% of the players reported feeling lost, while no one from either of the drama managed groups stated that they had felt lost during the game. If the designer prefers the player to have more wandering, this can be adjusted through the story density evaluation feature.

Players did have some feelings of manipulation with the DM turned on, which was reported as feeling like things happened out of their control. Unsurprisingly, more players felt like things happened out of their control with the maximally manipulative DM. Player 40 commented that "notes would appear in rooms I'd been to before." However, others liked this dynamic feeling to the dungeon. Player 32 reported that the "candle melting was unexpected and neat." This DM triggered event is an alternative way to receive the candle. No one in the control group felt like things happened out of their control, while 27% in the

first DM group and 40% in the maximally-manipulative DM group felt like things happened out of their control. Adding more actions with lower levels of manipulation would help alleviate this issue. Even adding a second DM action with a much lower manipulation cost of adding a note to a nearby unexplored room would address the concerns by Player 40.

Not surprisingly, in the DM condition with the manipulation feature included, the DM chose to make far fewer moves. The DM would typically step in to help the player stay on one quest at a time, or to ensure the player had information before finding objects within the dungeon. The fact that infrequent guidance could dramatically improve player's perceptions indicates a decent "default" design for the game. One way we could have cheated the play-test is to purposefully design a poor default layout for the game world, requiring significant DM intervention for the world to make sense. This result shows that this is not the case.

One surprising result from our user tests showed that drama managed players had lower evaluation scores than those who did not. In our tests, players with drama management off had an average story evaluation score of 0.87 on a scale of 0-1 based on the DM evaluation features, while players experiencing the game with drama management on had an average story evaluation score of 0.75. This was exactly opposite of our qualitative data and our hypothesis.

There are a couple of possible reasons for this. The first is that the player model we employed is not sufficient for real gameplay. A random walk is not necessarily accurate to what an informed player might do. A player who has encountered the prisoner, then finds

the key to unlock the prisoner, is much more likely to free the prisoner as the next step, even if they are closer to a different plot point.

Another possibility is that our evaluation features are not accurately representing the desired outcomes. Given the feedback we received from our players, we feel that we were able to meet the goal of our objectives, but the evaluation features may not be detailed enough to represent these goals within the system. So while the outcome was players with fewer questions and less feeling lost, the system calculated that these things were not the case.

Finally, it is possible that one evaluation feature that was less important to the players was overwhelming the final score. For instance, if the players didn't mind whether they got plot points for one quest line together and therefore did not play with that goal in mind, and thought flow had an unusually high weighting, the play through could have a strong negative bias without the player feeling like the experience was worsened. This is a result of player goals and designer goals sometimes being at odds with one another.

8 Conclusion

Given the results from the user studies, and the experiences using drama management, we chose to use some elements of drama management within GrailGM. We chose to continue using the plot point construct within GrailGM, however the DODM system had some issues with scaling when adding complexity of a large story world, since the search space grows exponentially with the number of possible plot points and drama management moves.

Because of this, GrailGM uses a more reactive method of choosing plot points, without using search-based methods.

Additionally, the DM was not as well suited to typical genres as it could be. In its current incarnation, the DODM system only takes action after a plot point has happened. This is problematic as it is not uncommon in these games for a player to wander for long periods of time, presumably lost as to what to do next to move the experience forward. However, since the player is not triggering a plot point, DODM will take no action. We took a different approach in GrailGM, in which any plot point can be triggered after any player action through dialogue mix-ins. This allows the system to choose when a plot point should happen at a much finer level.

While the mechanics look quite different, the philosophy of drama management still plays a large part within GrailGM's dynamic plot point selection. Creating *EMPath* and refining the drama management system were invaluable lessons for designing GrailGM and *Mismanor*.

CHAPTER 8

MISMANOR

1 INTRODUCTION

The focus of this dissertation has been working towards flexibility in CRPGs reminiscent of that found in table-top RPGs, using AI systems to support playable story at the player, quest, and game levels. To address these issues and explore our approach, we created a playable experience entitled *Mismanor*, a social CRPG with a focus on emergent character interactions and dynamic quest selection.

The available actions within the social space are dependent on the player character's traits, a player's past actions, and the current social state. Similarly, quests are chosen based on a game character's current motivations and feelings towards the player, with quests having multiple completion states with different consequences to improve player agency at the quest story. Finally, plot points are chosen based on the player's actions within the game world, and the author's desired quest level story structure.

2 RELATED GAMES

Prom Week (McCoy et al., 2011) and *Façade* (Mateas & Stern, 2003) are both games that explore the space of social interactions. *Prom Week* is a social simulation in which the player chooses characters to interact, and selects social games for them to enact, while *Façade* is an open-ended dramatic scene between the player character and two game-controlled characters. While we are also interested in exploring the space of complex social

interactions, we are interested in how a player character — personalized based on the user's preferences — can influence and interact with an integrated social simulation within a CRPG.

Pataphysic Institute (PI) (Eladhari, 2010) is a multiplayer role-playing game that deeply connects a player's character with the game world. The player's traits and abilities are based on their personality and state of mind — tracked by the *Mind Module* (Eladhari, 2009). A player can create new enemies to defeat based on their actions within the game. Similarly, we also have an interest in deeply tying the player's character and choices to the story and available actions. However, in *Mismanor* we are focused on non-combat interactions.

Commercially available, *The Sims* series (Maxis & The Sims Studio, 2000) gives players the ability to create complex characters and play them in a simulated world. The characters can interact with objects and other characters, buying and building a living environment for their characters. While the characters and interactions are of particular interest to us, *The Sims* is a simulation at heart, and does not have the same quest or game level story concerns that are found within a computer role-playing game.

Storyteller (Benmergui, 2008) is a flash-based game in which a story is presented in three scenes: "Once upon a time", "When they grew up", and "The End". The player may move characters and objects around which updates the three scenes automatically, creating different stories. While this is a fascinating application of creating mechanics which strongly affect the game level story, it lacks the player-created character and quest level storytelling aspects found in most CRPGs.

Figure 46: Screenshot of Violet rejecting the Gossip social move.

3 Design of Mismanor

The game *Mismanor* is a historical fantasy, set at a manor in the countryside of an English-like country. There are six characters the player may interact with.

The Colonel (Douglas): The head of the household who has became estranged from his family while he was at war, and rather desperately hopes to reconcile with his daughter.

Violet: The Colonel's daughter who was scarred by the loss of her mother at an early age and not-so-secretly blames her father's absence.

James: The stable-boy who has been caught up in the family drama from his desire to stay close to Violet.

Thomas: The Colonel's son who is angry at the way Violet treats the Colonel as well as the Colonel's blindness to it.

The Baroness (Margaret): Sister-in-law of Douglas, and aunt of Violet and James, the baroness married into money and is drinking her way out of it.

Liz: The serving girl and now ex-girlfriend of James, and is rather unhappy about his current secretive relationship with Violet.

As the player character interacts with the family, it is gradually revealed that some of the game characters are members of a cult, and the player character was invited to the dinner party for not entirely innocent reasons. Interaction between the player character and game characters takes place through social moves or actions which result in dialogue exchanges and modifications to the social state of the game world.

3.1.1 Game Design

The choice to use CiF-RPG and GrailGM guided the design of the game in a number of ways. Because we removed combat from the game, and focused on social interactions as the core mechanic and main method of player level storytelling, it was necessary to create a game experience that supported the level of desired social interactions. Typical CRPG tropes did not seem appropriate, as they are generally chosen to give meaning to the combat-oriented nature of these games. Instead, we needed a world and game level story that would work with story progression limited to social actions.

Social Move	Intent
Ask Out (3-person)	Add status isDating(responder, other)
Ask Out	Add status isDating(responder, initiator)
Backstab (3-person)	Decrease familyBond(responder, other)
Bad Council	Decrease familyBond(responder, initiator)
Fueling the Fire	Decrease familyBond(responder, initiator)
Play Against (3-person)	Decrease familyBond(responder, other)
Council (3-person)	Increase familyBond(responder, other)
Council	Increase familyBond(responder, initiator)
Intervention (3-person)	Increase familyBond(responder, other)
Apologize	Remove status angryAt(responder, initiator)
Compliment	Increase buddy(responder, initiator)
Expose Suffering (3-person)	Increase buddy(responder, other)
Give Gift (Item)	Increase buddy(responder, initiator)
Gossip	Increase buddy(responder, initiator)
Share Interests	Increase buddy(responder, initiator)
Talk Up Other (3-person)	Increase buddy(responder, other)
Break Up	Remove status isDating(responder, initiator)
Break Up (3-person)	Remove status isDating(responder, other)
Accentuate the negative (3-person)	Decrease romance(responder, other)
Find Faults	Decrease romance(responder, initiator)
Serious Argument	Decrease romance(responder, initiator)
Court	Increase romance(responder, initiator)
Give Romantic Gift (Item)	Increase romance(responder, initiator)
Share Feelings	Increase romance(responder, initiator)
Blame	Decrease trust(responder, initiator)
Embarrass	Decrease trust(responder, initiator)
Confide In (Knowledge)	Increase trust(responder, initiator)
Reminisce (3-person)	Increase trust(responder, other)

Figure 47: Sampling of the fifty social moves available in Mismanor.

During design brainstorming, it was tempting to gravitate towards game ideas that revolved around psychosis or mental instability. Using this type of story lends itself to well-defined characters and interesting possibilities within dialogue. However, upon further consideration, we realized that this did not lead to ideal conditions for a player learning a new type of game mechanic. It is difficult enough to give the player enough information with which they may strategize about the outcome of their actions, but adding in psychosis and mental instability, it would make it much more difficult for the player to be able to predict the game characters responses to a chosen in-game action.

For this reason, we instead chose a somewhat H.P. Lovecraft inspired cult story, with less insanity and more interpersonal drama. The cult storyline lends some interest and tension to the story, while the interpersonal issues give the player a game level story landscape in which they may create interesting player level stories.

The social moves or actions were designed with this story in mind. There are 50 possible social actions available to the player and characters within the game, a sampling of which can be seen in Figure 47. Which actions are available are dependent on the current state of the world—described by the statuses and relationships between the characters—as well as the traits that define each character's personality.

Character Traits	Character Statuses
Kind	Category: Feeling Good
Sympathetic	Category: Feeling Good about Someone
Humble	Category: Feeling Bad
Forgiving	Category: Feeling Bad about Someone
Charming	Excited
Tactful	Cheerful
Loyal	Tipsy
Honest	Embarrassed
Independent	Shaken
Observant	Sad
Confident	Anxious
Persuasive	Guilty
Outgoing	Heartbroken
Shy	Confused
Impulsive	Drunk
Disapproving	Vulnerable
Irritable	Offended
Domineering	Distracted
Arrogant	Grateful Towards
Jealous	Angry At
Oblivious	Pities
Vengeful	Envies
Unforgiving	Afraid Of
Nagging	Resentful Towards
Snide	Estranged From
Indecisive	
Stubborn	
Dishonest	
Gabby	
Hostile	
Promiscuous	
Male	
Female	

Figure 48: List of the character traits and statuses used in Mismanor

Given the importance of the player's traits on which actions were available to them throughout the game, and the impact on the development of the story, we wanted to give players control over their traits during character creation. The list of traits available to the player can be seen in Figure 48. We use a character creation process similar to Hero System (Long, 2009) and GURPS table-top role-playing systems (Jackson, Punch, & Pulver, 2004). These systems support creating complex characters by choosing from a set of personality descriptions as opposed to all characters having the same descriptors to varying degrees which is common in current CRPGs. In Mismanor, the player starts out with a given number of points, and each trait has a set cost. Traits that are considered negative (e.g. *selfish*, *unforgiving*, etc.) add points to the player's pool instead of subtracting from it. This adds an incentive for the player to create an interesting character with both strengths and flaws, as flaws must be taken to be able to afford more strengths. Some traits are mutually exclusive, such that the player cannot have them both (e.g. *forgiving* and *unforgiving*). The costs of the traits were set after creating the social moves and micro-theories, and calculating how much effect each trait had.

Because traits play such a central role in player customization, we chose to highlight traits when designing and creating micro-theories. This means that our most robust micro-theories are those created around traits. For instance, the micro-theory describing the trait *charismatic* has 24 influence rules associated with it.

Choosing to create a social-based game meant that it was also necessary to change the design and focus of our quest level stories, or quests. In a traditional CRPG, quest types are focused on combat, movement, and environment manipulation (Oh & Kim, 2005) (Walker,

2007) (Sullivan, Mateas, & Wardrip-Fruin, 2012). For our purposes, quests are focused around changing the social state of the world — that is, increasing or decreasing relationship values, or adding or removing statuses. This gives us new quest types, and also forced us to re-evaluate the quest structure. Because human relationships are often more complex than killing, movement, and item manipulation, we felt that it was important for the system to notice how a player chooses to complete a quest. For instance, to return to the example quest shown in Chapter 5, Section 1, the player may be tasked with breaking up Violet and James, but they may choose to try wooing one of the characters as part of breaking them up. We felt this should lead to a different outcome, as the quest giver was likely to care about that detail. We therefore changed our quest design to accommodate multiple quest completion states.

