The Design of Mismanor: Creating a Playable Quest-Based Story Game

Anne Sullivan, April Grow, Michael Mateas, Noah Wardrip-Fruin
Expressive Intelligence Studio
University of California, Santa Cruz
{anne, agrow, michaelm, nwf} @ soe.ucsc.edu

ABSTRACT

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*. In this paper we present an existence proof for a new approach to CRPG stories that addresses this, while providing details of our implementation. We designed and created a new playable experience, *Mismanor,* to test our theories of playable stories. We discuss the design decisions made as well as the details of the CiF-RPG and GrailGM systems used for complex quest generation and story management based on player's traits and the social state of the game world.

Categories and Subject Descriptors

I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems – *games.*
K.8.0 [**Personal Computing**]: General – *games.*

General Terms

Design, Experimentation, Theory.

Keywords

Role-playing games, quests, story management.

1. INTRODUCTION

Computer role-playing games (CRPGs) often provide a rich game world with well-crafted stories for the player to experience. The world is enhanced by giving the player complex and robust combat systems, allowing for interesting and meaningful choices through player-crafted strategies. Players are often given the option to personalize their character, adding further interest and complexity to the player's combat options. As an example, in a fantasy-based CRPG, a player can choose to play a rogue with high stealth which gives the player different combat options than if they chose a mage with a strong spell-casting ability.

However, this personalization and strategic gameplay does not extend to the stories, which are meant to give meaning to player actions. Narrative is often linear; the player moves through the experience, fulfilling checkpoints to advance the story. The player may be given choices along the way during pre-determined branch points within the narrative; however these choices generally have localized impact, with the overall story arc remaining the same. Character personalization, likewise, has little effect on the story other than perhaps a few word replacements within the dialogue.

CRPGs typically use quests to tie player's actions to the storyline, to give meaning to their actions. While this can provide coherence and believability, it can also lead to the player feeling forced to follow a pre-determined set of actions which may not correspond to their desires for their player character. Unfortunately, because story progression is linked to quest progression in a fixed manner, the player may be left choosing between their desired player character stories or progressing through the game.

Table-top role-playing games, the predecessor to CRPGs, use human game masters (GMs) to sidestep this issue. Story flexibility is provided by the GM adapting and responding to the player's choices, creating new stories and quests as the game progresses, weaving the player's desired story into the overarching game story. This provides the player with interesting and meaningful choices for personalization within the story itself, providing a *playable* story [19].

In part, we are interested in working towards the same type of flexibility in CRPGs, using AI systems to adapt the story through quests based on the player's actions and personalization; to create playable quest-based stories. To address these issues and test the viability of our approach, we have begun creating a playable experience entitled *Mismanor*, a non-combat 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 goal states with different consequences to give players control over the story.

In previous papers we have focused on high-level design [18], character creation, representation and social interactions [17]. In this paper we address the design considerations in creating *Mismanor* along with details for the AI systems CiF-RPG and GrailGM that were required to create playable story through quest structures.

2. RELATED WORK

Other games have been created with similar goals as those we had when creating *Mismanor*. In addition, there is a substantial amount of research related to interactive storytelling. In particular, we are focusing on research relating to narrative generation and quest generation.

2.1 Related Games

Prom Week [12] and *Façade* [11] are games that explore the space of social interactions. *Prom Week* gives the player the ability to choose characters and have them interact with each other, 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 quest-based RPG.

Pataphysic Institute (PI) [7] 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* [8]. 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, and on a narrative built through the quest structure.

2.2 Narrative Generation

There are several related research paths in Narrative Generation. One path is the research conducted on autonomous agents, such as the work of Pizzi, et al. [15]. This work implements NPCs as autonomous agents, such that they react to the player in an interesting and believable manner. The narrative evolves based on the character's interaction with these agents, but there is no guarantee of coherence throughout the story.

Drama Management (DM) systems guide the user towards a more coherent story by adapting the options available to the player based on the player's actions. Both the beat-based DM in *Façade* [11] and the *PaSSAGE* system [21] employ a content-selection model of drama management. In *Façade*, the beat-based DM maintains a probabilistic agenda of dramatic beats. Each beat coordinates autonomous characters in carrying out a bit of dramatic action while supporting player interaction during the beat. Similarly, *PaSSAGE* contains a library of character encounters in a role-playing game, dynamically selecting the next encounter as a function of a model of the player. Thue, et al. [20] expand this work into a delayed authoring system which aims to infer player state to offer a more player-specific experience.

