A Framework for Coherent Emergent Stories

Gail Carmichael
School of Computer Science
Carleton University
Ottawa, Canada
gail_c@scs.carleton.ca

David Mould
School of Computer Science
Carleton University
Ottawa, Canada
mould@scs.carleton.ca

ABSTRACT

Crafting satisfying narratives while preserving player freedom is a longstanding challenge for computer games. The quest structure used by many games allows players to experience content nonlinearly, but risks creating disjointed stories when side quests only minimally integrate with the main story. We propose a flexible, scene-based emergent story system that reacts to the player's actions while maintaining a reasonable amount of authorial control over the story. Based on the philosophy of story scenes as kernels or satellites, we define a minimal story graph that initially contains mostly disconnected nodes. Over time, the graph is built dynamically from offered to the player. In this paper, we describe the framework of our system and present an early prototype game as a case study. We end with a vision of how our framework could be used to create more coherent, emergent stories in games.

Categories and Subject Descriptors

K.8.0 [**General**]: Games

General Terms

Storytelling in games

Keywords

interactive storytelling, games, agency

1. INTRODUCTION

Effectively integrating stories into games has long been a challenge for game designers and technologists alike. Some games provide a story on rails, where everything is predetermined and players have little to no choice with respect to the story. Others offer a substantial amount of freedom, resulting in a disjointed story with an unsatisfactory lack of direction. Stories in games still do not reliably react to gameplay while maintaining a satisfactory level of structure and player guidance.

Stories seen in traditional media such as films and novels are more dramatically compelling than stories in video games. Games often lack the strong themes and believable characters found in other media, and when they do exist, story systems do not always represent them well [4]. Other media also makes connections between story elements more effectively, giving what Bordwell and Thompson call "a sense that 'everything is there"' [3]. Connecting back to previous events is important in storytelling [11], yet games rarely make an effort to do this apart from the main plot.

Quest systems, used in role-playing and adventure games, partition game activities into small pieces; each quest contains a fragmentary story and provides short-term gameplay goals. Story consistency is enhanced by making the quests self-contained and largely independent. Quest-based games feature open worlds where players are free to navigate a geographical space and choose what quest to do next, thus accessing game content in a nonlinear fashion, but the resulting story tends to be disjointed and even inconsistent [22]. In part, the problems are due to players encountering quests that do not fit with the current story state, and the fact that quests do not purposefully make connections to the story seen so far.

We propose a framework that considers both gameplay and story progression when dynamically building stories for open world roleplaying and adventure games. The framework would ensure that only scenes that fit with both player actions and the current state of the story are available at any given moment. This does somewhat limit the player's freedom in choosing what to do next, but as Swartjes [23] points out, "the player does not *want* complete navigational and expressive freedom per se, but wants to be able to pursue action that is *meaningful*." Our goal is to produce a system that connects to gameplay and offers relevant story choices, but whose content is relatively easy to reason about. The story that results should be coherent as well as satisfying to players. We explore a specific use of our framework in our case study using the story of Jason and the Golden Fleece, described in Section 4. We conclude with a more detailed vision of how our framework could be used as it is further developed in Section 5.

2. BACKGROUND

In this paper, we are concerned specifically with emergent stories. Lindley describes emergent stories as the "emergence of well-defined high level narrative forms from the interaction of smaller scale elements ... in a system that does not contain any representation of that high level form" [12].

A spectrum of approaches to designing such systems spans from character-based to plot-based. Dória et al. [8] define a character-based story as one that "emerges from the real time interaction between autonomous agents, each one with its own objectives" while a plot-based story keeps players on a path that does not stray far from the author's intent.

Façade [14] is character-based. Though story progression statistics help maintain a reasonable story arc, its story is largely organized according to interactions with and reactions from the two non-player characters. *Prom Week* [15], based on the social mechanics engine Comme il Faut, is also character-based. Instead of a set plot, players are presented with a social goal and a selection of social interactions with which to obtain it. Sullivan's *Mismanor* [22], whose engine is based on Comme il Faut, also focuses on social mechanics, but is closer to the plot-based end of the spectrum with its traditional quest structure. Games that feature believable characters acting as autonomous agents, but whose plot is closely overseen by a story director, are found in the middle of the spectrum. Riedl and Stern's *IN-TALE* [17] and Swartjes' *Story Facilitator* [23] are examples. Systems whose organizing principles are based on story elements rather than autonomous agents are on the plot-based end of the spectrum. This includes games with traditional quest structures and is where our framework lies.