Similarly, because the player is changing the social state of the world, it felt overly static for the quest giver to always give the same quest in the same way, regardless of their feelings towards the player. Because we already allowed multiple endings, we allowed each of the endings to be a possible desired goal state. In the previous example, if the Colonel likes the player character enough, he may be interested in having the player character woo his daughter Violet away from James.

Unlike killing, movement, and item manipulation — in which the character has simply either done it or not — having complex relationships allowed us to also include more complex completion states, including states that are the opposite of what the quest giver originally intended. While the player may have been asked to break up the ill-begotten lovers, the player can instead choose to strengthen their relationship to the point in which they decide

to elope. This leads to a very different quest ending and consequences. Because this is not a state that the quest giver is likely ever going to desire, it is not included in one of the possible quest introductions.

Because so much of the interaction in the game is delivered through dialogue, it made sense for our major story elements to also be delivered this way. Given the more open-ended nature of our quests, and the ability of the player to change the social state of the world, it didn't make sense for the storyline to be linear; if the player character had gotten close to Violet, they should see a different story than if they got close to the Colonel.

To accommodate this, we broke the story into plot points and categorized the plot points into story lines about each character, as well as the central plot line about the cult. We also identified eight different endings based on how the different characters felt about the player. While the player may see parts of each plot line, they will see only one complete character plot line in a given play through; the cult plot line is central to the overall story, so is always seen. Which character plot line the player sees is dictated by their standing with the various characters. A list of plot points is available in Figure 49.

3.1.2 Example

To help illustrate the capabilities of our system, we will look further at the quest in which the Colonel has learned about the relationship between his daughter, Violet, and the stable boy, James. He strongly disapproves of the relationship and asks the player character to break them up. The pre-conditions for receiving this quest are that the Colonel knows about

ID	Plot Point
0	Colonel doesn't like James hanging around with his daughter
1	James is obsessed with appeasing the Colonel
2	Violet and James are together a lot
3	Liz got James his job as a stable boy
4	Liz used to be in a relationship with James
5	Liz wants James back (unused)
6	Violet ignores the Colonel
7	Violet resents the Colonel for leaving when she was younger
8	Violet has been distant since Colonel returned home
9	Violet's mother died while Colonel was away at war
10	The Colonel is disappointed in Thomas
11	Thomas wants the Colonel's approval
12	Thomas is afraid of losing the Colonel
13	Thomas is protecting the Colonel
14	Liz thinks something suspicious is happening in the mansion
15	James doesn't like talking about the manor
16	Violet has strange scars on her wrist
17	The Colonel doesn't know about some rooms in the manor
18	James is afraid of Violet
19	The manor is the headquarters of a dark cult
20	James is a member of the cult
21	James is not entirely brainwashed
22	Violet is a member of the cult
23	Thomas knows Violet is running a cult
24	Violet is blackmailing Thomas
25	There is a cult ritual tonight to bring back Violet's mother
26	The player is the planned sacrifice for the ritual

Figure 49: List of plot points and associated ID.

the hidden relationship, the relationship is still active, and the Colonel has a low friendship towards James. The default completion state is that the relationship is no longer active.

The player has a number of options to accomplish this task depending on the situation and the player character's traits. If Violet or James trusts the player character enough, and the player character has the *manipulative* trait, one action available that can be chosen is to talk badly about the character's partner, lowering trust or romance to a point where a breakup will happen. If the player character has the *confidence* or *promiscuous* trait, the player may be able to improve the romance levels with that character to the point where the character will leave their partner for the player character. It is also possible to share a damaging secret about one of the lovers, or raise Violet's feelings for the Colonel to the point where she can be convinced to break up. If the player character successfully flirts with James, Violet (who has a high *jealousy* trait) will become angry at James (and the player character) which can be taken advantage of by the player to cause a break up.

Additionally, there are other possible completion states for this quest. One ending is to improve the relationship between the Colonel and the stable boy, James, such that the Colonel no longer disapproves of the relationship. Another ending is that the player can choose to strengthen the relationship between James and Violet to the point that they elope. This removes them from the game, which leads to a different game ending, as Violet is one of the key players in the cult.

All of these things could, theoretically, be accomplished through traditional technical approaches to quest implementations. However, creating quests with a large number of

possible completion paths and goal states is rarely done in commercial RPG development, even by teams that are initially interested in such approaches. As discussed previously in Chapter 5, Section 1, standard implementation approaches such as quest flags and branching if-then trees used in CRPGs such as *Planescape: Torment* (Black Isle Studios, 1999) and *Dragon Age: Origins* (BioWare Edmonton, 2009) allow for some dynamic behavior, but create a rigid and exponentially growing structure as the quest flexibility grows. We have developed an approach for supporting such quests at a deep technical level, rather than requiring extensive ad-hoc work by designers on a foundation developed for more linear, single-solution quests.

4 Player Actions

While our decision to create a game based on social mechanics helped form our design choices for the game, the design choices for the game also helped refine our game mechanics and AI systems. In particular, GrailGM's original design was modified to handle the dynamic nature of the quest and story structure within our game.

With the design of *Mismanor* focused so heavily on social interactions, it was important for us to have a system that could richly simulate the social landscape of our game. A key aspect of the gameplay within *Mismanor* is that the player needs to actively strategize—molding their character and the social situation—to manipulate what actions are available.

4.1 User Interface

Creating a social CRPG presented a challenge in creating a user interface that displayed enough information without overwhelming the player. While other CRPGs may have social

simulation aspects, they track much less information, and the reason for an action happening is generally based on one or two checks, making sure the players stat is over a specified threshold. *The Sims* is an exception to this rule, but the user interface can be overwhelming with the number of state graphs. Additionally, *The Sims* is a simulation and does not involve a player character. Introducing a player character has an impact on the design of the user interface and on the way the player interacts with the game. In *Mismanor*, we wanted to strike a balance – offering complexity beyond that of *The Sims* while keeping the user interface intuitive and understandable.

In *Mismanor,* the player interacts with the game by choosing a character to interact with and then choosing a social action they would like to initiate with that character. Which social action is available to the player is based on the player's character traits and statuses, as well as the current social state of the world. The saliency of the move is calculated based on micro-theories and influence rules which describe how the character's traits and statuses affect their willingness to take a specific action. As discussed in Chapter 4, micro-theories are used as generalized knowledge of how specific traits, statuses and relationships weight different actions based on their intent. The intent is used to describe the desired outcome of a social action by the character initiating that action.

For instance, working with our previous example of attempting to break up Violet and James, the micro-theory about the *promiscuous* trait gives a positive weighting to all social moves with the intent to increase romance. In addition, each social action has special case influence rules which describe how that particular action is modified by various traits, statuses, relationships and history. So while the *promiscuous* trait micro-theory gives a

general positive weighting to social moves which increase romance, the trait would have a negative weighting towards the social move *Indirect Conversational Flirt* as a promiscuous character would be less likely to do something so subtle.

These factors are all used to calculate a weighting that represents the interest that the player character would have in initiating that action. The player is presented with the top four positively weighted actions to choose from.

The character that the player is interacting with can then choose to accept or reject the intent of the action. As previously discussed in more detail in Chapter 4, after the character accepts or rejects the initiated social action, the effects of the action are resolved. Every action changes the social state, either by changing a relationship or adding or removing a status. The effect, or outcome, of a social action is chosen based on the traits, statuses, and relationships between the two characters, as well as any history between them.

When a character accepts or rejects a social action, the game displays the top weighted rule for making that decision in the status bar. This was done to give the player some insight into the system, as well as clues towards what they can do differently to get a different outcome if desired. When a social move resolves, the chosen effect is also displayed on the status bar. This shows not only what the outcome of the move is, but also the condition for the change in the effect, so the player can understand what conditions their character, or the character they are interacting with, met to receive that particular effect.

Every action is also stored in a history of actions, called the Social Facts Database (SFDB). Some actions are tagged with author-specified information such as “romantic action” or

"mean action." These can be referenced in future actions to both modify the weights of which actions are available or chosen, as well as which effect (the dialogue instantiation and social state change consequences) is played out.

5 Story Expressivity Evaluation

For *Mismanor,* we are interested in measuring the amount of *expressivity* of the system, or the breadth and depth of the story space, and how well the story space supports different types of players. Story "goodness" is in some sense guaranteed by the required central cult plot point progression, with player adaptability being found in the side stories and different endings. To test the capabilities of the system, we created an automated player which could use various player models for both choosing social moves and choosing quest completion states.

5.1 Player Models

For our tests we used seven different player models: *random, nice, mean, shy, social,* and two versions of a *quest finisher.* The base model is the *random* model which chooses a random gender, a random set of 7 traits, and chooses its moves at random. A move can either be to move to a new location, or to interact with a character in that room. When interacting with a character, the player model chooses a social move from those presented at random.

The *nice* and *mean* player models are the opposite of one another. The nice player model has the kind, forgiving, sympathetic, loyal, honest, and tactful traits, while the mean player model has the irritable, domineering, vengeful, unforgiving, snide, and hostile traits. When

presented with social moves to choose from, the player models base the choice on the intents of the social moves. The nice player model prefers, in descending order: add positive status, remove negative status, network up, network down, remove positive status, add negative status. Likewise, the mean player model prefers, in descending order: add negative status, remove positive status, network down, network up, remove negative status, add positive status. In the case of multiple moves with the preferred intents, these player models will choose the move in which they have the highest volition.

The *shy* and *social* player models focus more on how the player models interact with the various non-player characters. The shy player model has the shy, humble, sympathetic, indecisive, and observant traits, while the social player model has the charming, charismatic, outgoing, gabby, and confident traits. When choosing which action they will take, they both take into account how long they have been talking to a specific character. In the case of the social player model, the player will not talk to the same character more than 3 times in a row and the player character has a high chance of choosing a different move. For the shy character, they will not talk to the same character more than 8 times, however there is a high probability that they will talk to the same character more than once in a row.

Finally, the *quest finisher* player model was created to work towards completing quest goals. There are two versions of the quest finisher, one that works towards the stated goal of a quest, and one that works against it. Both versions of the player model choos 7 traits at random. When presented with a choice of social moves, the quest finisher player model will choose a move based on what quest is currently active. This is done by matching the intent of the quest with the intent of a social move. If the intent is the same, the version of the

player model working towards the quest goal will pick that move. Otherwise, the other version of the player model will choose an opposite move. This means that the player model will try to move the specific network down if it the quest intent is for that network value to go up, and vice versa. For statuses, the antagonistic player model will attempt to remove a status if the quest intent is for that status to be added and vice versa.

If the intents are not a match between the available social moves and the active quest, the quest finisher player model attempts to find a similar social move to the desired move. This was achieved by pairing networks with specific statuses, and looking for moves within that pairing. For instance, the version of the quest finisher player model working towards a quest goal on a quest with the intent romance up, would prefer a romance up social move if possible, but if not found, would prefer a social move with the intent to add is dating. Similar pairings were done for the version of the quest finisher AI that works against a quest goal. If no preferred moves were found, the quest finisher player model chooses a move with the highest volition.

Additionally, the quest finisher AI player model has a high probability of talking to non-player characters that are involved in the quest. To do this, we used the tags associated with quests (quest tags and their use are described in Chapter 5, Section 1.4) to note which characters were associated with specific quests. Similarly, when playing social moves which involve a third character, the AI will try to choose a character involved in the quest if possible. This was done to maximize the amount of social change with the characters involved in the quest.

To make sure the AI did not get stuck talking to one character, we built in the concept of *boredom* in which the character will not talk to the same character more than 3 times in a row. Similarly, if it has been 10 moves since a plot point has happened, the AI will revert to randomly choosing a move. This was done to help alleviate the AI getting stuck and not being able to move the story forward.

5.2 Authorial Heuristics

When choosing quests and plot points, we use a series of heuristics to model author preferences for the story and quest progressions. Each heuristic is assigned a weight and the weights are normalized.

5.2.1 Quest Heuristics

For quest selection, we used three heuristics, *questMixing, questConcentration,* and *intent.* The *questMixing* heuristic prefers quests that are different types from previous quests used. An ordered history of past quests is tracked, with the position in the history assigned a value according to a geometric progression with a common ratio of ½. For every past quest that has the same type as the considered quest, that weight is added to a total score. Once the total score is found, the score is negated and multiplied by the questMixing weight assigned by the author. This gives a higher negative score to quests that have been used recently or multiple times in the past.

For *questConcentration* the opposite is true, in which quests of the same type are preferred. We use the same calculation as questMixing, but do not negate the score before multiplying

it by the questConcentration weight. These two heuristics are provided to account for different preferences across authors.