GrailGM, the AI system within *Mismanor* which maintains story coherence, combines these two approaches by offering both semi-autonomous agents and a system of offering options to the player based on their previous actions within the game, but guided by authorial intent. GrailGM is also not necessarily constrained to one model of story "goodness." *Façade* uses Aristotelian dramatic rules to create a story that follows appropriate tension, while *PaSSAGE* and delayed authoring create user models to predict player preference. In GrailGM, the goals of the system are defined by the designer and can follow the rules used by either of these systems, or entirely different rules depending on the author.

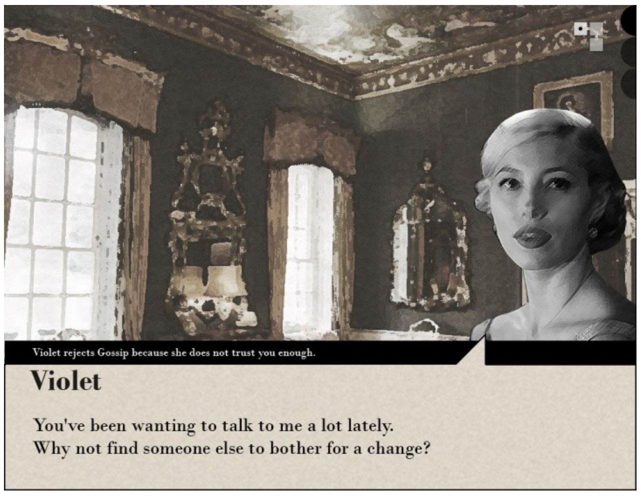
Finally, Peinado and Gervás [14] began work on an interactive storytelling system that models a game master (GM) and player models based on the heuristics set forth by Robin Laws [10]. The system described was never completed to our knowledge, and as such, players were restricted to one of seven pre-made characters which had pre-created player models associated with them. In *Mismanor*, the heuristics used for choosing quests and quest solutions are not built into the system, but are instead specified by the designer.

2.3 Quest Generation

There are currently only a handful of systems working with quest generation. *Charbitat* [1], a game system consisting of generated terrain tiles with randomly placed components based on player actions, was expanded to include lock-and-key style quests based on spatial progression through the world. As the level was generated, a quest could be created on a new tile which used the world state as context for the goal of the quest. Unlike *Mismanor*, quests within *Charbitat* are generated without author input, but are based instead on the tiles the system is generating.

ScriptEase [13] is a designer tool created to work with the *NeverWinter Nights* [4] Aurora toolset [2]. 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 extensible 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. Unlike *Mismanor*, the quests are statically placed and do not change based on the player's actions.

Grey and Bryson [9] suggest an agent-based approach to quest design, with agents able to observe, remember events, and communicate those events with other agents. These events are used to justify quests requested of the player to add believability to the agents. In *Mismanor*, we are interested in agent believability, but we take a more story-centric approach to focus on story coherence.


Violet rejects Gossip because she does not trust you enough.

Violet

You've been wanting to talk to me a lot lately.
Why not find someone else to bother for a change?

Figure 1. Screenshot of *Mismanor*. The player has initiated the *Gossip* social action, which Violet has rejected. The system status messages are shown in the black bar above the dialogue. Status messages are used to detail why an action has been accepted or rejected and why game characters choose specific actions.

3. DESIGN OF MISMANOR

The game *Mismanor* (see Figure 1) is a historical fantasy, set at a manor in the countryside. There are six characters the player may interact with including the Colonel who became estranged from his family while he was at war, his beloved daughter, Violet, who was scarred by the loss of her mother at an early age, and the stable-boy, James, who has been caught up in the family drama from his desire to stay close to Violet. As the player character interacts with the family, it is gradually revealed that some of the game characters are member 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 dialogue exchanges.

3.1 CiF-RPG

Because we are interested in providing an experience of game complexity focused on character relationships instead of fighting mechanics, we chose to use the Comme il Faut (CiF) system [12] to leverage the first-class models of multi-character social interactions. From this system, we created CiF-RPG [17] which supports a player character and treats items and knowledge as first-class objects similar to characters. Where CiF typically models two character social interactions, CiF-RPG models more complex multi-character and object interactions, such as character A asking character B to help them acquire knowledge K from character C.