Spierling et al.'s [21] four-level architecture for emergent stories spans the entire spectrum and allows for a variable amount of autonomy on each level. The story engine decides which scene to play next or provides a narrative function that the next scene should fulfill. The scene action engine presents the selected scene with a pre-authored or dynamically generated script. The character conversation engine ensures that characters act according to their personality traits through pre-scripted dialog or as intelligent, autonomous agents. Finally, the actor avatar engine takes care of animation, speech, and sound. The core of our framework involves deciding what scenes to offer to players next; this task is found at the level of the story engine. In our vision, chosen scenes would be dynamically modified according to story and game history, similar to what the scene action engine accomplishes. We are not focused on the character-level engines, but Spierling et al.'s system gives insight into how character-based aspects of emergent stories can be incorporated into our framework.

Story graphs are commonly used to represent stories in games. Graphs can be used at various levels of abstraction. Stories can be broken into scenes that are stored in nodes. Our scene nodes can represent scenes at different levels of granularity, but typically a node will contain a single story or gameplay event. Mott and Lester's narrative planner U-Director [16] uses a story graph with individual events as nodes. The planner uses the current state of the game world and the user's beliefs about it to decide what event should be presented to the user next. Bayesian inference algorithms are used in the planning process. Additionally, producing the story networks needed by the planner is an arduous task. Our framework uses simple calculations to prioritize scene nodes and gives authors the ability to reason about each node independently.

Central to our system is the notion that scene nodes can be categorized as either kernels or satellites, terminology adopted from Chatman [5]. Kernels represent plot points and are like a story's skeleton [6]: the story would not survive intact without them. Satellites flesh out the story, develop character, and illustrate theme. Any individual satellite could be skipped without penalty. Game stories typically contain a much lower ratio of satellites to kernels than traditional media [10], suggesting that one way to create more dramatically compelling stories in games is to increase this ratio. Stories created with our framework will contain many more flexible satellites than rigid kernels; the question is how these satellites should be arranged so that the resulting story makes sense.

The model of kernels and satellites is compatible with traditional ways of thinking. "Beads-on-a-string" involve "small areas where there is some freedom of action until some event occurs, at which point a transition to the next bead is opened." [7] The beads-on-a-string model is similar to the kernel structure in that both offer freedom of action in between key events. The use of kernels and satellites also allows for implicit story creation; that is, the ability to specify a model from which a story will dynamically emerge at run-time [21, 23].

The idea of strategically building stories during play contrasts with other approaches that help make game stories more coherent or satisfying. For example, Shelley aims to prevent players from breaking out of a predefined sequence of events rather than build those sequences on the fly [20]. Other systems build dynamic models of gameplay styles to provide content most suitable to the player [25, 24]. We propose building a local graph dynamically by making a subset of suitable nodes available for players to choose from at any given time. Suitability is based on gameplay as well as story progression.

3. OVERVIEW

We propose a framework based on a collection of scene nodes organized into a loosely connected story graph. A scene node, also referred to simply as a node, typically focuses on a single story or gameplay event. For example, a node might reveal a key piece of information about the player's current challenge, or represent the act of acquiring a specific inventory item. By storing gameplay events as scene nodes, the framework can control access to both story and gameplay nodes using the same mechanisms. Some nodes are attached to a specific spatial location or a type of geographic area, such as in the woods or on water. Nodes may also involve choices whose outcomes can affect the story. A player might perform a sacrifice to feed the crew or skip it to appease the gods.

Our framework allows game designers to create coherent emergent stories based on a strong but minimal backbone of kernels and a large grab bag of loosely connected satellites. Kernel scene nodes, also called kernels, represent major plot points. All players must experience a complete path through the kernel structure, ensuring there are no unresolved plot events. Kernels need not be linear; branching is possible. Satellite scene nodes, or satellites, can be used to further the story's themes or characters, always reinforcing the overall story progression. Satellites used for character development will focus on characters that are predetermined by writers, rather than the player-defined protagonist common in RPGs. Satellites do not introduce new themes; rather, they reinforce existing themes that are interwoven through the story. Satellites can also potentially alter the player's understanding of the story without changing the

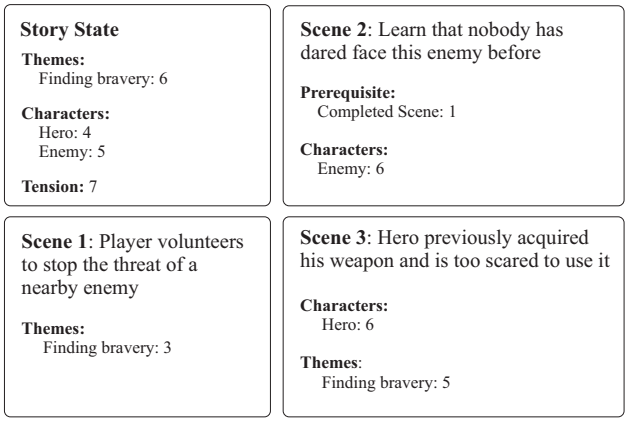

Figure 1: Example story and scene states.

events themselves. While kernels represent rigid plot points, satellites are flexible; they can fit anywhere, and while they do not change things, they can give new insights. Every scene node is either a kernel or a satellite.