The *intent* heuristic applies CiF-RPG reasoning (discussed in Chapter 4) to the intent of the quest, to calculate the volition of the quest giver to give that quest. Because quests have goal states that specify a specific social state (quests and goal states are discussed in Chapter 5, Section 1), we first look at the goal states of the quest and match which one the quest giver would choose when assigning the quest. We then use the quest giver as the initiator, and the characters or items assigned to the goal state as the responder and other. We then run the micro-theory rules with those assignments to calculate volition for the quest giver to want this outcome. This may seem counter-intuitive, in that we are not calculating the intent for the quest giver to actually give the quest, but the point of giving the quest is that the quest giver wants the goal accomplished. If the quest giver does not have a high desire for that outcome, then they would have a low volition to assign the quest. Initiators, responders, and micro-theories are all discussed in detail in Chapter 4.

5.2.2 Plot Point Heuristics

For plot point selection, we use the following three heuristics, *storyCohesion, plotMixing,* and *plotConcentration.* The heuristic for storyCohesion prefers a story that goes deep instead of broad. For instance, if the player starts to learn about a subplot, storyCohesion prefers that the player learn more about that subplot. This is calculated by weighting each possible plot point by the number of story pre-conditions that were required for that plot point to be available. The heuristic calculates upwards through the game tree, so that the further down sub-plot, the higher the weighting.

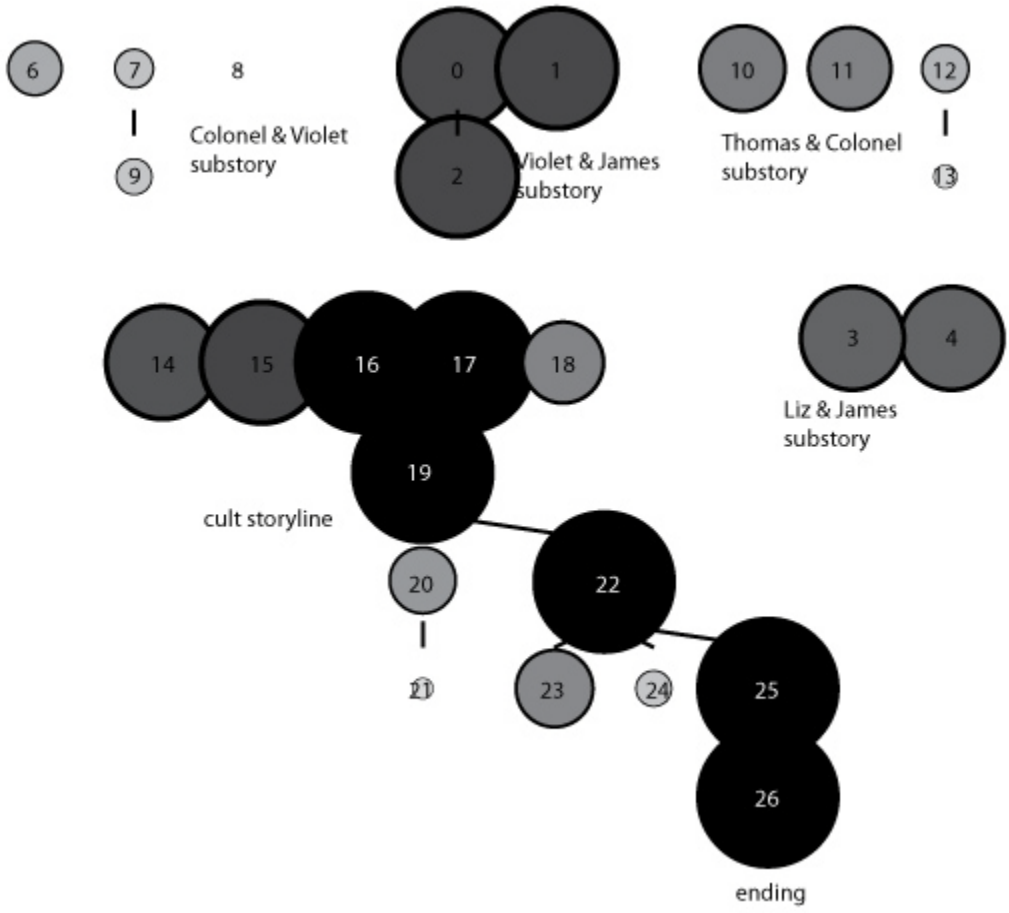

Figure 50: The percentage of time a plot point is seen over 1000 runs by the random player model. White means never visited, black means always visited. The size of the plot point node also corresponds to the number of times it was seen. The larger the node, the more often that plot point was seen.

For *plotMixing* and *plotConcentration*, the calculation is very similar to the questMixing and questConcentration heuristics explained in the previous chapter. However, instead of looking at quest types, plotMixing and plotConcentration instead look at which plotline a particular plot point is part of. For instance, the plot point “Thomas is protecting his father”

is part of the Thomas plotline. Another difference is that plotMixing and plotConcentration change weight over the course of the game. This is because our authors preferred higher plotMixing at the beginning of the game, but higher plotConcentration as the game progressed. So a maximum and minimum are specified for the two heuristics, and they move from one extreme to the other as the game progresses.

5.3 Results

We ran the automated tester 1,000 times for each player model, tracking plot points, quests, quest endings, game endings, and social moves that each player model chose over the course of each game. We also ran the automated tester 100 times with the random player model with only one heuristic active for each of the plot point and 100 times with the questfinisher player model for the quest heuristics. Our hypotheses are twofold.

#1. For story to be playable, there must be a discernible effect on the story for different types of players, unlike other CRPGs in which every player type experiences the same story. Therefore, we should see differences in quest, plot point, and game ending distribution based on the player models favoring different game mechanics.

#2. We use three heuristics to choose plot points and three heuristics to choose quests. Changing these heuristics gives authors some control over the game output. Therefore, when isolating each heuristic, there should be differences in which quests and plot points are favored.

Nice AI	Mean AI
The Colonel is disappointed in Thomas	James is obsessed with appeasing the Colonel
James is obsessed with appeasing the Colonel	Liz used to be in a relationship with James
Colonel doesn’t like James hanging around with his daughter	Colonel doesn't like James hanging around with his daughter.
Violet and James are together a lot	Violet and James are together a lot
Thomas wants Colonel's approval	James doesn't like talking about the manor
James doesn't like talking about the manor	Liz thinks something suspicious is happening at the mansion
Thomas is afraid of losing the Colonel	Colonel doesn't know about some rooms in the manor
James is afraid of Violet	Liz got James his job as a stable boy
Violet has strange scars on her wrist	Violet ignores the Colonel
Liz thinks something suspicious is happening at the mansion	Violet has strange scars on her wrist
Colonel doesn't know about some rooms in the manor	The manor is the headquarters of a dark cult
Thomas is protecting the Colonel	Violet is a member of the cult
The manor is the headquarters of a dark cult	There is a cult ritual tonight to bring back Violet's mother
Violet ignores the Colonel	The player is the sacrifice for that ritual
James is a member of the cult	ENDING: Sadly having made no friends in the manor the player is sacrificed to raise Violet's mother.
Violet resents Colonel for leaving when she was younger	
Violet's mother died while Colonel was away	
Violet is a member of the cult	
Thomas knows Violet is running a cult	
There is a cult ritual tonight to bring back Violet's mother	
The player is the sacrifice for that ritual	
ENDING: Realizing her love for the player and her distaste for her brother Violet is convinced to sacrifice Thomas instead of the player.	

Figure 51: Sample playtraces for the Nice and Mean player models.

5.3.1 EXAMPLES

To illustrate the differences at the three levels of storytelling that we were interested in improving playability within *Mismanor,* we took a sample playtrace from the mean and nice player models, shown in Figure 51.

The first noticeable difference is that the mean player sees fewer plot points. This is extreme in this particular case, but is consistent with the averages of 17.4 plot points per game for the nice player model versus 15.7 plot points per game for the mean player model. This is due to the fact that the mean player model has a slightly lower chance at receiving dialogue mix-ins due to negative social move instantiations having a high tendency for the $hate$ mix-in tag, but there being fewer plot point revelations associated with a hateful mood.

In our example, the nice player model saw the entirety of the Thomas and Colonel sub-story, while the mean player model saw none of it. This is also consistent with our distribution graphs shown in Figure 53, and described in more detail in Section 5.3.2. In the example play-trace, the entire Thomas and Colonel sub-story was revealed with the like mood from Thomas, while none of the plot points were revealed by Thomas to the mean AI. These are examples of how a player's actions can affect the game level story, by changing the prevalent mood of the dialogue exchange, which changes the probability of different plot points being revealed.

At the quest level, the mean AI receives the quest "Get Colonel to like James" after hearing that James is obsessed with appeasing the Colonel. This is because the plot point and quest share the same tagged characters, and James has a high volition towards getting the Colonel to like him. However, the mean player model has a stronger weight towards lowering

networks than raising them, so after initiating *Express Disappointment* and *Argue About* with Colonel towards James, the Colonel dislikes James enough that a completion state is matched. In this case, the completion state is not the same as the goal state, so James trusts the player less, gains the *angryAt* status towards the player, and lets the player know through dialogue that he is angry at making the Colonel dislike him so much.

In contrast, the nice player model receives the quest to help Liz get over James (lose the heartbroken status) after learning the plot point that Liz got James his job as a stable boy. Because the nice player model is weighted towards social moves such as *Conversational Flirt* and *Share Feelings,* it does not take long for the nice player model to meet the completion state that matches the given goal state, and Liz loses the heartbroken status. This raises James' trust towards the player as well as gives James the *gratefulTowards* status towards the player.

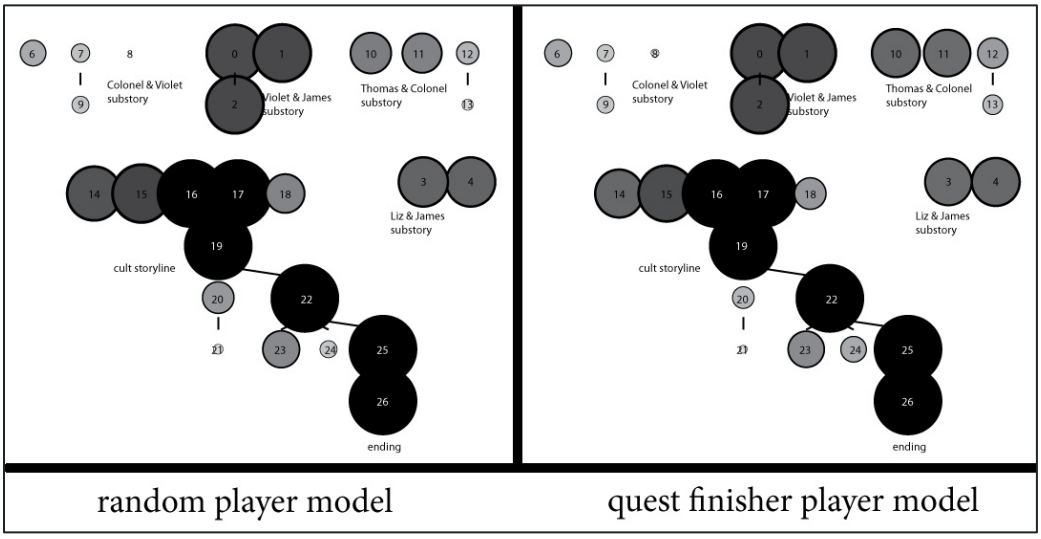

Figure 52: The random (left) player model and quest finisher (right) player model show the largest difference in plot point distribution.

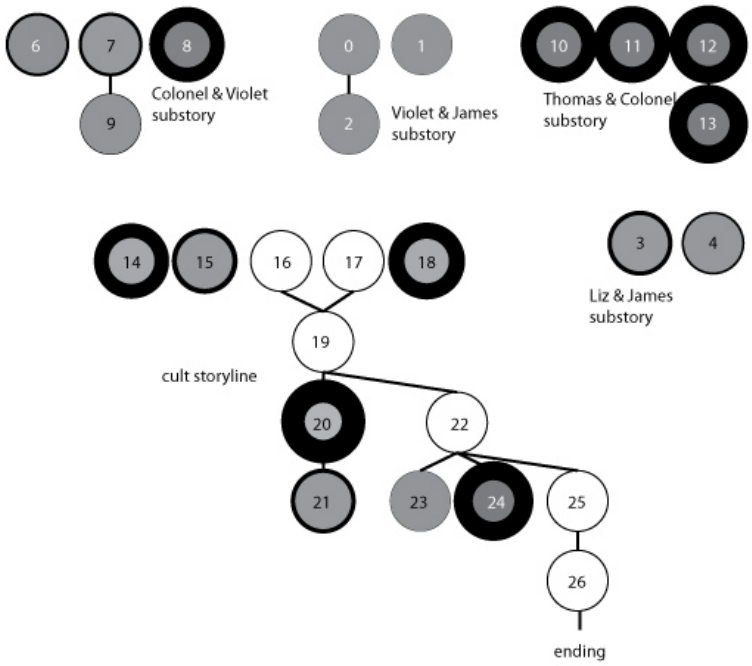

Figure 53: Difference map between random and quest finisher player models. The white nodes are the required path, so there were no differences found. The size of the border around a node indicates the size of the difference. White numbers mean that the quest finisher had the larger value, black numbers mean the random player model had the larger value.