CiF-RPG includes first-class models of multi-object social interactions — 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 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.

For our purposes, one of the most important aspects of CiF-RPG is the object representation. Each object within CiF-RPG is described by a set of traits and statuses. Traits are static descriptors such as *sentimental*, *unforgiving*, or *secret*. Statuses are transitory, representing temporary states such as *AngryAt*, *empty*, or *false*. Traits and statuses, like the other models within CiF, have micro-theories associated with them. For instance, a character that is unforgiving and heartbroken will be less likely to initiate a positive social action like "Compliment" with the character that caused the heartbreak.

Unlike typical CRPGs in which every character has the same set of descriptors (e.g., strength and dexterity) with differing levels, CiF-RPG gives us the ability to create character descriptions that capture the complexities of personality — and have the character's traits deeply tied to the actions available to the player.

Because objects and knowledge are represented as top-level game objects, as mentioned above, they also support traits and statuses with associated micro-theories and social actions. When CiF-RPG reasons about social actions, it considers all possible objects in a specific role. We created role types such that the system could reason about characters, knowledge, and items in these roles. For example, in the social action *Discuss Secret*, the system considers all combinations of character *A* wanting to discuss knowledge *S* with character *B*. A and B are constrained to character roles (items and knowledge cannot talk) while S is constrained to knowledge with the trait *secret*. For each combination, rules from appropriate micro-theories determine a final score for how much A wants to discuss S with B.

Because physical items are treated as fundamentally the same as other game objects, we created special case social actions that the player can use with items. These actions rely on the traits and statuses of objects to describe the possible interactions the player may take. For instance, an object with the trait *drinkable* and the status *full* allows the player to drink the object. If the object also has the trait *alcoholic*, upon drinking, the player will gain the status *tipsy*. 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 interactions will generally be simpler than that for social interactions, the full power of CiF-RPG is available to support making physical object interactions as context and history dependent as desired.

Another key component of our design is CiF-RPG's model of relationships. CiF-RPG maintains multiple dynamic relationship spectrums between characters which are bi-directional, but not necessarily reciprocal; that is, while Violet and James both have a value representing their level of friendship with one another, James may have stronger feelings of friendship for Violet than vice versa. Since inter-personal feelings are multi-dimensional, CiF-RPG supports a number of different relationships such as friendship, trust, and romance. Again, these are associated with micro-theories; for instance, a character with a high romance level with another character has a positive weighting to ask that character out.

3.2 Quest and Story Design

The choice to use CiF-RPG 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, it was necessary to change the design and focus of our quests. In a traditional CRPG, quest types are focused on combat, movement, and environment manipulation [16]. 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, 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 (discussed in more detail in section 4.3) 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.

3.3 Example

To help illustrate the capabilities of our system, we will look further at the quest in which the Colonel (Douglas) 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 the hidden relationship, the relationship is still active, and that the Colonel has a high level of trust with the player character. 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. Standard implementation approaches such as quest flags and branching if-then trees used in CRPGs such as *Planescape: Torment* [6] and *Dragon Age: Origins* [3] 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. PLAYABLE QUEST PROGRESSION

Given the design choices that were made and discussed in the previous section, it was necessary to modify our game mechanics and AI systems from our original design. In particular, GrailGM was modified to handle the dynamic nature of the quest and story structure within our game.

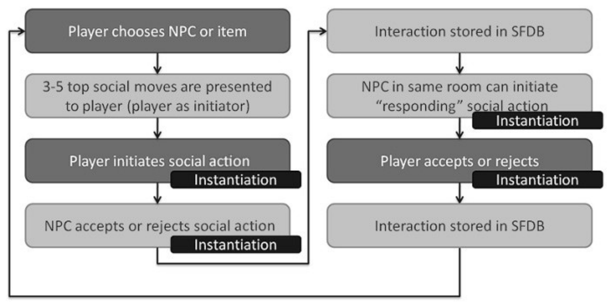

Player chooses NPC or item
3-5 top social moves are presented to player (player as initiator)
Player initiates social action
Instantiation
NPC accepts or rejects social action
Instantiation
Interaction stored in SFDB
NPC in same room can initiate "responding" social action
Instantiation
Player accepts or rejects
Instantiation
Interaction stored in SFDB

Figure 2. The game loop in *Mismanor.* The player character and game characters take turns interacting, and the interactions are stored as history in the social facts database (SFDB).

4.1 Player Actions

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.