A subset of all scene nodes is made available to players at all times, depending on a set of prerequisites and priorities assigned in a process described below. Players can usually choose to consume one of the available nodes, though sometimes they are required to consume one node in particular during a forced encounter.

A key aspect of the framework is its ability to match the current story state with the node choices that become available. Mallon and Webb's empirical research on players of roleplaying and adventure games emphasized the need to remember player actions and provide "calculated, measured consequences, appropriate to circumstance and player motivation" [13]. We ensure the current set of nodes available to players is consistent with the story state by constantly monitoring the story state and using it to recalculate the priorities for scene nodes. How this is done is explained along with other aspects of the framework in the following sections.

3.1 Definitions

Before we proceed with explaining the framework's mechanics, we define the key components of our framework. A *quantifiable story element* (QSE) is a numeric or boolean variable representing some story element in a quantifiable way. Examples include a numeric tension value, the time since a particular theme has been shown, and the contentedness of a particular character. QSEs for tension, character, and theme appear in the sample story and scene states depicted in Figure 1.

The *story state* is defined as a collection of state values – one for each QSE. The story state is used to calculate priority values for scene nodes. Gameplay statistics are not included here, though they can also be used to calculate scene priorities. An example story state is shown in Figure 1. Some QSEs represent *desire* to see a particular element such as a theme: the higher the number is, the longer it has been since a scene featured that theme. The desire is increased after any node is consumed according to a *rate* assigned to each QSE. Other QSEs represent quantities of interest to the designer, such as tension.

A *scene state* is used to tag a scene node with information about how that scene functions within the story. The scene state is a collection of state values representing the QSE relevant to that scene. The higher a value is, the more prominently that element features in the scene. We call this value the QSE's *relevance*. Most scene state values are organized into categories, such as theme and character. These values in a scene state are used when calculating a scene's priority. In Figure 1, Scene 2 is about the enemy, so the enemy character's relevance is high. Scene 3 is about the hero and reflects the theme 'finding bravery,' so these relevance values are high for this scene.

A *prerequisite* is a rule that determines whether a scene should be available at all. Each node can have zero or more prerequisites assigned to it. Scenes that fail any of their prerequisites will not be available to players. For example, Scene 2 will not be available until Scene 1 has been consumed, while Scene 3 may be available before that happens.

A *priority* is assigned to each node so that the nodes with highest priority can be made available to players. A node's priority is calculated as the sum of *modifier* values. A modifier is an aggregate score for a category of QSEs. For example, there will be one modifier for the themes category and one for the character category. The sum of all modifiers relevant to a particular scene state yields the priority for that scene. We discuss how the modifier value is calculated in the next section.

A scene's *outcome* adjusts specified story state values after a scene has been presented. For example, the outcome would cause the story state's tension value to decrease if the scene was intended as comic relief. The outcome also stores a representation of any feedback given to the player after the scene is finished. Feedback may have come after the player makes a choice during the scene. Scene 2 in Figure 1 would reduce the story state's desire for the enemy character, and Scene 3 would reduce the desire values for both the 'finding bravery' theme and the hero character.

3.2 Mechanics

Our framework calculates priorities for scene nodes in real time to ensure that the player is presented with the scenes that best fit with the player's story progression and gameplay up to that point. The overall flow of this process is summarized in Figure 2. The game will first obtain a list of all scene nodes defined by the designer, and then filter these nodes according to whether or not they pass their prerequisites. Priorities are calculated for the nodes that remain by computing a modifier for each category and then summing the category modifiers together. The top-scoring nodes are made available to the player, and the player is free to choose which to consume next.

Prerequisites can be one of several types. Access to a scene can be controlled by a criteria based on:

- the value of a QSE in the story state (for example, tension must exceed 4)
- the value of a variable in the game state (for example, whether an item is in the player's inventory)
- the history of completed scenes (for example, a scene may no longer be available beyond a certain point in the story)

1. Discard nodes that do not meet all prerequisites
2. Calculate scores for remaining scenes
   (a) For each category, compute modifier value for each state value and keep the maximum
   (b) Sum the category modifiers
3. Make nodes with the highest priorities available to players
4. Get input from player and present chosen scene node
5. Update the story state

Figure 2: Summary of the mechanics behind the framework's presentation and consumption of scene nodes.