At the player level, the nice player model raises Violet's romance at first with *Conversational Flirt* and *Court*. Then, the nice player model chooses *Ultimatum*, asking Violet to break up with James and choose the player instead, which Violet accepts. This leads to the end game scenario in which Violet decides to save the player and instead sacrifice her brother in the ritual. The mean player model meanwhile chooses *Confide In* (there are no network down moves available) with Violet, raising her trust enough that she accepts *Accentuate the Negative (3-person)* towards James. This causes their romance to lower, but unfortunately,

the player model manages to make a bad enough impression that they are still sacrificed at the end of the game.

5.3.2 Plot Point Distribution

We created plot point distribution graphs to demonstrate the percentage of time particular player models saw each plot point. For the plot point distribution graphs, we first calculated the percentage of time each plot point was triggered over the thousand test runs for a particular player model. Each node is colored based on that percentage from white to black. If a node was never visited, it is colored white, and if a node was always visited (as is the case for the required plot line) it was colored black. Similarly, the size of the node is indicative of the percentage of time a plot point was visited. A small node was rarely visited while a large node was visited often. A graph was created for every player model.

As is shown in Figure 50, there is not an even distribution of plot points with the base (random) player model. This is due to the fact that not all plot points are equally easy to discover. While a plot point is active (possible for the player to trigger), the plot point is not triggered until the player is interacting with the appropriate NPC, and there is a mix-in available of the appropriate mood. This means that if a plot point may only be revealed by one NPC and only with one mood type, it is far less likely to be triggered than a plot point that has multiple NPCs and multiple possible moods. Additionally, because the neutral and like moods are the most common plot point mix-in type, only providing plot point instantiations for one of the extreme moods (love or hate) makes the plot point much less likely to trigger.

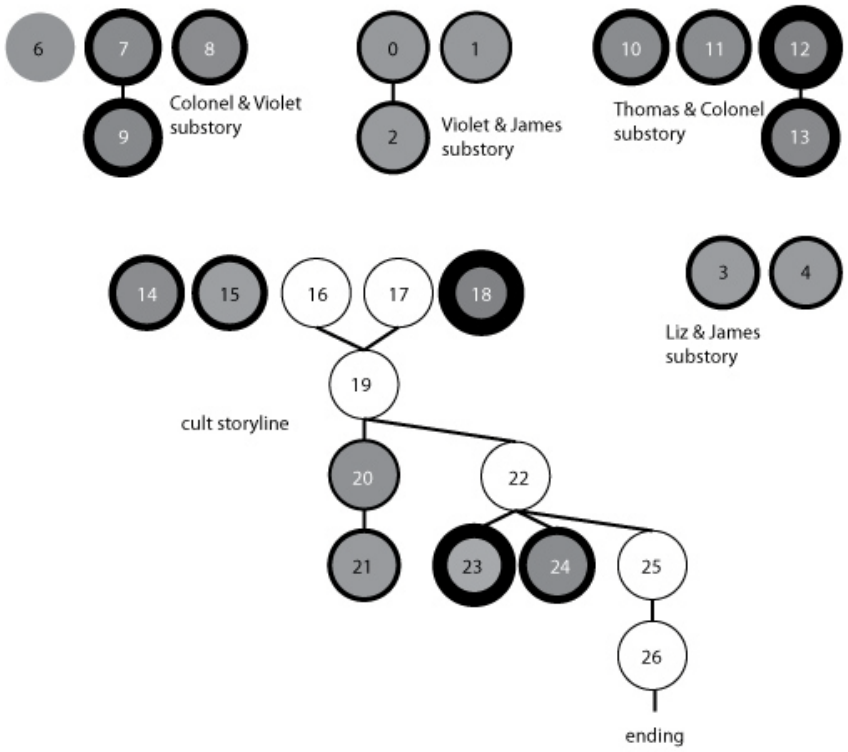

Figure 54: The mean and nice player model difference graph. The white nodes are the required path, so there were no differences found. The size of the border around a node indicates the size of the difference. White numbers mean that the nice player model had the larger value, black numbers mean the mean player model had the larger value.

There were differences between every player model. However, the most significant differences were between the random and quest finisher player model (shown in Figure 52 and Figure 53), and the mean and nice player model (shown in Figure 54). The quest finisher player model saw a larger number of plot points dealing with the Thomas and Colonel subplot, while seeing fewer plot points associated with the cult plot line in comparison to the random player model. This is due to the fact that almost 60% of the time, the quest finisher player model received the quest from Violet asking the player to hurt the

relationship between Thomas and the Colonel. This would cause the quest finisher player model to focus on interacting with those characters which would in turn make it more likely that the player model would see those plot points. The predominance of this quest is discussed further in Section 5.3.4.

The differences between the mean and nice player model are also understandable, as they are two extremes of the same spectrum, and we would expect to see telling differences between the distributions. These differences are due to how the player models skew towards different interaction types. A mean interaction is more likely to trigger a hateful or neutral mix-in, while a nice interaction will more likely trigger a love or like tagged mix-in.

	Mean player model	Nice player model
Hate	15.7%	2.4%
Neutral	38.9%	28.3%
Like	40.6%	60.1%
Love	3.9%	8.8%

Figure 55: Distribution of plot point moods. The mean player model triggered hate and neutral mood plot point instantiations more often, while the nice player model triggered like mood instantiations the most.

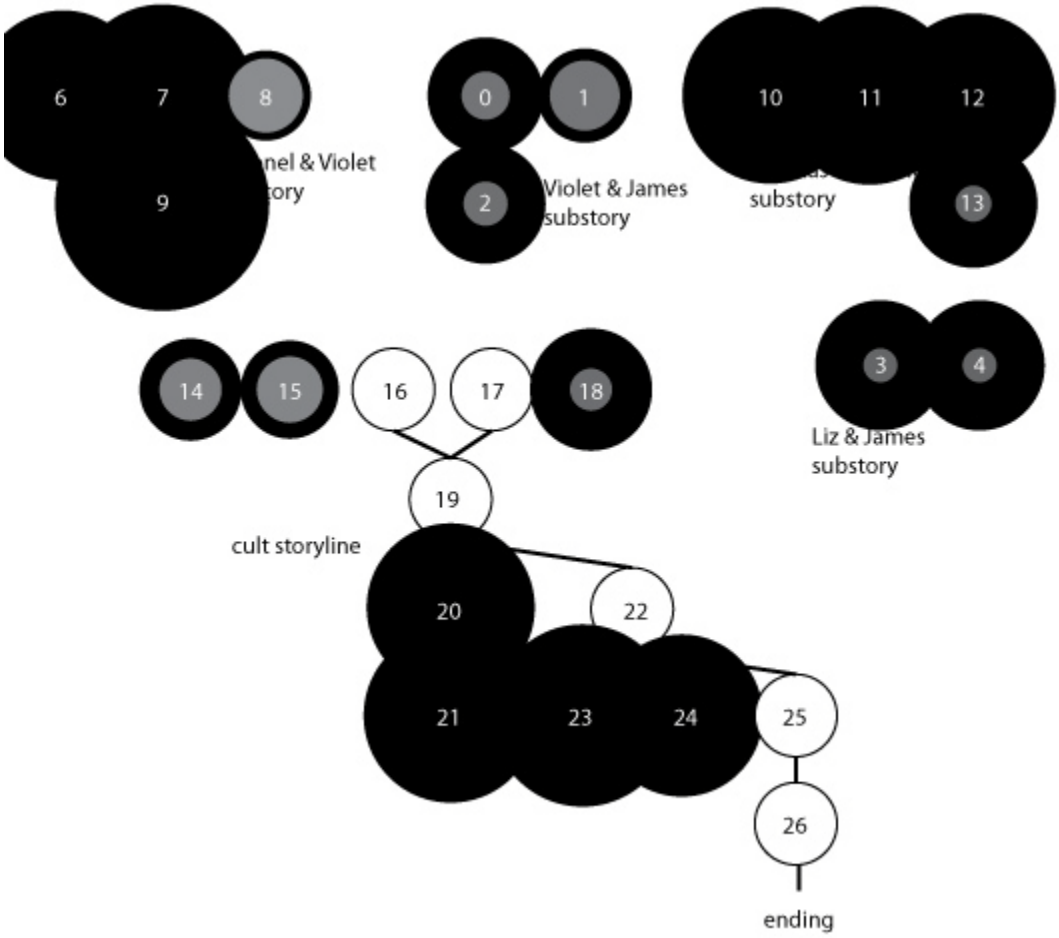

Figure 56: The plot concentration heuristic favors sticking to the same storyline, while the plot mixing heuristic favors a sampling of plot lines. The difference between the number of times a player model visited each node based on each heuristic is shown here. The size of the border indicates the size of the difference. In all cases, the plot mixing heuristic was the larger of the two values.

Because some plot points don't make sense to be told in certain moods, there is sometimes no mix-ins available for that particular mood type. For instance it does not make sense for

Thomas to reveal to the player that he is afraid of losing his father, the Colonel (plot point ID #12) in a hateful or angry way. Therefore, the mean player model triggered that plot point only 26% of the time, versus the nice player model which triggered the plot point 37% of the time. Similarly, plot point #23 (Thomas knows Violet is running a cult) is not something the authors felt should be told in anything other than one of the extreme moods. Therefore, the mean player triggered this plot point 54% of the time versus the nice player who triggered it 44% of the time. The total distribution of mix-ins can be seen in Figure 55.

When testing plot point distribution across authoring heuristics, it was unsurprising that the biggest difference was found between the *plot mixing* and *plot concentration* graphs. These two heuristics prefer different author goals; that of sampling many different plots versus sticking to just one plot line. Figure 56 shows the large difference in plot point coverage for each heuristic. The plot point distribution graphs show that by modifying the weights of these two heuristics, the author is able to get either coverage or depth.

5.3.3 GAME ENDINGS

In *Mismanor,* there are four possible prefabricated endings, and many possible generated endings. For ease of discussion, we grouped all generated endings into one category. We also added a forced ending if the player model had completed 1000 moves without triggering a new plot point, as that was indicative of the AI getting to an unwinnable state. According to our hypothesis, different player models should be able to have an influence on the game ending for the player's choices to have been meaningful. As seen in Figure 57, this is the case.

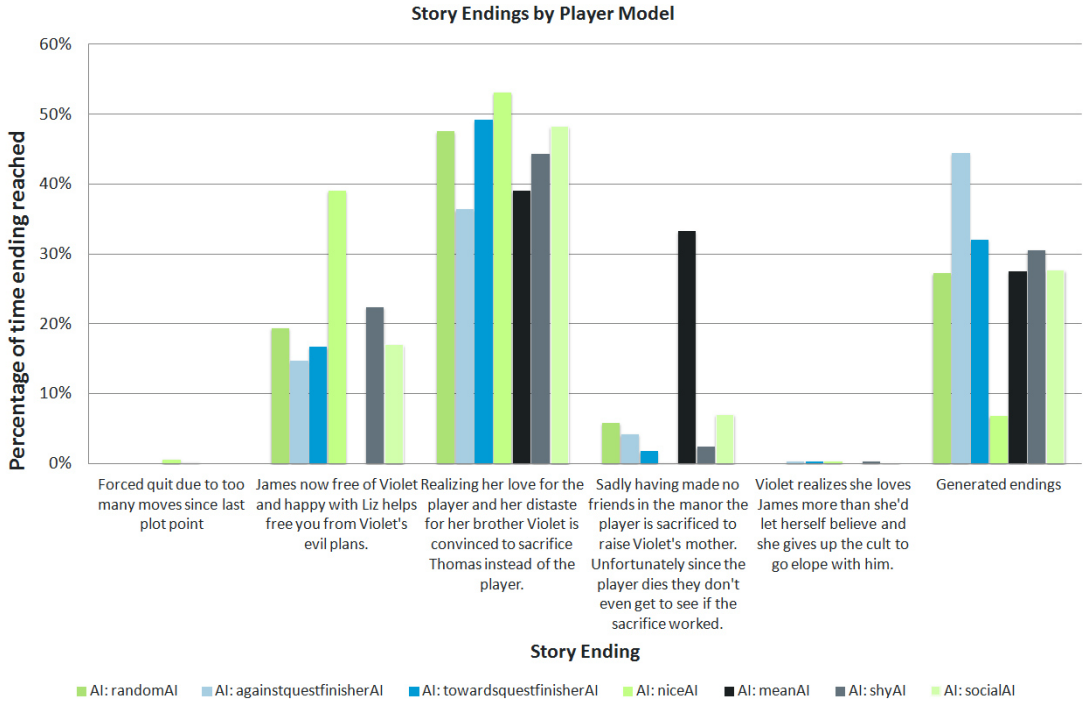

Figure 57: Shown above is the percentage of time an ending appeared by player model.