As seen in Figure 2, 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 above, 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 three to five 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. 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.

Every action is also stored in a history of actions, called the Social Facts Database (SFDB). Some actions are also tagged with author-specified information such as "romantic action" or "mean action." These can be referenced in future actions to both weight which actions are available or chosen, as well as which effect (the dialogue instantiation and social state change consequences) is played out.

4.2 GrailGM Quest Management

To support our desire for playable quests, we have continued working on the GrailGM system, a run-time game master which uses the current social state and the player's character to dynamically select quests and story plot points. GrailGM supports quests with multiple goal states, allowing the player to choose 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 discernable impact on the narrative.

4.2.1 Quest Types

One of the ways in which we reason about quests is based on the quest type. As mentioned above, the types of quests in CRPGs can be described by the actions required of the player. For instance, *delivery* quests require movement and item manipulation actions, and *kill* quests require combat actions. Because CiF-RPG supports many player actions (particularly social actions) not typically found in RPGs, it was necessary to create new types of quests.

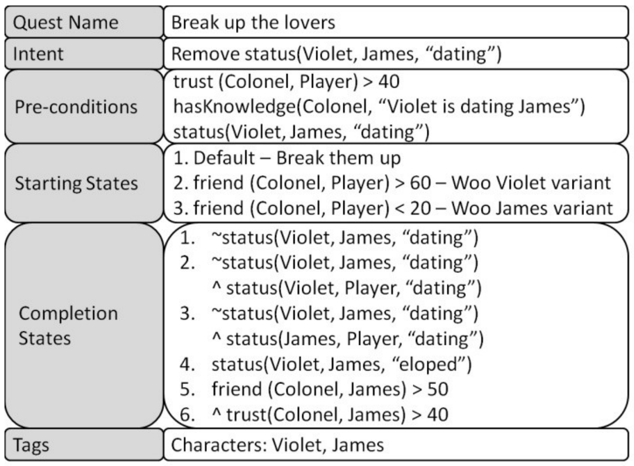
We chose to base the quest types on the intent, or motivation, the NPC would have for giving the quest — similar to the categorization of social actions. The intent of the quest correlates with the change in the world state if the quest is completed according to the default completion state. The new quest types are: *Relationship Up/Relationship Down* (e.g. improve Friendship or decrease Romance), *Status Gain/Status Lose* (e.g. gain Dating, lose AngryAt), and *Knowledge Gain/Knowledge Share* (e.g. learn the cult exists, share secret relationship).

4.2.2 Quest Structure

To support the amount of dynamic content we envisioned, 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. The use of flags is considered simple, but it requires designers to carefully think through all the possibilities of the situation. Additionally, 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 by Wardrip-Fruin [22], even in the award-winning RPG *Star Wars: Knights of the Old Republic* [5], it is possible to complete quests and meet characters in orders not foreseen by the designer, which can lead to inappropriate story moments or quests becoming impossible to complete.

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, and goal states within our quest structure as seen in Figure 3. This allows the designer to describe the desired states without needing to track the various ways to get to that state.

Using our predicate system, we can label each quest with a set of intents, which represent the motivation for an NPC to give that



Quest Name	Break up the lovers
Intent	Remove status(Violet, James, "dating")

Pre-conditions	trust (Colonel, Player) > 40
	hasKnowledge(Colonel, "Violet is dating James")
	status(Violet, James, "dating")

Starting States	1. Default – Break them up 2. friend (Colonel, Player) > 60 – Woo Violet variant 3. friend (Colonel, Player) < 20 – Woo James variant
Completion States	1. ~status(Violet, James, "dating") 2. ~status(Violet, James, "dating") ^ status(Violet, Player, "dating") 3. ~status(Violet, James, "dating") ^ status(James, Player, "dating") 4. status(Violet, James, "eloped") 5. friend (Colonel, James) > 50 6. ^ trust(Colonel, James) > 40
Tags	Characters: Violet, James

Figure 3. The quest structure for the example quest to break up Violet and James.

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.

Because quests are chosen dynamically, it is also possible for some quests to be given by different NPCs. Therefore, each quest can also be associated with a set of pre-conditions for which NPCs can give the quest, as well as the pre-conditions for world state for the quest to be available. For our example, only the Colonel would give the quest to break up Violet and James, so the quest giver preconditions were left out of the diagram for space.