Progression through the story's plot is governed by prerequisites, especially game state and scene history. In cases where the order of events matters, judicious use of prerequisites can ensure that players do not visit kernels out of order. For satellites, which generally do not advance the plot, fewer global constraints are needed and we can make use of local information to compute priority scores.

Once the prerequisites determine which nodes are potentially available, the nodes' priorities are calculated. Each category of QSEs contributes one modifier to the final priority. The maximum score calculated within a category of QSEs becomes that category's modifier. If a QSE does not appear in the scene state, it is considered to have a value of zero. Within a category, an individual QSE's value is the product of the relevance in the scene state and the corresponding desire in the story state. A QSE that has a high relevance in a scene state as well as a high desire in the story state will potentially lead to a larger modifier value for a given category. This process is summarized in Equation 1:

$$P = \sum_{C} \max_{Q_C} (d_Q \times r_Q), \quad (1)$$

where P is the priority, C represents a set of categories, Q_c is the set of QSEs within category C, and d_Q and r_Q are the desire and relevance for the QSE Q.

For example, in Figure 1, Scene 1 would have a score of 3*6=18 for the 'finding bravery' theme, while Scene 3 would have 5*6=30 for the same element. The current desire to see a scene reflecting the 'finding bravery' theme is strong, and Scene 3 reflects that theme better, so Scene 3's theme modifier is higher. These scores are the maximum for the theme and character categories, and thus become their modifiers. Scene 3 additionally features the hero character with a high relevance value, making its character modifier high as well. After summing the theme and character modifiers together, Scene 3 will have the highest-priority of the three scenes shown, which is desirable given the story state and its need to show particular themes and characters.

Once the priorities for scene nodes have been calculated, the top N nodes are made available to players. In addition, if none of the top N nodes are kernels, we make the highest priority kernel available as well, provided there is at least one kernel that passes all prerequisites. How the choices are illustrated depends on the game: in our case study below, nodes are represented visually as clickable dots on a map. Players should have a choice in which node they wish to consume next, and they should be given some clues as to what might happen if they were to choose a particular node. Alternatively, node priorities can be used to choose the best node for a forced encounter where players may not be given a choice but rather have an event inflicted upon them. Another type of node might be always available to the player as long as it passes its prerequisites and its priority surpasses some threshold.

After a player chooses to consume a node, that node's scene is presented to the player by whatever means makes sense in the game environment. In our case study, event text is presented on the screen. In other games, conversations with non-player characters and other standard gameplay activities are among the possibilities. For some types of games, game statistics may be altered while consuming a scene, for example by shooting an alien fleet or solving a puzzle, even if the story state is not affected. Such changes in the game state may indirectly affect the story since prerequisites can be based on non-story game statistics. It is important to note that once consumption of a node begins, a player is committed to completing it and any challenges contained within. A possible outcome of consuming the node might be to exit early, but this must be explicit and is not the default for all nodes.

One important aspect of our framework is the ability to dynamically connect scene nodes to the overall story so far. Strong stories have a web of interconnections. Events, characters, and objects that are introduced at one point turn out to be relevant again at a later point. Themes recur, and resolutions echo other resolutions [11]. The use of QSEs in calculating priorities for scenes helps ensure that elements like characters and themes appear often enough. Further connections can be added dynamically to the scene chosen by the player. Our vision for how this can be accomplished is discussed in Section 5.

Finally, after a node is consumed, its outcome must be applied. For any scene, certain housekeeping changes have to be made to the story state: the scene consumed must be added to the 'scenes seen' list, and the story state's desire values need to increase according to their corresponding rates. Once this is complete, the specific outcomes defined in the scene are used to further adjust the story state. Usually, values for QSEs are affected. For example, if a theme was particularly relevant in the scene, then that theme's desire will decrease in the story state. In Figure 1, the desire for the enemy character will decrease after Scene 2 is consumed.

3.3 User Interface Issues

Beyond the mechanical underpinnings of the framework, there are several user interface issues to contend with. The first is informing players that there is a scene node available to consume. For our case study, available nodes are displayed explicitly as clickable circles on the screen. *Red Dead Redemption* [18] did something similar in its 3D game environment by displaying a stylized X in locations that trigger a scene or quest to start. Design breakthroughs are likely needed to solve this problem more than technical solutions, partly because each game's individual setting will dictate what is and is not available for use.

After indicating that there is a node to be consumed, the next challenge is giving players an idea of the kind of content

they can expect should they choose to consume it. It can be very frustrating to trigger a scene only to find out that you are embarking on a quest that you cannot quit and are not yet ready for. In our case study, we display a short line of 'teaser text' when hovering over a node to provide a hint about what that node's content is about. We consider this an inelegant approach and hope to find a better one.