The large spike for the mean player model happens at the bad ending. This happens 33% of the time for the mean AI. This makes sense, as that ending correlates with the player not having grown close to any of the characters, or helping improve the relationships between other characters. This is an obvious preference in our writing, and the authors could account for this by creating different endings for negative outcomes if desired.

Not surprisingly, the nice AI has the fewest number of bad endings. Instead, the nice AI has a higher percentage of endings in which the player survives in different ways. The nice AI has the least number of generated endings due to the player model reaching a high number of authored endings. As many of the endings are related to improving relationships between characters, and the nice AI is designed to improve relationships, it makes sense that these

endings would be more likely to be reached. The questfinisherAI that works against the quest goals has a higher number of generated endings than any other player model. Because many of the endings correspond to quest goals that they player may have reached throughout the game, it makes sense that the player model working against these goals would not reach the generated endings.

The ending in which Violet and James elope is very rare. This is because it is very difficult to get characters to elope. It requires a very high romance and a specific set of circumstances. Therefore, it is not surprising that this ending came up so infrequently. This was a design choice, and is not considered a problem with the system.

There are still a large number of generated endings, as the player models all had 27-44% generated endings (other than the nice AI which reached mostly pre-authored endings). If the author would prefer the player to reach fewer generated endings, this could be alleviated by creating more special case endings.

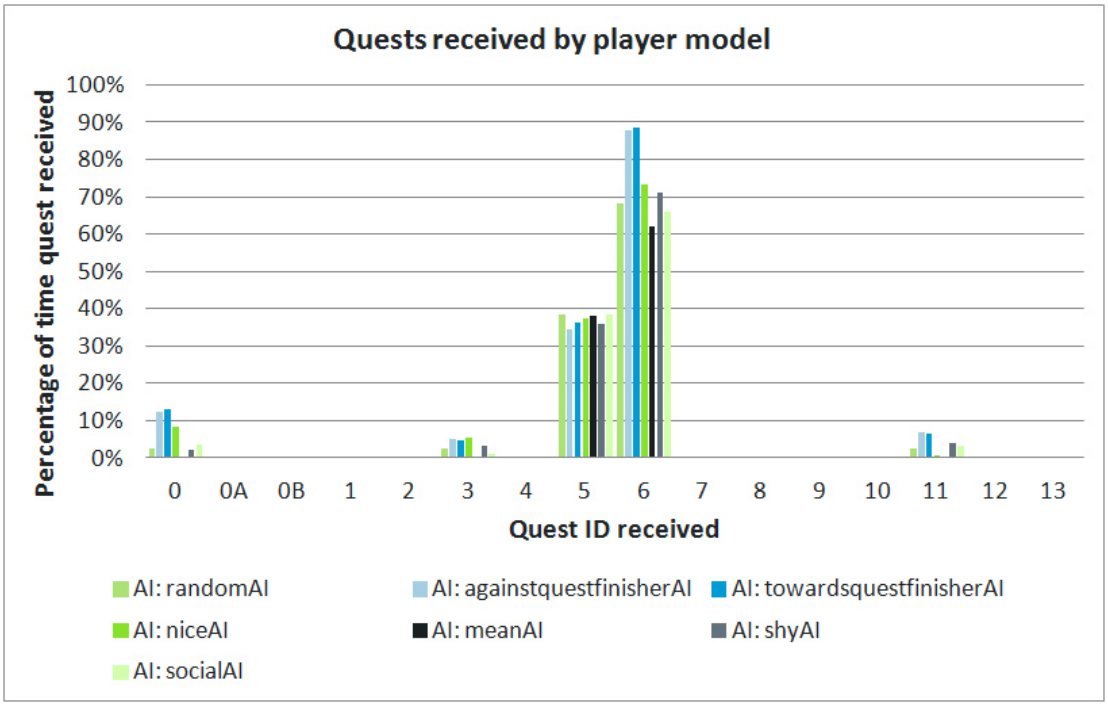

Figure 58: Quests initiated based on player model. This shows the percentage of time a particular quest was received in 1000 runs by each player model.

5.3.4 Quest Distribution

Our hypothesis is that for playable quests, the quests the player receives must change based on the player's actions (to have meaningful impact on the story). In addition, the ways in which the quests were completed should also be different based on the player model. The game should be able to support different styles of play and have meaningfully different outcomes based on that play. Our game had 14 possible quests, with one quest being an example of having three possible goal states. The quest distribution is shown in Figure 58. The list of IDs is found in Figure 59.

ID	Quest Giver	Quest Name
0	Colonel	Break up James and Violet
0A	Colonel	Get Violet and James to break up by dating James.
0B	Colonel	Get Violet and James to break up by dating Violet.
1	Violet	Convince Liz to get over James and date someone else.
2	Thomas	Find evidence of Violet's misdoings by obtaining her diary.
3	Liz	Get Baroness to like Liz enough to hire her.
4	James	Get Colonel to like James
5	Liz	Get James to come back to Liz.
6	Violet	Get Thomas to stop protecting the Colonel.
7	Colonel	Get Violet to like her father again.
8	Thomas	Help Thomas repair his relationship with James.
9	James	Help Liz get over James.
10	James	Find out how serious Violet is about the cult by getting her diary.
11	Liz	Find out what's going on with the house by getting Violet's diary.
12	Violet	Get access to the Colonel's secrets.
13	Colonel	Get Thomas to stop meddling in my affairs.

Figure 59: List of quests and their associated IDs. ID 0A and 0B refer to other possible stated goal state for quest 0.

As shown, both questfinisherAI variants received the quest *Get Thomas to stop protecting the Colonel by ruining his family bond* more frequently than the other player models, receiving the quest 88% of the time regardless of the variant. All the player models received this quest frequently, in part because the game is designed to give a quest as soon as possible. This quest is given by Violet, who is in the room that the player starts in. Additionally, Violet has a high volition towards giving this quest at the beginning of the game, so the intent heuristic weights this quest quite high.

The meanAI received this quest the least amount, likely due to the player model not meeting the relationship threshold pre-requisite for receiving the quest. Many quests have pre-conditions that the starting relationship between characters is not too low to begin with. A mean player model would have a tendency towards lowering networks between all characters, not just between the player character and NPCs.

The questfinisherAIs both received a larger number of quests in general as they are more likely to finish quests and get a second quest, so had a higher chance of receiving this quest in general. The questfinisherAI variations both completed 32% of the quests received, versus 1%-12% completed for the other player models.

While there is a noticeable difference in quest distribution, there would be a greater distribution if the player models were able to do more quests during a playthrough or if the game did not assign a quest as soon as possible. Additionally, as seen in the quest heuristic results in Figure 60, because the intent heuristic was weighted quite heavily in our test runs, it had a strong effect on the quests that were received by the different player models. We discuss the quest finisher AI more in the discussion section below.

When looking at the quest heuristics, we found that there was very little difference between quest mixing and quest concentration. This is due to the fact that the game is not balanced for long enough game play to give the player models a chance to complete quests. Additionally, creating a player model that is adept at completing quests is a complex task, requiring its own research. On average, ~1.1 quests were initiated per game over all of the

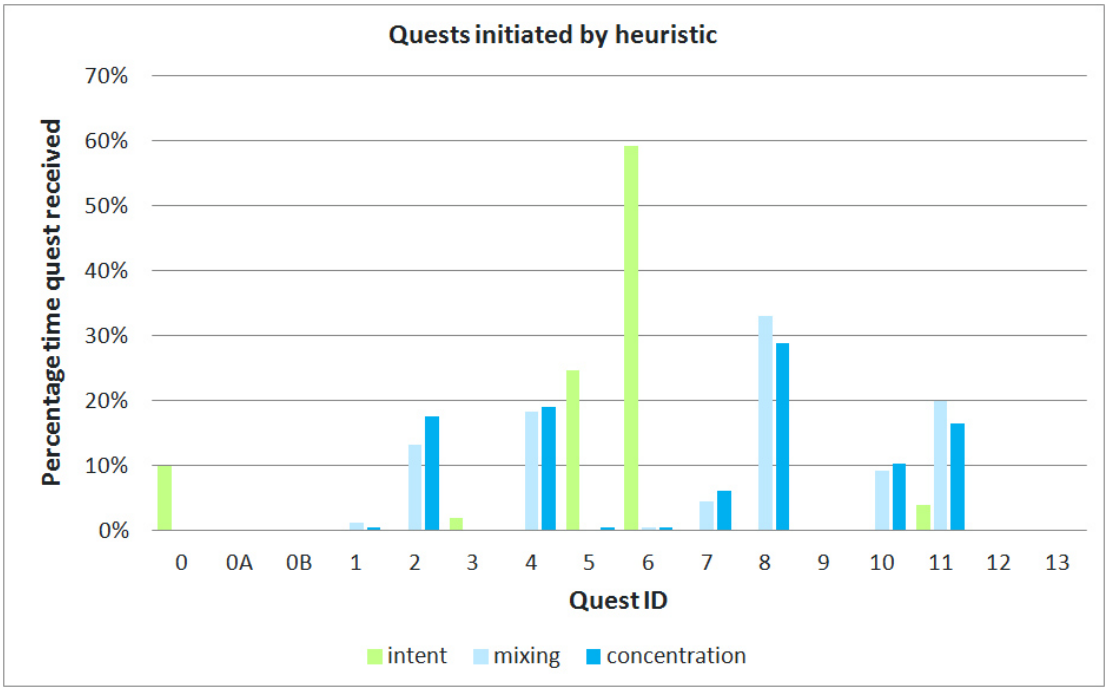

Figure 60: Quests initiated based on quest heuristics.

test runs. This means that the quest mixing and quest concentration had very little chance to come into play.

In contrast, looking at characters' intent showed that there was a high volition for Liz to request the player to convince James to come back (due to her *heartbroken* status and *promiscuous* and *jealous* traits), and a high volition for Violet to request the player to get Thomas out of the way (due to her *domineering, vengeful, irritable* and *dishonest* traits). Again, if more quests were completed, intent would likely not have such strong spikes due to the social landscape changing over time.

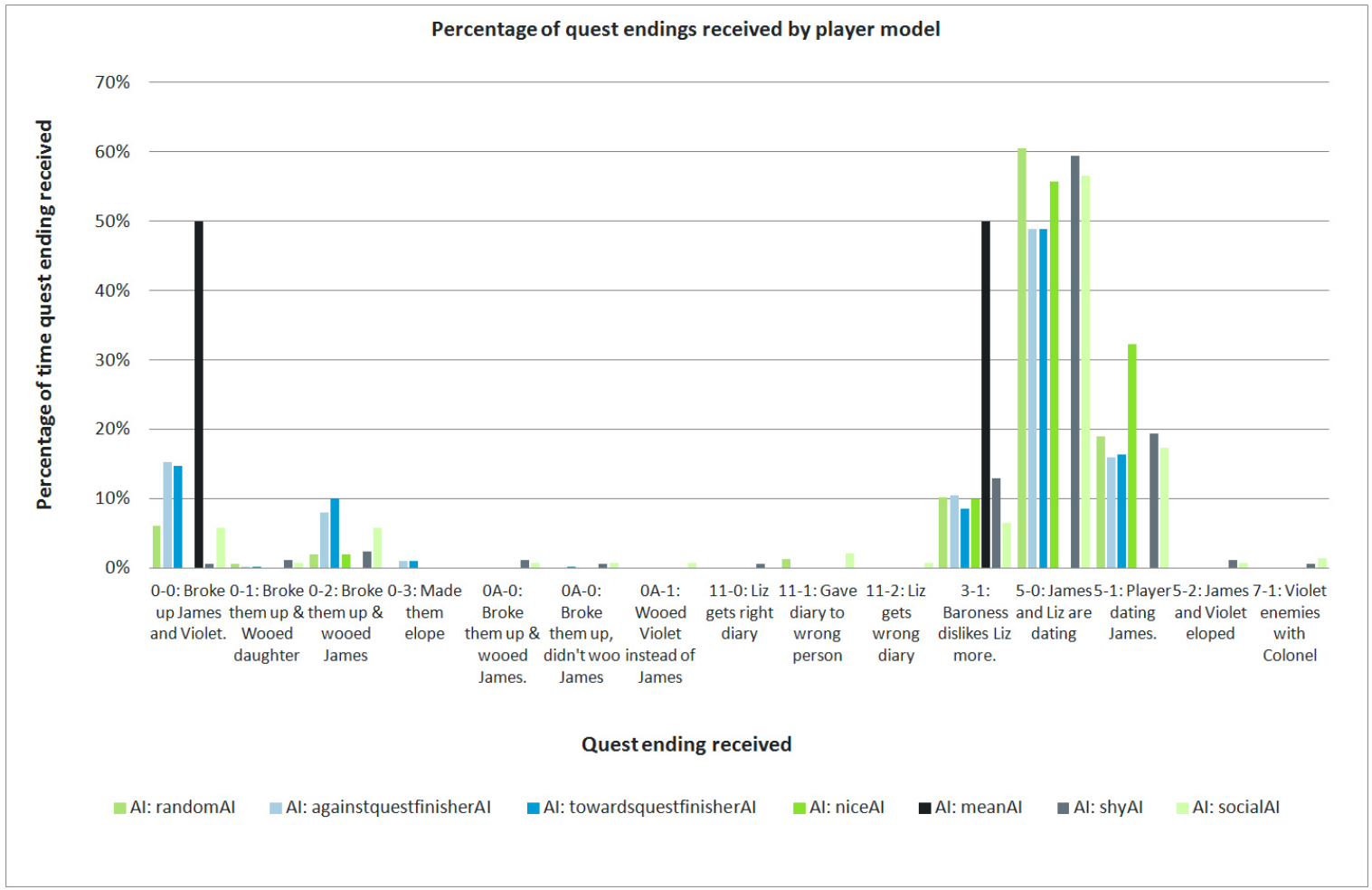

Figure 61: Percentage of quest endings received by player model. The numbers refer to the questID and ending ID.