This system allows for author flexibility such that they may later drop in a new NPC without having to tie the NPC explicitly to every quest that is possible for them to give. For instance, if we assign a new NPC a romantic trait and give them a high amount of romance towards another character, they would match the pre-requisites for a quest requesting the player to deliver an item to the character they felt romantic towards.

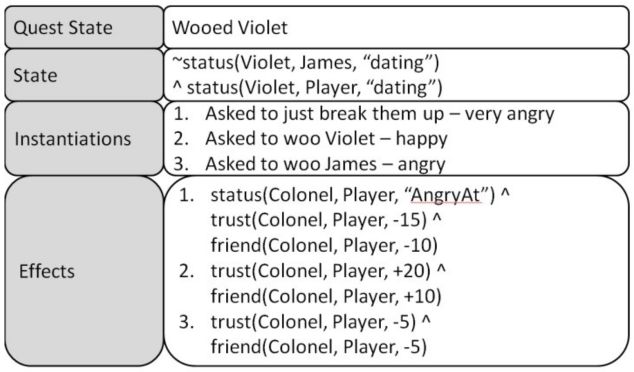
Finally, each quest is given a set of completion states. Each completion state has a set of scenes associated with it (described below) and the description of the social state that matches that completion state. Quests are not required to have multiple completion states, but the addition of them gives the player more flexibility and control over the story.

4.2.3 Scenes

To support this flexibility, each quest has an arbitrary number of scenes associated with it. Scenes are used to describe both quest introductions and quest completions. A scene is comprised of a dialogue exchange between a specified character and the player, as well as a change to the social state of the world. Each scene is associated with a state which describes the necessary game state required to receive that scene.

Upon receiving a quest, the appropriate scene is chosen based on the most complex state that has been matched. Each quest has a default starting scene which has no state pre-requisites. It is important to note that these pre-requisites are in addition to any general quest pre-requisites, so while a quest is not necessarily always available to the player, a starting state must always be available if that particular quest is chosen. An author may then specify other potential starting states. For instance, as discussed above, the player may receive a quest from the Colonel asking the player to break up Violet and James (the default scene). However, if the Colonel really likes the player, they may receive a scene in which the Colonel asks the player to woo Violet away from James. Depending on which scene is selected, the player will see a different dialogue exchange; however all starting scenes lead the player to receive the *OnAQuest* status.

Upon quest completion, the completion scene is chosen based on the most specific state that has been matched. Using the previous quest as an example, there are five possible completion scenes associated with the following states: Violet and James no longer have the dating status, the player and Violet have the dating status and Violet and James do not, the player and James have the dating status and Violet and James do not, Violet and James have the eloped status, and finally the Colonel thinks highly enough of James that he becomes okay with the relationship.



Quest State	Wooed Violet
State	~status(Violet, James, "dating") ^ status(Violet, Player, "dating")
Instantiations	1. Asked to just break them up – very angry 2. Asked to woo Violet – happy 3. Asked to woo James – angry
Effects	1. status(Colonel, Player, "AngryAt") ^ trust(Colonel, Player, -15) ^ friend(Colonel, Player, -10) 2. trust(Colonel, Player, +20) ^ friend(Colonel, Player, +10) 3. trust(Colonel, Player, -5) ^ friend(Colonel, Player, -5)

Figure 4. Example completion state with associated scenes. Instantiations are full dialogue exchanges, but they are shortened here for space. A scene is described by an instantiation and effect pair. Here they are broken out for readability purposes.

Completion scenes have multiple dialogue exchanges and effects — one set for each possible starting scene. If the Colonel asked the player to break up Violet and James as the starting scene, but the player returns having wooed Violet, the Colonel is obviously going to have something different to say than if the player had only broken them up — and the consequences should also be different (instantiation and effect pair #1 in Figure 4).

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 instantiations 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.

4.2.4 Dynamic Quest Selection

In *Mismanor*, a 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.

To implement quest giving in *Mismanor,* we made the act of giving a quest a social action supported by CiF-RPG. We created a social action for giving each subtype of quest, with the intent of the action matched to the quest type's intent, and added these to the CiF-RPG library. For instance, *Give Romance Up Quest* is a social action, as is *Give Lose Dating Quest*.

This allows us to leverage the micro-theories and influence rules we use for other social actions to choose the most salient quest type. The system uses CiF-RPG to reason about the NPC intent and GrailGM to reason about the particular quest to give within that subtype.