In addition to knowing what might happen when a node is consumed, in most cases a player should have a choice in whether or not they want to consume a node. (An exception might be a forced encounter that forces a player to consume an event.) *Red Dead Redemption*'s X accomplishes this, but is not particularly elegant. It also blocks players from moving around a particular area since walking into the X will always trigger an event. Can we avoid blocking a player this way?

Finally, what is the best way to communicate the outcome of a node to players? Relevant banter can be overheard by the player, and statistics can be directly overlaid on the screen (e.g. "Heroism + 4"). *Fallout 3*'s PIP-boy [2] provides a diegetic solution to displaying statistics on a device the player character is carrying, but does not generalize to other settings. The Outsider's Heart in *Dishonored* [1] is an example outside of a science fiction setting, but this technique will become stale if it becomes ubiquitous in games. While we have implemented simple solutions to these issues in our own case study, we consider them to be open problems for our framework and beyond.

4. CASE STUDY

We based our game story on the Greek myth of Jason and the Golden Fleece. We chose this story for several reasons. It is a colorful story with many interesting events and characters. It is structured as a quest, as are many games, so lessons from this example will carry over to other game stories. It is broken up into many episodes, with many minor occurrences that can be neglected in any particular telling, so it fits naturally into our kernel-satellite model. Finally, the events of the story are distributed through space, so placing events on the map is a natural way of communicating options to the player.

Our version of Jason and the Golden Fleece is heavily influenced by Graves's telling [9]. In the Greece of Graves, the gods are a constant presence, and propitiating the gods is one of the main tasks of the Argonauts. Graves's Argonauts are a quarrelsome lot, and one of Jason's chief challenges is enforcing order and settling their squabbles. There are also external hazards, such as storms, disease, giants, unfriendly locals, and the terrible clashing rocks called the Symplegades.

Jason is the nephew of Pelias, the king of Iolcos. Jason is rightfully the king, but Pelias conceives a scheme to rid himself of the troublemaker: he proposes that Jason fetch the fabled Golden Fleece from the distant kingdom of Colchis. Jason accepts the challenge and raises a crew of heroes, the Argonauts, to sail the Argo to Colchis and retrieve the Fleece. We take up the story as the Argo is preparing to set sail from Iolcos. In the role of Jason, the player must manage the happiness and health of the crew, the favor of the gods, and win past the many dangers that await on the long journey. In the complete story, Jason and the Argonauts secure the fleece and return to Iolcos, accompanied by Medea, princess of Colchis. Medea tricks Pelias's daughters into killing him, whereupon Jason becomes king of Iolcos.

Themes	Characters	
Heroism	Argonauts	Hermes
Treachery and honor	Ephemus	Zeus
Will of the gods	Hercules	Aphrodite
	Atalanta	Athena
	Orpheus	

Table 1: Modifier categories of quantifiable story elements in the Jason and the Golden Fleece case study.

Crew morale	Ephemus Happiness	Hermes Happiness
Jason's Heroism	Hercules Happiness	Zeus Happiness
Tension	Atalanta Happiness	Aphrodite Happiness
	Orpheus Happiness	Athena Happiness

Table 2: QSEs not associated with a modifier category.

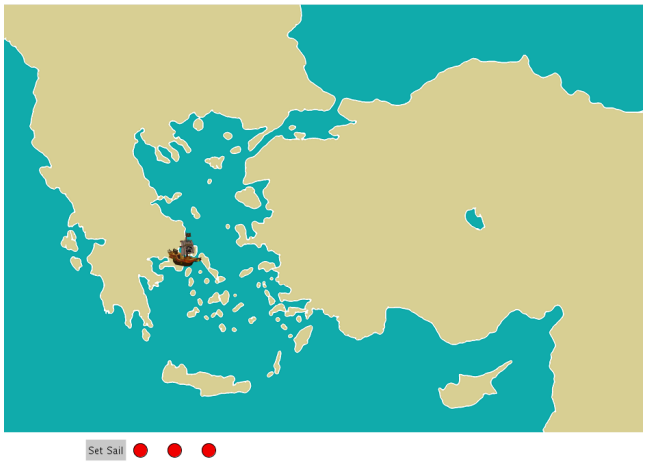

Figure 3: A screenshot from the beginning of the Jason and the Golden Fleece game prototype. The player is contemplating setting sail.