5.3.5 Quest Endings

Each quest had a number of possible completion states (explained in detail in Chapter 5, Section 1.6). Our hypothesis is that by having different ways that a quest could be completed, with different effects based on the way it was completed, the quest level story would have interesting and meaningful choices, and therefore be playable. Therefore we should see different player models able to complete quests in different ways. As shown in Figure 61, this is the result that we saw.

The meanAI has the most striking spikes, and was able to complete the fewest quests in total. Of the 8 quests completed by the meanAI, the endings were split evenly between *Baroness dislikes Liz more than before* and *Broke up James and Violet*. Both of these quest endings are reached by lowering relationship networks, so it is not surprising that these were the two common endings received. Similarly, the meanAi never got the two most common endings for the other player models: *Player is dating James* and *James and Liz are dating*. Because these require improving relationships, it is unlikely that the meanAI would receive this ending.

The niceAI has a spike for the quest ending in which Liz has asked the player to get James back to her, but instead the player ends up dating James. While this does not seem like a very nice thing to do, when looking at the moves preferred by this player type, it is not very surprising that the player model that likes to improve relationships, regardless of who they

are interacting with, would improve their romance with James to the point that they would end up dating.

The distribution of endings shows that even though most of the player models received the same quests, there is more variation in how the different player models completed the quest. Since each ending has a different effect on the social state of the world, this supports our hypothesis that different player types will have interesting choices with meaningful outcomes.

5.4 Social Moves

At the player story level, *Mismanor* needs to be able to support different preferences in social moves based on player type. Therefore, we should see a different distribution of moves chosen based on our player models. Since CiF-RPG offers social moves based on traits and social state of the world, it is unsurprising that our results show that player models had very different distributions of social moves chosen as seen in Figure 62.

In particular, the niceAI is more likely to use the social moves *Ask Out (3-person), Conversational Flirt (Direct), Encourage,* and *Truce (3-person)*, while the meanAI favors *Discuss (Item), Find Faults,* and *Petty Argument.* The Discuss Item is an anomaly due to the fact that the status *KnownBy* is incorrectly categorized as a negative status, and the meanAI has a high desire to add negative statuses. The questFinisherAI meanwhile prefers *Council/Bad Council* (depending on which variant of the AI is being used)*, Flatter (3-person), Share Feelings* and *Court*. These moves are all linked to quests that the AI is likely to be on;

council to improve the family bond between Thomas and his father, Flatter to improve the friendship between Liz and the Baroness, and Share Feelings and Court to improve romance.

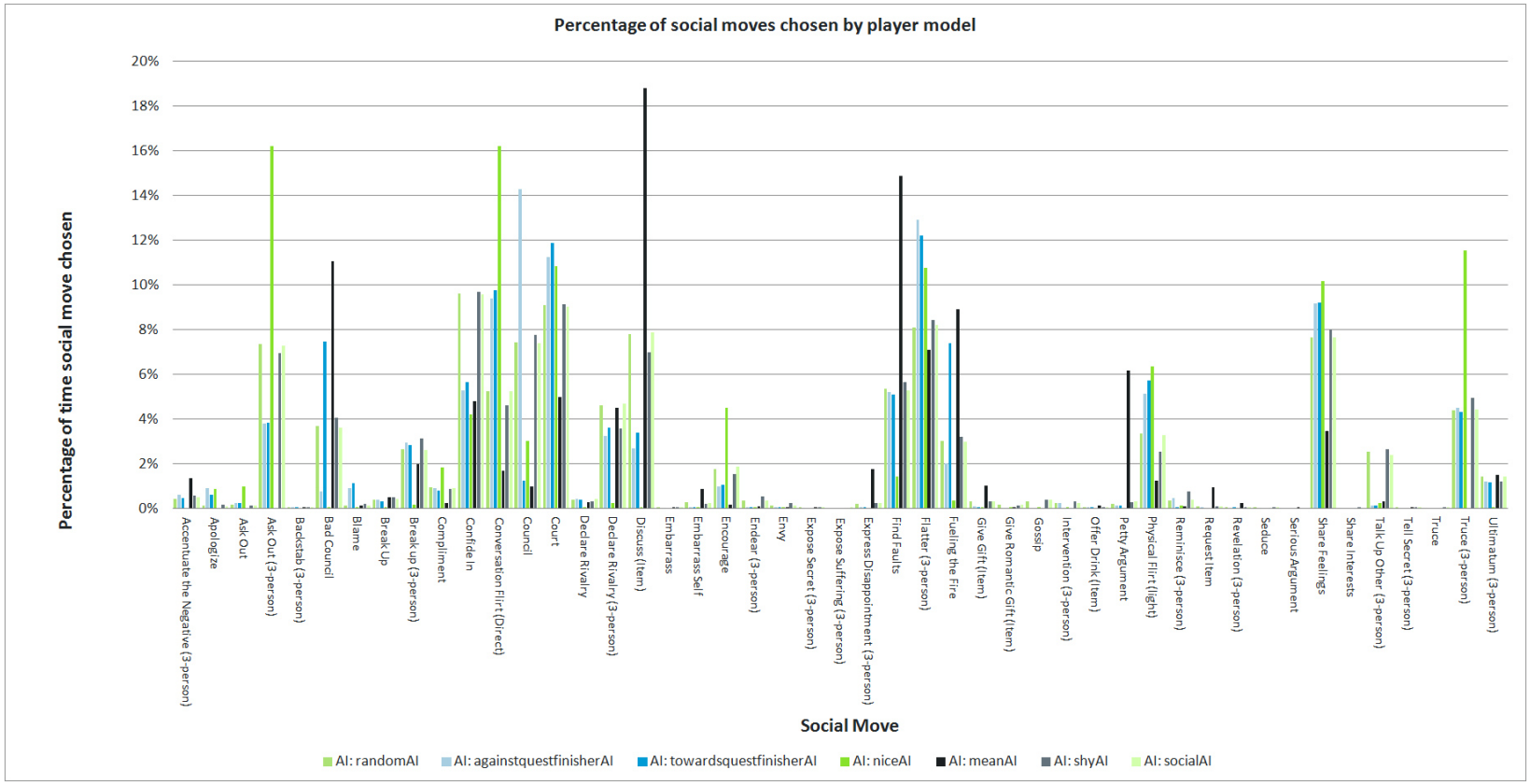

Figure 62: Social moves chosen by player model. This shows a lot of variety in social moves chosen by different player models.

5.5 Discussion

We stated that our hypothesis was twofold:

#1. For story to be playable, there must be a discernible effect on the story for different types of players, unlike other CRPGs in which every player type experiences the same story. Therefore, we should see differences in quest, plot point, and game ending distribution based on the player models favoring different game mechanics.

#2. We use three heuristics to choose plot points and three heuristics to choose quests. Changing these heuristics gives authors some control over the game output. Therefore, when isolating each heuristic, there should be differences in which quests and plot points are favored.

The results we have presented show a discernible effect on quest, plot point and game ending distribution, especially when looking at extreme player types. A mean player will have a different story experience from a nice player, in a way that is consistent with the player's interactions with the story world.

Likewise, a player that is interested in following the quests will have a different experience than that of a player who does not pay attention the quests. While it is more subtle in scope, a shy player who prefers to build a relationship with one character over a player who is more social and interested in talking to many characters will also have a different experience. Overall the shy character sees more complete sub-stories, but fewer plot points in total, as their character is more likely to get the full story out of the character they are

talking to. The shy character also has stronger relationships with fewer characters, which has an effect on the endings and quests received. Meanwhile, a social character sees more plot points overall, but not as many complete sub-stories, and has more relationships and therefore different endings and quests.

In response to our second hypothesis, when we focus on one author heuristic at a time, other than quest mixing versus quest concentration, there is a distinct difference between which quests and plot points are favored, implying that there is authorial control over the outcome of the game.

In the case of quest mixing and quest concentration, because the player models did not complete a large number of quests (32% for the questfinisherAIs which was substantially better than the other player models) it was not possible to test how well these heuristics worked. In general, the questfinisherAI was an improvement over the other player models in terms of completing quests, but there is still much work that could be done to make the AI more robust. While creating the AI logic, it was found that creating an agent that can reliably complete a social quest is a difficult problem, which helps enforce our argument that our system is complex enough that it cannot be solved by a simple AI.

6 Conclusions & Future Work

We have presented the *Mismanor* role-playing game, which employs the CiF-RPG and GrailGM systems to create playable quest and story structures. We discussed the design implications of our desired goal as well as the systems we chose to incorporate. We use the CiF-RPG system for rich social simulations and dynamic social interactions between the

player and in-game characters. We use GrailGM for dynamically choosing quests and plot points based on the player's past actions and the current social state of the game world.

The creation of *Mismanor* led to some insights for future work when creating a social CRPG. In combat, the intent of an action is made obvious to the player (harm, heal, buff, debuff) through color coding, or tool tips. *Mismanor* labels each social action with the intent of the action. However, there are far more intents, with the space to have many more. We limited our intents to increasing or decreasing along each of the four relationship networks (family, romance, trust, friend), losing or gaining a few key statuses (dating, enemies, estrange), or losing or gaining item or knowledge. This is obviously a much more complex set of intentions for actions, and can lead to confusion with the player.

Part of this confusion is that it is not obvious how to get actions for a specific intent type. Most CRPGs use spell bars or action bars to show the player all possible actions, while limitations are made through resource management (mana bar) or time-limited reuse (cool downs on an action). For *Mismanor* there were far more actions available (55) and the player was only shown 4 based on the social state. How to get a specific action to show within that subset of 4 actions was not made clear to the player.

In retrospect, it may have made more sense to go deep rather than broad; that is have fewer actions with more possible effects. With fewer statuses and networks, there would have also been fewer intents, which all would have led to a less overwhelming number of options which could lead to more strategic possibilities.

There is still more work to be done in *Mismanor* towards leveraging the full power of CiF-RPG. Currently we drastically under-utilize the social facts database and cultural knowledgebase portions of CiF-RPG. One possibility is expanding the capabilities of the SFDB to be able to store plot points that have been encountered as well as quest information. This would allow this information to be usable during dialogue exchanges during future social moves. Additionally, *Mismanor* rarely uses information from the CKB, which could be populated with more information for use during social moves.

While *Mismanor* is a step towards playability in story, it is merely an introductory step towards the future of storytelling in CRPGs. Balancing the storytelling desires of the designer with the players' demands for interesting and meaningful choices is an ongoing task. Purely reactive heuristics do not effectively capture all the nuances of hand-authored storytelling. Integrating some of the look-ahead features of declarative optimized drama management into the GrailGM story generation system is one possible direction to improve this.

Additionally, there are many considerations that an author uses to create a storyline. Only a few of these have been considered within the Grail Framework. There are a number of ways to model a narrative arc, and being able to formalize these story methods would greatly improve the capabilities of the system.

CHAPTER 9

FUTURE WORK AND CONCLUSION

This dissertation looks towards the future of computer role-playing games, with a focus on improving story playability. To accomplish this, we broke down story into four levels: *player, quest, game* and *world*. We focused primarily on the player, quest and game levels. Each of these three levels of story contributed to the overall narrative experience in a unique way, and each level has areas that can be improved to strengthen player agency and playability. To address this, we created the Grail Framework, which consists of CiF-RPG, GrailGM, and the authoring tools, Brainstormer and Social Mechanics Design Tool.

CiF-RPG is a system which enables social mechanics within a CRPG. With social mechanics, players are able to move past combat-centric player stories and begin creating social-based stories. GrailGM improves playability at the quest and game level by incorporating the social state of the world (changed by the player through the social mechanics) into the heuristics used to choose quests and plot points offered to the player.

For quests, players are given a high-level goal as opposed to a set of tasks to fulfill to complete the quest. Additionally, the designer can easily specify special goal states that the player might reach that should have a different effect on the social state of the world. This gives the player not only playability within the quest, but of the quest outcome as well.

At the game level, the game endings are based on the state of the world, which gives the player much more control over the outcome of the story. The designer may specify special states which get pre-created story endings, and GrailGM generates endings for states that are not matched.