When a *give quest* social action is chosen as the NPC's top choice, CiF-RPG sends a request to the GrailGM system to ask for a quest of the appropriate type. Currently, GrailGM checks preconditions and returns a quest of that type. If no quest is available, CiF-RPG chooses the next highest weighted social action for the NPC to enact. We are currently incorporating author-specified story heuristics to give GrailGM additional control over shaping the player experience. For example, a quest involving a powerful item would have a higher weighting during the end game than at the beginning of the game, and a quest following up a piece of knowledge recently received (e.g. in the last 2 actions) would have a high context rating.

4.3 GrailGM Story Management

We are interested in not only giving the player moment-to-moment interaction with the story through dynamic quests, but also allowing the player's actions to have an effect on the global story line as well. To accomplish this, we needed to add a form of story management to GrailGM.

In our system, Knowledge is a first-class object within the system, and plays a large part in tracking both the main story and general information about the game world and its characters.

4.3.1 Plot Points and Knowledge

In *Mismanor* it is necessary to make a distinction between general world knowledge and story knowledge. General world knowledge refers to pieces of information that the player (and other characters) can learn that add some depth and interest to the world. These range from likes and dislikes of the characters to opinions about different locations. The key distinction is that these pieces of knowledge are not necessary to progress the main story in the game. General world knowledge can be used in knowledge-based social moves such as *Discuss Information* and *Tell Secret*.

Plot points are represented as story knowledge. This is a specific type of world knowledge that the game treats slightly differently. Because we are interested in being able to dynamically shape the story using author-level heuristics, 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 more in section 4.3.3) and plot points are revealed during other social moves as dialogue mix-ins. In this way the player has some indirect control over the plot-points (which social actions they choose) but the possibilities are chosen by the AI system.

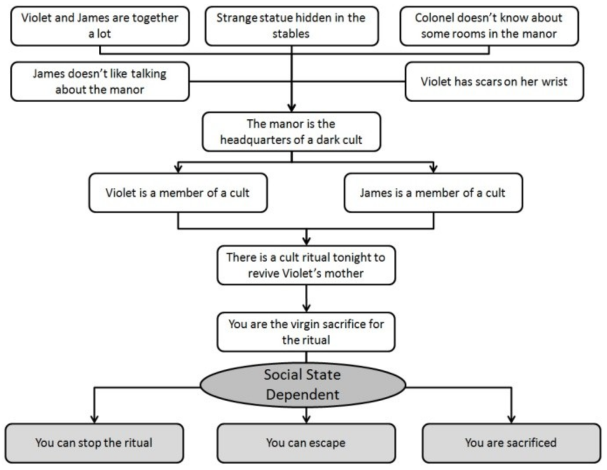

Violet and James are together a lot
Strange statue hidden in the stables
Colonel doesn't know about some rooms in the manor
James doesn't like talking about the manor
Violet has scars on her wrist
The manor is the headquarters of a dark cult
Violet is a member of a cult
James is a member of a cult
There is a cult ritual tonight to revive Violet's mother
You are the virgin sacrifice for the ritual
Social State Dependent
You can stop the ritual
You can escape
You are sacrificed

Figure 5. A portion of our story DAG, showing the plot points associated with the cult storyline, as well as some of the possible endings.

4.3.2 Plot Point Structure

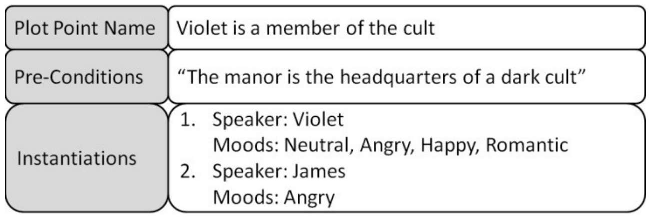
Plot points are specialized forms of knowledge with the addition of pre-conditions and a set of possible instantiations.

Each plot point has a trait specifying that it is a plot point, along with a trait that describes which storyline it belongs to. Our story is broken up into seven storylines; one storyline per non-player character, and one storyline describing the cult. In any playthrough, the player will only see the one full character-based storyline determined by their relationships with the characters.