Our game prototype currently covers the story from the beginning of the journey to the passing of the Symplegades. The player controls a ship indirectly by clicking on the next location Jason and the Argonauts should travel to. A new set of nodes appears on or below the map any time the ship changes location or a node is consumed. Nodes below the map are either not tied to a specific physical location or represent events that occur when the ship is not sailing. Tables 1 and 2 summarize the quantifiable story elements used in the game. Table 1 shows the two categories, themes and characters, that are used in regular priority calculations. Table 2 shows remaining QSEs not belonging to any category; these are used for prerequisites, occasional forced encounters, and, in rare cases, adjusting priority calculations.

Currently, the four nodes with the highest priority are displayed. The number four has been chosen arbitrarily for this prototype, but can easily be changed in code. We envision more variability in the number of nodes available throughout the game: when there are many good choices, we should

offer the player more, but at some points in the story the player's choices should narrow down to a fateful choice. The player can hover over a node to get a clue as to what content they will encounter should they choose to consume that node. To consume a node, a player simply clicks it. If the node is below the map, it is consumed immediately; if it is on the map, the ship must first sail to it, allowing opportunity to cancel by sailing elsewhere or choosing another node.

We can see a snapshot of the early game in Figure 3. Some satellites are available as the Argonauts prepare to set sail; these introduce or reinforce the themes of the game, provide background and characterization of some of the Argonauts, and allow the player to manage tradeoffs between crew harmony, Jason's heroism, and the favor of the gods. For example, the Argonauts can prepare a sacrifice to Zeus, increasing his favor, but incurring delay at which the Argonauts chafe. Alternatively, in a scene adapted from Graves, Jason can offer to renounce his captainship of the Argo in favor of Hercules; this pleases the Argonauts, who resented serving under the untried Jason. Hercules refuses to become captain, warning Jason that he must take command of his own destiny; this lightly touches on the recurring theme of the nature of heroism. Each node visited will alter the story state. Eventually the player will decide to set sail, activating the first kernel and beginning the long journey to Colchis.

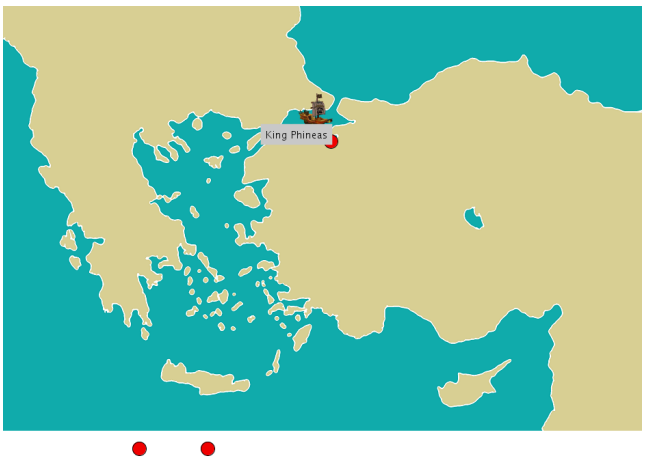

Figure 4: A screenshot from later on in the Jason and the Golden Fleece game. The player is considering clicking on the node about King Phineas rather than the neighboring node or nodes along the bottom.

Later in the game, as the Argo enters the Marmara Sea, we reach a critical period of the journey: the Argonauts approach the Symplegades, huge rocks that slam together to crush passing ships. Various answers to the peril are available but must be found, for example, by winning the favor of Athena, or by rescuing King Phineas from the harpies so that he can tell them the secret of how to cross. The latter was the solution adopted by the Argonauts of myth. Story nodes are scattered along the coast of the Marmara Sea; two are available, a visit to the realm of King Phineas and a trip to the land of the Dolionians where the Argonauts can participate in the violent and mirthful Dolionian Games. In addition, there are many potential nodes not linked to any particular location, including nodes relating to individual Argonauts, omens from the gods, and incidental hazards such as storms. In the figure, nodes relating to the theme of "treachery and honor" are available. The other themes have played a part in recently encountered nodes, and hence the story's desire to revisit the other two themes is lower; nodes about these themes, the individual heroes, and the gods will become available in the future as those desires increase. The Argo's passage through the Symplegades is the next kernel, but it is not available yet because none of its prerequisites have been met.

As we develop the prototype further, we will incorporate gameplay activities such as managing the contentedness of the Argonauts and gods as well as maintaining a minimum level of provisions. Gameplay choices made by players will have both mechanical consequences (the ship will sail more slowly) and story consequences (the player may be forced to seek out scene nodes that allow them to re-provision to avoid the ship sinking). It is important to us that gameplay and story are tightly intertwined, with one affecting the other. Game statistics will affect what nodes are prioritized, and the nodes chosen by the player will in turn affect the game statistics.