Because of the dynamic nature of playable story, we created tools to help with ideation and creation of content. The Brainstormer tool uses a common-sense knowledgebase to help designers design quests without needing to fall to standard quest tropes. The Social Mechanics Design Tool (SMDT) is a tool for writers to help create the amount of content required for a game. The SMDT was used for six months to create all the content for *Mismanor* which has given us valuable insights for future tools.

EMPath and *Mismanor* are playable experiences that were created to explore the possibilities of design spaces using Drama Management and the Grail Framework respectively. *EMPath* provided guidance for designing the game level dynamic story selection portion of GrailGM.

Mismanor is the culmination of working towards improving three levels of storytelling, and has helped us move towards our vision of the future of computer role-playing games.

By constructing GrailGM, CiF-RPG, and *Mismanor,* we have created an existence proof for a new approach to story in role-playing games — one with power and flexibility that addresses known problems in the game genre and allows it to move toward experiences that have proven impractical with today's technical and design approaches. From here, there are a few directions that we suggest for future research.

1 Communicating to the player

To effectively make story playable, the player must be able to make an informed choice, which requires communicating current state and possible outcomes of an action to the player.

While the games industry has spent many years fine tuning how to communicate state and progress to the player during combat-based games, there is a void of how to communicate these things during a social-based game. While *The Sims* series (Maxis & The Sims Studio, 2000) has tackled this problem, the user interface ends up being overwhelmed with state graphs, which is a similar problem encountered with *Mismanor*.

Not only does all the information get overwhelming, but it requires metric-based gameplay. It does not intrinsically make sense that a non-player character would agree to date the player character at a romance of 76 but not when romance was 75. The nuances of emotions and feelings seem to require a more subtle method of capturing the information and presenting it to the player.

Giving the player a sense of how their actions will affect the world is equally difficult. It is not a matter of simply showing the player the direct effects of an action they might take, as it is not obvious what effect, for example, +10 romance with a character will have on character's responses or possible actions.

Creating a better visual feedback system so that the player can make strategic choices about which actions to take when is vital to making truly playable stories.

2 Tools

Creating content for dynamic storytelling games is a huge bottleneck. Until natural language generation has solved the AI-complete problem of creating high-quality natural sounding stories, story games are limited by the amount of content an author can create. We looked at some possibilities for future work in tools that help authors with idea generation and with asset creation in Chapter 6. However, there is more work that can be done with AI-assisted tools.

Creating visualization tools for authoring and testing would be invaluable given the number of possible paths through the generated story. One possibility lies in creating a visualization in which the author can easily see the possible story graphs. This can help easily identify orphaned or overly gated plot points. Similarly, a visualization that is able to map the possible paths to various completion states given a game state along with a quest would make it possible to check the emergence of the quest as well as check that it is able to be completed.

3 Conclusion

Currently, there are neither interesting nor meaningful choices for a player to make in most CRPG stories. Instead players are handed a task list for a quest that they must follow the instructions precisely or, at best, choose between a binary choice between the completions associated with one faction/morality or another. At the game level, players are given few options which shape the story world, and these choices are rarely sign posted, leaving the player with no way to make an informed decision.

This makes meaningful role-playing (which one might expect at the heart of a genre called the “role-playing game”) impossible—and a significant experience of agency unlikely except in combat-centric situations. Perhaps, then, it should be no surprise when even a famous CRPG designer such as Rolston quips, “I hate getting quests. I hate the toil of completing quests. I hate the formal and predictable resolution of quests. At best, I feel a Puritan sense of rectitude for laboring dutifully, of doing my duty to uncover the fog of narrative war” (Rolston, 2009).

One alternative—Rolston’s—is to turn away from the structuring mechanism of story in favor of environmental and systems-oriented exploration. But even with human-level intelligence, this approach often leads to a formless experience often associated with a poor DMing. Rather than this, we propose a third alternative for CRPGs, which we have here called *playable* story.

Ambitious computer game designers and authors have already attempted some exploration of playable story. But, given the available tools and the assumptions they reify, the attempts are partial, bug-prone, or both. In response, the Grail Framework was created, which seeks to give designers the tools necessary to create truly playable CRPG stories.

Specifically, our project supports social game mechanics, constructs quests as goal-oriented pursuits that may move in a variety of directions, and structures story as an explorable space appropriate for role-playing different types of characters and influenced by the history of past role-play. Quests and game level story are supported by explicit knowledge representations of elements important to computer and table-top RPGs, including the

evolving knowledge state of the player character. The construction of story is supported by an authoring approach that allows designers to build on our existing packages of rules or extend the system with new rules. It is supported by an open-ended brainstorming tool that can help discover interesting connections between elements of the fictional world and creation tools which eases asset creation.

Together, we hope that these elements of the Grail Framework can help spark a new generation of computer role-playing games, one inspired by the non-combat play made powerful by successful DMs and GMs. Tosca states that "…the success of pen&paper games is precisely in the common creation of a story. […] Paradoxically, this is what cannot be reproduced by computer games" (Tosca, 2003). We feel that while the success of table-top games might not be able to be reproduced exactly without the creation of an AI-complete DM, our system moves CRPGs one step closer towards the depth of story table-top games have enjoyed, while retaining the strengths that have made the CRPG a successful game genre.

REFERENCES

Aarseth, E. (2005). From HUNT THE WUMPUS to EVERQUEST: Introduction to quest theory. *Entertainment Computing-ICEC 2005*, 496–506.

Achterbosch, L., Pierce, R., & Simmons, G. (2007, October 1). Massively multiplayer online role-playing games: the past, present, and future. *Computers in Entertainment (CIE)*, *5*(4), 9.

Actoz Soft. (2006). *La Tale*. Actoz Soft.

Ashmore, C., & Nitsche, M. (2007). The quest in a generated world. *Proc. 2007 Digital Games Research Assoc. Conference: Situated Play* (pp. 503–509). Presented at the DiGRA, Tokyo, Japan.

Association, C. G. (2007). Casual Games Market Report 2007: Business and art of games for everyone.

Atlus. (1996). *Shin Megami Tensei: Persona series*. Atlus.

Barber, H., & Kudenko, D. (2007). Generation of dilemma-based interactive narratives with a changeable story goal. *Proceedings of the 2nd international conference on INtelligent TEchnologies for interactive enterTAINment*, INTETAIN '08 (pp. 6:1–6:10). ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). Retrieved from http://dl.acm.org/citation.cfm?id=1363200.1363208

Bartle, R. (1996). Hearts, clubs, diamonds, spades: Players who suit MUDs. *Journal of MUD research*, *1*(1), 19.

Bates, J. (1992). Virtual reality, art, and entertainment. *Presence: The Journal of Teleoperators and Virtual Environments*, *1*(1), 133–138.

Belman, J., Nissenbaum, H., Flanagan, M., & Diamond, J. (2011). Grow-A-Game: A Tool for Values Conscious Design and Analysis of Digital Games.

Benmergui, D. (2008). *Storyteller*.

Bennett, D., Richmond, C., Reith, S. A., Scott, E., Harding, D., Greenland, J., & project community. (1991). *Discworld MUD*.

Bethesda Game Studios. (2006). *The Elder Scrolls IV: Oblivion*. 2K Games.

Bethesda Game Studios. (2011). *The Elder Scrolls V: Skyrim*. Bethesda Softworks.

Bethesda Softworks. (2002). *The Elder Scrolls III: Morrowind*. Ubi Sot.

Big Fish Games. (2005). *Mystery Case Files series*. Big fish Games.

Big Fish Studios. (2009). *Drawn series*. Big Fish Games.

BioWare. (2000). *Baldur's Gate II: Shadows of Amn*. Black Isle Studios.

BioWare. (2002a). *Aurora Engine Toolset*.

BioWare. (2002b). *Neverwinter Nights*. Infogrames/Atari, MacSoft.

BioWare. (2007). *Mass Effect series*. Microsoft Game Studios & Electronic Arts.

BioWare. (2011a). *Star Wars: The Old Republic*. Electronic Arts, LucasArts.

BioWare. (2011b). *Dragon Age II*. Electronic Arts.

BioWare Edmonton. (2009). *Dragon Age: Origins*. Electronic Arts.

BioWare, & LucasArts. (2003). *Star Wars: Knights of the Old Republic*.

Black Isle Studios. (1999). *Planescape:Torment*. Interplay.

Blizzard Entertainment. (2004). *World of Warcraft*. Blizzard Entertainment.

Blizzard Entertainment. (2008). *World of Warcraft: Wrath of the Lich King*. Blizzard Entertainment.

Blue Tea Games. (2010). *Dark Parables series*. HK studio.

Boal, A. (1993). *Theatre of the Oppressed*. (C. A. McBride, Trans.). Theatre Communications Group.

Bulmahn, J. (2009). *Pathfinder Roleplaying Game: Core Rulebook*. Paizo Publishing, LLC.

Burkinshaw, R. (2009). Alice and Kev. *Alice and Kev*. Retrieved May 5, 2012, from http://aliceandkev.wordpress.com/

Campbell, J. (2008). *The hero with a thousand faces* (Vol. 17). New World Library.

Castronova, E. (2012, January 16). Star Wars: The Dead Republic. *Terra Nova*. Retrieved April 29, 2012, from http://terranova.blogs.com/terra_nova/2012/01/star-wars-the-dead-republic.html

Cavazza, M., Charles, F., & Mead, S. J. (2002). Character-based interactive storytelling. *Intelligent Systems, IEEE*, *17*(4), 17–24.

CCP Games. (2003). *Eve Online*. CCP Games.

CD Projekt. (2007). *The Witcher series*. Atari, Inc.

Chiang, O. (2010, October 8). Blizzard On World Of Warcraft's 12 Million Subscribers, And Its Upcoming MMO Successor - Forbes. *Forbes*. Retrieved from http://www.forbes.com/sites/oliverchiang/2010/10/08/blizzard-on-world-of-warcrafts-12-million-subscribers-and-its-upcoming-mmo-successor/

Cook, D. Z. (1995). *Player's Handbook Advanced Dungeons & Dragons* (2nd ed.). TSR Inc.

Cook, M., Tweet, J., & Williams, S. (2000). *Player's Handbook: Core Rulebook I*. Wizards of the Coast.

Costikyan, G. (2007). Games, storytelling, and breaking the string. In P. Harrigan & N. Wardrip-Fruin (Eds.), *Second Person: Roleplaying and Story in Playable Media* (pp. 5–14). Cambridge: The MIT Press.

Crawford, C. (2004). *Chris Crawford on Interactive Storytelling (New Riders Games)*. New Riders Games.

Crawford, Chris. (2003). *Chris Crawford on Game Design* (1st ed.). Indianapolis: New Riders Games.

Doran, J., & Parberry, I. (2011). A prototype quest generator based on a structural analysis of quests from four MMORPGs. *Proceedings of the 2nd International Workshop on Procedural Content Generation in Games* (p. 1).

eGenesis. (2003). *A Tale in the Desert*. eGenesis.

Eladhari, M. P. (2009). Emotional Attachments for Story Construction in Virtual Game Worlds. Citeseer.

Eladhari, M. P. (2010). The Pataphysic Institute. *Sixth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Fairclough, C. (2004). *Story games and the OPIATE system: Using case-based planning for structuring plots with an expert story director agent and enacting them in a socially simulated game world*. Trinity College Dublin.

Figueiredo, R., Brisson, A., Aylett, R., & Paiva, A. (2008). Emergent Stories Facilitated. *Proceedings of the 1st Joint International Conference on Interactive Digital Storytelling: Interactive Storytelling*, ICIDS '08 (pp. 218–229). Berlin, Heidelberg: Springer-Verlag. doi:10.1007/978-3-540-89454-4_29

Fine, G. A. (2002). *Shared Fantasy: Role Playing Games As Social Worlds*. Chicago: University of Chicago Press.

Four Leaf Studios. (2012). *Katawa Shoujo*. Independent.

Gera, E. (2011, November 15). Why Skyrim is a glimpse into the future of RPGs - VideoGamer.com. *VideoGamer.Com*. Retrieved from http://www.videogamer.com/xbox360/the_elder_scrolls_v_skyrim/features/article/why_skyrim_is_a_glimpse_into_the_future_of_rpgs.html

Grey, J., & Bryson, J. (2011). Procedural quests: A focus for agent interaction in role-playing-games.

Gygax, G., & Arneson, D. (1974). *Dungeons & Dragons: Rules for Fantastic Medieval Role Playing Adventure Game Campaigns: Playable with Paper and Pencil and Miniature Figures*. Tactical Studies Rules (TSR).

Haas, P. (2012, January 16). Star Wars: The Old Republic Review: I'm The Baddest Motherf*cker In The Galaxy. *Gaming Blend*. Retrieved April 29, 2012, from http://www.cinemablend.com/games/Star-Wars-Republic-Review-I-Baddest-Motherf-cker-Galaxy-38671.html

Havasi, C., Speer, R., & Alonso, J. (2007). ConceptNet 3: a flexible, multilingual semantic network for common sense knowledge. *Recent Advances in Natural Language Processing* (pp. 27–29).