Plot point preconditions are story-based prerequisites on when a plot point is available. These are represented by the story DAG (partially represented in Figure 5), and are hard-constrained ordering requirements on the story designated by the author at design time. Because we want to enable dynamic choices, it was important that we didn't over-constrain the story space. This was done partially by breaking the story up into the four storylines mentioned above, and also by allowing sets of plot points to be revealed in an unspecified order. For instance, in our story, there are five cult-based plot points that can be learned in any order before unlocking the plot point revealing the cult. Purely optional knowledge was moved out of our story knowledge and represented as game knowledge.

Additionally, plot points have associated instantiations, which are dialogue exchanges that are inserted into social moves based on who the player is talking to and the context of the surrounding dialogue. These *mix-in* instantiations are based on the character speaking, as well as the mood of the conversation that we are trying to match. For instance, if the player is talking to Violet, and they are having an angry exchange, the plot point mix-in would be different than if the player was having a pleasant exchange with James.

Figure 6 shows that it is possible for the player to learn that Violet is a cult member through various means from Violet, but only during an angry exchange with James. The Colonel does not know about the cult, so it is not possible to learn about it through him.



Plot Point Name	Violet is a member of the cult
Pre-Conditions	"The manor is the headquarters of a dark cult"
Instantiations	1. Speaker: Violet Moods: Neutral, Angry, Happy, Romantic 2. Speaker: James Moods: Angry

Figure 6. A sample plot point. The instantiations are full dialogue exchanges, they have been shortened here for space.

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.

4.3.3 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 available for the player. This number was chosen based on the number of plot points a player would see in one playthrough and the number of quests the average player would complete in one game session.

Plot points are only available to uncover while the player is on a quest. This design decision was made to keep the quest and story system tightly coupled, and the game is designed such that the player is almost always on a quest. Back story general world knowledge is available at any time, so the player can still learn about the world regardless of their questing status.

GrailGM then chooses quests based on three tiers of constraints. The first tier looks at the hard constraints, which are the story-level pre-conditions for plot points. These are designated by the story DAG, as described in section 4.3.1. If more than one plot point is available, each plot point is weighted based on author-level preferences. These preferences are stored as a set of heuristics which measure *story cohesion*, *storyline mixing*, and *storyline concentration*.

Story cohesion looks at the graph structure for the story DAG. If there are multiple branches that are available to traverse, story cohesion prefers the branch with the most discovered nodes. This leads to the player uncovering plot points that are highly related to previous plot points.

Storyline mixing and storyline concentration are related, but describe opposite effects. Near the beginning of the game, the authors prefer the player to have a broad mixture of the different storylines. This allows players to choose which character they would like to learn more about, without forcing their hand too early. In contrast, storyline concentration prefers deep storyline. As the game progresses this plays a stronger part, and the player will end up revealing plot points about one particular character.

Finally, plot points and quests are tagged at design time with any associated characters or locations. For instance, the plot point "Violet has a scar on her wrist" is tagged with the character Violet. GrailGM uses the author-level weighted plot points, choosing from the top three based on which plot point has the most shared tags with the active quest. For instance, if the player is currently on the quest to break up James and Violet, a plot point about James or Violet would be chosen over one about the Colonel.

5. CONCLUSIONS AND 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.

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.

We see a couple of interesting extensions to this work. Our goal is to create RPG story experiences with the playability and complexity currently found within combat systems. Combat provides the player with strong feedback in how they are progressing: hit-points track progress and death is used to mark when the player has reached an unbeatable state. However, using a social model, it is not immediately clear how to communicate progress to the player.

We have begun pilot studies which have shown that there is still work to be done this area. In combat, actions are clearly labeled as to the intent of the action (buff, debuff, heal, harm) and the current status of a character is easily shown as a hit point bar. We are planning to borrow from this system, labeling each action with the intent of the action, and showing the current state of relationships and statuses for each character within the GUI.

Additionally, we feel that it is important to create visualization tools for the author, given this is a new method of authoring quests and stories. We are interested in creating a visualization in which the author can easily see a story graph which will help to identify orphaned or overly gated plot points. Similarly, a tool 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.

6. ACKNOWLEDGMENTS

We would like to extend our thanks to Joseph Allington, Tabitha Chirrick, Shawn Hampton, Kevin Meggs, Lauren Scott, Max Stokols, and Vanessa Valencia for their hard work and dedication to this project. This material is based upon work partially supported by the National Science Foundation under Grant No. IIS-0747522.

7. REFERENCES

1 Ashmore, Calvin and Nitsche, Michael. The Quest in a Generated World. In *Situated Play: International Conference of the Digital Games Research Association* (Tokyo, Japan 2007).