5. VISION

The system as presently conceived already supports the typical questing story model of current open-world CRPGs: a quest would be a satellite node, or a series of nodes with prerequisites, and the 'main story' plotline would be a series of kernels. Branching based on story parameters, such as the Paragon/Renegade meters in the *Mass Effect* series, or Karma in the *Fallout* series, is also possible.

However, while traditional quest structures do little to enforce the overall unity of the story, our framework encourages authors to consider how each quest ties into broader themes and other story elements and to explicitly tag the nodes according to which story elements are served. By making nodes available according to the need to reinforce specific quantifiable story elements, as controlled by the rate parameters associated with each element, we can ensure that no element is neglected and a sensible global balance is maintained. If it has been some time since the character Orpheus was visible in the story, nodes involving Orpheus will become more prominent. We hope writers will be encouraged to look for opportunities to reflect story elements in the scene nodes, enriching the story without mechanics.

In addition, our framework allows authors to control the story's progression. Prerequisites can be used to ensure that kernel nodes become available only when appropriate. Prerequisites could be based on the history of the story or on game statistics such as items obtained, battles fought, and goals accomplished. Prerequisites might also dictate that a kernel becomes available after a certain number of satellites have been seen, and measures of tension can help ensure an effective interest curve (as described by Schell [19]) is maintained. Spatial barriers can be used either in a literal way within the game world, or again as prerequisites for kernels. The player's proximity to a node's location would factor into that node's availability. Distance might be measured as the crow flies, or at a more conceptual level, say distance measured in train stops.

Large games are created by large teams of people working together. Managing large coherent stories is difficult, but breaking it down into individual kernels and satellites will

ameliorate the difficulties. Designers can reason about a single node and decide under what conditions the node should be available.

In the future, we envision even more dynamic changes to the storyline by making slight adjustments to the content of a node depending on the story state. As mentioned in Section 3.2, we hope to make connections to prior player decisions through subtle changes to the scene, for example, using visual motifs and banters, changing the lighting, and adjusting the timing of dialogue. The player's earlier refusal to help Baron Wulfston could be echoed in the story's conclusion by having the camera linger on the baron's empty chair. Visual motifs already present in the scenes could be opportunistically emphasized as the emerging story warrants. Even if many players fail to pick up on these nuances, there would be a sense of significance heightening the tension; there would also be something to reward more observant players. Also, the more interconnections that are included, the greater the chance that at least some of them would be noticed. The ideas behind Spierling et al.'s character conversation and actor avatar engines [21] may also be used in conjunction with the discussed dynamic changes to produce a strong set of believable characters that react appropriately to the story's history. The character conversation engine ensures that "characters act according to their personality traits, social roles and relationships," giving the opportunity to have characters act according to story state. The actor avatar engine takes care of animation; its level of autonomy determines how well the output reacts to the current story context. Here, the visuals rather than character decisions can reflect the story state.

As well as deliberately authoring connections between story elements, repetition of motifs and themes throughout the game will result in serendipitous linkages within any particular play-through. By strengthening the fabric of the story and revisiting the same themes many times, patterns will emerge, recognizable by the player. A story, in a game or otherwise, comes alive most strongly when it is able to enlist the imagination of the audience to flesh out details not explicitly presented. Ultimately, we would want to make it possible for the player to communicate these observations back to the game, so they can then be further elaborated in later play; as yet we do not have concrete notions of how this might be achieved, but our vision includes the player as an author, not only controlling an avatar in the game, but making suggestions about the nature of the story itself.

It is worth noting that when we talk about using our framework to prioritize nodes that feature characters not reinforced recently, we are referring to characters whose attributes have already been set by a designer. Many roleplaying games give players a blank slate for their characters, allowing them to fill in the details as they play. In these types of games, other non-player characters can still be revealed and reinforced periodically with our framework. Even if the player character begins as a blank slate, we envision locking the player into a certain path when they choose enough nodes that reflect a particular character trait. For example, if someone playing as a princess chooses enough violent nodes she will no longer be able to dissolve the threat to the kingdom through peaceful measures.

Although we have proposed a particular scoring method for calculating priorities, more testing is needed. An open question is whether our simple approach will be sufficient. Currently, we use a greedy approach; use of heuristics or global optimization may be more effective. In addition, experimentation with the desirability rate is required. Presently fixed, it might make sense to make the rate variable. The changes to the desirability rate can be connected to the outcome of another story node or physical location. For example, the theme of fire and ash should perhaps be more prominent if the player enters Mordor, while hopefulness might not be featured at all. Or perhaps after the prince's death, he is revealed to be a betrayer, causing the betrayal theme's rate to increase dramatically.