Heinsoo, R., Collins, A., & Wyatt, J. (2008). *Dungeons & Dragons Player's Handbook* (4th ed.). Wizards of the Coast.

Her Interactive. (1998). *Nancy Drew Adventure series*. DreamCatcher Games, Her Interactive.

Howard, J. (2008). *Quests: Design, Theory, and History in Games and Narratives*. A K Peters Ltd. Retrieved from http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20&path=ASIN/1568813473

Jackson, S., Punch, S., & Pulver, D. (2004). *GURPS Basic Set: Characters, Fourth Edition* (4th ed.). Bollix Books.

Jelsoft Enterprises, Ltd. (2000, current). Giant in the Playground Forum. *Giant in the Playground Forum*. Retrieved May 5, 2012, from http://www.giantitp.com/forums/

Johnson, L. (2012, January 16). Star Wars: The Old Republic Review. *gamespy*. Retrieved April 29, 2012, from http://pc.gamespy.com/pc/bioware-mmo-project/1215744p1.html

Karlsen, F. (2008). Quests in context: a comparative analysis of discworld and world of warcraft. *Game Studies*, *8*(1), 1–18.

Kolan, N. (2012, January 6). Star Wars: The Old Republic Review. *IGN*. Retrieved April 29, 2012, from http://pc.ign.com/articles/121/1214622p1.html

Lamstein, A., & Mateas, M. (2004). A search-based drama manager. *Proceedings of the AAAI-04 Workshop on Challenges in Game AI*.

Lebowitz, M. (1985). Story-telling as planning and learning. *Poetics*, *14*(6), 483–502.

Long, S. S. (2009). *Hero System 6th Edition Volume I: Character Creation* (Sixth ed.). Hero Games.

Love, C. (2011). *Don't take it personally, babe, it just ain't your story*. Independent.

Lucasfilm Games. (1990). *The Secret of Monkey Island*. LucasArts.

Magerko, B. (2005). Story representation and interactive drama. *Proceedings of the First Annual Conference on Artificial Intelligence and Interactive Digital Entertainment*. Presented at the (AIIDE-05).

Marvelous Interactive. (1996). *Harvest Moon series*. Natsume.

Mateas, M., & Stern, A. (2003). Façade: An experiment in building a fully-realized interactive drama. *Game Developers Conference, Game Design track* (p. 82).

Maxis, & The Sims Studio. (2000). *The Sims series*. Electronic Arts.

McCoy, J., Treanor, M., Samuel, B., Mateas, M., & Wardrip-Fruin, N. (2011). Prom Week: social physics as gameplay. *Proceedings of the 6th International Conference on Foundations of Digital Games* (pp. 319–321). Presented at the FDG, Bordeaux.

McCoy, J., Treanor, M., Samuel, B., Tearse, B., Mateas, M., & Wardrip-Fruin, N. (2010). Authoring game-based interactive narrative using social games and comme il faut. *Proceedings of the 4th International Conference & Festival of the Electronic Literature Organization: Archive & Innovate*. Presented at the ELO, Providence, RI.

McNaughton, M., Cutumisu, M., Szafron, D., Schaeffer, J., Redford, J., & Parker, D. (2004). ScriptEase: Generative design patterns for computer role-playing games. *Proceedings. 19th International Conference on Automated Software Engineering* (pp. 88–99). Presented at the ASE, Linz, Austria.

Mechner, J. (2007). The Sands of Time: Crafting a video game story. In P. Harrigan & N. Wardrip-Fruin (Eds.), *Second Person: Role-Playing and Story in Games and Playable Media* (pp. 111–120). Cambridge: The MIT Press.

Mona, E. (2007). From the Basement to the Basic Set: The Early Years of Dungeons & Dragons. In P. Harrigan & N. Wardrip-Fruin (Eds.), *Second Person: Role-Playing and Story in Games and Playable Media* (pp. 25–30). Cambridge: The MIT Press.

Mott, B. W., & Lester, J. C. (2006). U-director: a decision-theoretic narrative planning architecture for storytelling environments. *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems* (pp. 977–984). Presented at the AAMAS 2006.

Murray, J. (1997). *Hamlet on the Holodeck: The Future of Narrative in Cyberspace*. The Free Press. Retrieved from http://portal.acm.org/citation.cfm?id=572887

Nelson, M. J., & Mateas, M. (2005). Search-based drama management in the interactive fiction Anchorhead. *Proceedings of the First Annual Conference on Artificial Intelligence and Interactive Digital Entertainment*. Presented at the (AIIDE-05).

Nolan, C. (2000). *Memento*. Summit Entertainment.

Oh, G., & Kim, J. (2005). *Effective Quest Design in MMORPG Environment*. Presented at the Game Developer's Conference, San Francisco.

Ontañón, S., Jain, A., Mehta, M., & Ram, A. (2008). Developing a drama management architecture for interactive fiction games. *Interactive Storytelling*, 186–197.

Perren, J., & Gygax, G. (1971). *Chainmail*. Guidon Games and TSR, Inc.

Phanthapanya, A. (2011, October). Is The RPG Genre Dying? *MediaBeast*. Retrieved April 29, 2012, from http://mediabeast.net/2011/10/is-the-rpg-genre-dying/

Pizzi, D., Cavazza, M., & Lugrin, J.-L. (2007). Extending character-based storytelling with awareness and feelings. *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems* (p. 12).

Remo, C. (2008, October 31). A New Galaxy: Daniel Erickson On Writing The Old Republic. *Gamasutra*. Retrieved April 29, 2012, from http://www.gamasutra.com/view/feature/3835/a_new_galaxy_daniel_erickson_on_.php

Roberts, D. L., Nelson, M. J., Isbell, C. L., Mateas, M., & Littman, M. L. (2006). Targeting specific distributions of trajectories in MDPs. *Proceedings of the National Conference on Artificial Intelligence* (Vol. 21, p. 1213).

Rollings, A., & Morris, D. (1999). *Game Architecture and Design: Learn the Best Practices for Game Design and Programming*. Coriolis Group Books.

Rolston, K. (2009). My Story Never Ends. In P. Harrigan & N. Wardrip-Fruin (Eds.), *Third Person: Authoring and Exploring Vast Narratives*. Cambridge: The MIT Press.

Ryan, M. L. (2001). Beyond myth and metaphor: The case of narrative in digital media. *Game Studies*, *1*(1), 1–17.

Salen, K., & Zimmerman, E. (2005). Game design and meaningful play. *Handbook of Computer Game Studies*, 59–79.

Schank, R. (1995). *Tell Me a Story: Narrative and Intelligence*. Northwestern University Press.

Schell, J. (2008). *The Art of Game Design: A Deck of Lenses*. Schell Games.

Sheldon, L. (2004). *Character Development and Storytelling for Games (Game Development Series)*. Premier Press.

Si, M., Marsella, S. C., & Pynadath, D. V. (2005). Thespian: Using multi-agent fitting to craft interactive drama. *Proceedings of the fourth international joint conference on*

Autonomous agents and multiagent systems (pp. 21–28). Presented at the AAMAS 2005, ACM.

Sierra Entertainment. (2008). *King's Quest series*. Sierra Entertainment.

Square Enix. (1987). *Final Fantasy series*.

Sullivan, A., Chen, S., & Mateas, M. (2009). From Abstraction to Reality: Integrating Drama Management into a Playable Game Experience. *Proc. of the AAAI 2009 Spring Symposium*. Presented at the Intelligent Narrative Technologies II, Palo Alto.

Sullivan, A., Grow, A., Chirrick, T., Stokols, M., Wardrip-Fruin, N., & Mateas, M. (2011). Extending CRPGs as an Interactive Storytelling Form. *Interactive Storytelling*, 164–169.

Sullivan, A., Mateas, M., & Wardrip-Fruin, N. (2012). Making Quests Playable: Choices, CRPGs, and the Grail Framework. *Leonardo Electronic Almanac*, *17*(2).

Team, W. (2003). *Dungeons & Dragons Players' Handbook: Core Rulebook I, v. 3.5* (3.5 ed.). Wizards of the Coast.

Thue, D., Bulitko, V., Spetch, M., & Wasylishen, E. (2007). Interactive storytelling: A player modelling approach. *Artificial Intelligence and Interactive Digital Entertainment conference* (pp. 43–48). Presented at the AIIDE, Stanford, CA.

TigerMyth36. (2010, December 3). Is the single player RPG dying? *TigerDroppings.com*. Retrieved from http://www.tigerdroppings.com/rant/display.aspx?sp=23095551&s=2&p=23094966

Tito, G. (2011a, December 28). The State of D&D: Present. *The Escapist*. Retrieved from http://www.escapistmagazine.com/articles/view/features/9293-The-State-of-D-D-Present

Tito, G. (2011b, December 30). The State of Dungeons & Dragons: Future. *The Escapist*. Retrieved from http://www.escapistmagazine.com/articles/view/features/9294-The-State-of-Dungeons-Dragons-Future

Tolkien, J. R. . (1954). *The Lord of the Rings, 3 vols*. London: George Allen & Unwin.

Tomaszewski, Z. (2011). On the Use of Reincorporation in Interactive Drama. *Workshops at the Seventh Artificial Intelligence and Interactive Digital Entertainment Conference*. Presented at the AIIDE, Stanford, CA.

Tosca, S. (2003). The quest problem in computer games. *Proceedings of Technologies for Interactive Digital Storytelling and Entertainment*. Presented at the TIDSE, Darmstadt.

Tronstad, R. (2001). Semiotic and nonsemiotic MUD performance. *1st Conference on Computational Semiotics for Games and New Media 10–12 September 2001 CWI-Amsterdam* (p. 79). Retrieved from http://www.cosignconference.org/downloads/papers/proceedings_cosign_2001.pdf#page=89

Tychsen, A., Tosca, S., & Brolund, T. (2006). Personalizing the player experience in MMORPGs. *Proceedings of the Third international conference on Technologies for Interactive Digital Storytelling and Entertainment*, TIDSE'06 (pp. 253–264). Berlin, Heidelberg: Springer-Verlag. doi:10.1007/11944577_26

Ubisoft Montreal. (2003). *Prince of Persia: Sands of Time*. Ubisoft, SCEJ.

van der Spek, E. D., van Oostendorp, H., & Ch. Meyer, J. (2012). Introducing surprising events can stimulate deep learning in a serious game. *British Journal of Educational Technology*. doi:10.1111/j.1467-8535.2011.01282.x

Vanderwende, L., Kacmarcik, G., Suzuki, H., & Menezes, A. (2005). MindNet: an automatically-created lexical resource. *Proceedings of HLT/EMNLP on Interactive Demonstrations* (pp. 8–9). Retrieved from http://dl.acm.org/citation.cfm?id=1225738

Vash108. (2008, January). Are RPG games dying. *Penny Arcade*. Retrieved from http://forums.penny-arcade.com/discussion/47805/are-rpg-games-dying

Walker, J. (2007). A network of quests in World of Warcraft. In P. Harrigan & N. Wardrip-Fruin (Eds.), *Second Person: Role-Playing and Story in Games and Playable Media* (pp. 307–310). Cambridge: The MIT Press.

Wardrip-Fruin, N. (2009). *Expressive Processing: Digital Fictions, Computer Games, and Software Studies*. MIT Press.

Wardrip-Fruin, N., Mateas, M., Dow, S., & Sali, S. (2009). Agency reconsidered. *Breaking New Ground: Innovation in Games, Play, Practice and Theory. Proceedings of DiGRA 2009*. Presented at the International Conference of Digital Games Research Association, London.

Wedemeyer, M., Felton, R., & LeNeave, J. (2007, current). Obsidian Portal. *Obsidian Portal: Tabletop RPG meets Web 2.0*. Retrieved May 5, 2012, from http://www.obsidianportal.com/

Weyhrauch, P. (1997). *Guiding interactive drama*. Carnegie Mellon University.

Wilson, J. (2009). The Future of Single-Player RPGs from 1UP.com. *1Up.com*. Retrieved April 29, 2012, from http://www.1up.com/features/future-single-player-rpgs

Wright, W. (2007, March). *Will Wright Keynote*. Presented at the SXSW, Austin, TX.

Xagiri. (2008, November 3). “Shovel Basics” quest error. *OGPlanet*. Retrieved from http://nforum.ogplanet.com/latale/forum_posts.asp?TID=73445

Yee, N. (2006). Motivations for play in online games. *CyberPsychology & Behavior*, *9*(6), 772–775.

Young, R. M., & Riedl, M. O. (2003). Towards an architecture for intelligent control of narrative in interactive virtual worlds. *Proceedings of the 8th international conference on Intelligent user interfaces* (pp. 310–312).

Young, S. (2006, 2007). The DM of the Rings. *Twenty Sided*. online comic. Retrieved from http://www.shamusyoung.com/twentysidedtale/?p=1017