2 BIOWARE. *Aurora Toolset*. 2002. ships with NeverWinter Nights.

3 BIOWARE EDMONTON. *Dragon Age: Origions*. PC, Electronic Arts. 2009.

4 BIOWARE. *NeverWinter Nights*. 2002.

5 BIOWARE. *Star Wars: Knights of the Old Republic*. PC, Lucas Arts. 2003.

6 BLACK ISLE STUDIOS. *Planescape: Torment*. PC, Interplay Entertainment. 1999.

7 Eladhari, Mirjam Palasaari and Mateas, Michael. Rules for Role-Play in Virtual Game Worlds - Case Study: The Pataphysic Institute. In *Proceedings of Digital Arts and Culture* (Irvine 2009).

8 Eladhari, Mirjam Palosaari and Sellers, Michael. Good Moods: Outlook, Affect and Mood in Dynemotion and the Mind Module. In *Proceedings of the 2008 Conference on Future Play: Research, Play, Share* (New York 2008), ACM.

9 Grey, John and Bryson, Joanna. Procedural Quests: A Focus for Agent Interaction in Role-Playing Gaes. In *Proceedings of the International Conference on Interactive Storytelling* (Edinburgh 2010).

10 Laws, Robin D. *Robin's Laws of Good Game Mastering*. Steve Jackson Games, 2002.

11 Mateas, Michael and Stern, Andrew. Façade: An Experiment in Building a Fully-Realized Interactive Drama. In *Game Developers Conference, Game Design Track* (San Jose, CA 2003).

12 McCoy, Josh, Treanor, Mike, Samuel, Ben, Tearse, Brandon, Mateas, Michael, and Wardrip-Fruin, Noah. Authoring Game-based Interactive Narrative Using Social Games and Comme il Faut. In *Proceedings of the 4th International Conference & Festival of the Electronic Literature Orginization: Archive & Innovate* (Providence 2010).

13 McNaughton, Matthew, Cutimisu, Maria, Szafron, Duane, Schaeffer, Jonathan, Redford, James, and Parker, Dominique. ScriptEase: Generative Design Patterns for Computer Role-Playing Games. In *International Conference on Automated Software Engineering* (Linz, Austria 2004).

14 Peinado, Federico and Gervás, Pablo. Transferring Game Mastering Laws to Interactive Digital Storytelling. In *International Conference on Technologies for Interactive Digital Storytelling and Entertainment* (Darmstadt, Germany 2004).

15 Pizzi, David, Cavazza, Marc, and Lugrin, Jean-Luc. Extending character-based storytelling with awareness and feelings. In *Proceedings for International Conference on Autonomous Agens and Multiagent Systems* (Honolulu 2007).

16 Sullivan, Anne. Gender-Inclusive Quest Design in Massively Multiplayer Online Role-Playing Games. In *Proceedings of the 4th International Conference on Foundations of Digital Games* (Orlando 2009).

17 Sullivan, Anne, Grow, April, Chirrick, Tabitha, Stokols, Max, Wardrip-Fruin, Noah, and Mateas, Michael. Extending CRPGs as an Interactive Storytelling Form. In *Proceedings for the International Conference on Interactive Digital Storytelling* (Vancouver 2011).

18 Sullivan, Anne, Wardrip-Fruin, Noah, and Mateas, Michael. QuestBrowser: Making Quests Playable with Computer-Assisted Design. In *Proceedings for Digital Arts and Culture* (Irvine 2009).

19 Sullivan, Anne, Wardrip-Fruin, Noah, and Mateas, Michael. Rules of Engagement: Moving Beyond Combat-Based Quests. In *Proceedings of Foundations of Digital Games* (Monterey 2010).

20 Thue, David, Bulitko, Vadim, and Spetch, Marcia. Making Stories Player-Specific: Delayed Authoring in Interactive Storytelling. In *Joint International Conference on Interactive Digital Storytelling* (Erfurt, Germany 2008).

21 Thue, David, Bulitko, Vadim, Spetch, Marcia, and Wasylishen, Eric. Interactive Storytelling: A Player Modelling Approach. In *Artificial Intelligence and Interactive Digital Entertainment* (Palo Alto, CA 2007).

22 Wardrip-Fruin, Noah. *Expressive Processing: Digital Fictions, Computer Games, and Software Studies.* MIT Press, Cambridge, 2009.