6. CONCLUSION

We have introduced a framework for coherent emergent stories and explored an implementation in a case study based on the story of Jason and the Golden Fleece. Our framework, based on the idea of rigid kernels and flexible satellites, uses a prerequisite and story-based priority system to ensure that scenes offered to players make sense both in terms of the story's progression and the player's gameplay history. Our prototype game allows players to bring Jason and his crew to the Symplegades in a flexible manner; we will continue to flesh out the story and further explore the theme of honor and treachery with Medea's story on the return voyage. We hope that with frameworks like ours, designers will be better able to create dramatically compelling stories for games that are coherent but still incorporate player actions and choices.

7. REFERENCES

[1] Arkane Studios. Dishonored (PlayStation 3), October 2012. Bethesda Softworks.

[2] Bethesda Game Studios. Fallout 3 (Windows), 2008. Bethesda Softworks.

[3] David Bordwell and Kristin Thompson. *Film Art: An Introduction*. McGraw-Hill Humanities/Social Sciences/Languages, 10 edition, 2012.

[4] Selmer Bringsjord. Is it possible to build dramatically compelling interactive digital entertainment (in the form, e.g., of computer games)? *Game Studies*, 1(1), 2001.

[5] Seymour Chatman. *Story and Discourse: Narrative Structure in Fiction and Film*. Cornell University Press, 1980.

[6] Steven Cohan and Linda M. Shires. *Telling Stories: A Theoretical Analysis of Narrative Fiction*. Routledge, 1988.

[7] Greg Costikyan. Games, storytelling, and breaking the string. *Electronic Book Review*, 2007.

[8] Thiago R. Dória, Angelo E.M. Ciarlini, and Alexandre Andreatta. A nondeterministic model for controlling the dramatization of interactive stories. In *Proceedings of the 2Nd ACM International Workshop on Story Representation, Mechanism and Context*, SRMC '08, pages 21–26, New York, NY, USA, 2008. ACM.

[9] Robert Graves. *The golden fleece*. Cassell and company ltd, 1944.

[10] Barry Ip. Narrative structures in computer and video games: Part 1: Context, definitions, and initial findings. *Games and Culture*, 6(2):103–134, 2011.

[11] Keith Johnstone. *Impro: Improvisation and the Theatre*. Routledge, 1987.

[12] Craig Lindley. *Developing Interactive Narrative Content: sagas/sagasnet reader*, chapter Story and Narrative Structures in Computer Games. High Text Verlag, 2005.

[13] Bride Mallon and Brian Webb. Stand up and take your place: identifying narrative elements in narrative adventure and role-play games. *Computers in Entertainment (CIE)*, 3:1–20, 2005.

[14] Michael Mateas and Andrew Stern. Structuring content in the facade interactive drama architecture. In *Proceedings of Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2005)*, 2005.

[15] Josh McCoy, Mike Treanor, Ben Samuel, Aaron A. Reed, Michael Mateas, and Noah Wardrip-Fruin. Prom week: Designing past the game/story dilemma. In *Proceedings of Foundations of Digital Games 2013*, 2013.

[16] Bradford W. Mott and James C. Lester. U-director: A decision-theoretic narrative planning architecture for storytelling environments. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '06, pages 977–984, New York, NY, USA, 2006. ACM.

[17] Mark O. Riedl and Andrew Stern. Believable agents and intelligent story adaptation for interactive storytelling. In Stefan Göbel, Rainer Malkewitz, and Ido Iurgel, editors, *Technologies for Interactive Digital Storytelling and Entertainment*, volume 4326 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin Heidelberg, 2006.

[18] Rockstar San Diego. Red Dead Redemption (Xbox), May 2010. Rockstar Games.

[19] Jesse Schell. *The Art of Game Design: A book of lenses*. Morgan Kaufmann, 2008.

[20] Matthew Shelley. On the feasibility of using use case maps for the prevention of sequence breaking in video games. Master's thesis, Carleton University, 2013.

[21] Ulrike Spierling, Dieter Grasbon, Norbert Braun, and Ido Iurgel. Setting the scene: playing digital director in interactive storytelling and creation. *Computers & Graphics*, 26(1):31 – 44, 2002.

[22] Anne Sullivan. *The Grail Framework: Making Stories Playable on Three Levels in CRPGs*. PhD thesis, University of California Santa Cruz, 2012.

[23] Ivo Martinus Theodorus Swartjes. *Whose story is it anyway? : How improv informs agency and authorship of emergent narrative*. PhD thesis, University of Twente, 2010.

[24] David Thue, Vadim Bulitko, Marcia Spetch, and Trevon Romanuik. A computational model of perceived agency in video games. In *Proceedings of the Seventh AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2011.

[25] David Thue, Vadim Bulitko, Marcia Spetch, and Eric Wasylishen. Interactive storytelling: A player modelling approach. In *The Third Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, 2007.