PRISM: A Framework for Authoring Interactive Narratives

Yun-Gyung Cheong, Yeo-Jin Kim, Wook-Hee Min,
Eok-Soo Shim, and Jin-Young Kim

Graphics&OS Group, Software Labs, SAIT, CTO, Samsung Electronics CO., LTD.
Giheung-Gu, Gyeonggi-Do, South Korea
{yuna.cheong,yeoj.kim,wookhee.min,esp.shim,
claire.j.kim}@Samsung.com

Abstract. The advances in computing technologies enable the computer users to create and share their own stories to the community at large. However, it is still regarded as complicated and laborious to author interactive narratives, where a story adapts as the user interacts with it. In authoring interactive narratives, two main approaches—branching graphs and AI planning—have been significantly used to augment interactivity into conventional linear narratives. Although each approach offers its own possibilities and limitations, few efforts have been made to blend these approaches. This paper describes a framework for authoring interactive narratives that employs an adapted branching narrative structure that also uses planning formalism to enable automated association between nodes. We expect that our work is valuable for non-expert users as well as AI researchers in interactive storytelling who need to create a large quantity of story contents for varied endings for a story.

Keywords: Interactive digital storytelling, authoring tool, interactive contents, story representation, branching narrative, story graph, AI planning.

1 Introduction

The advances in computing technologies enable the computer users to create and share their own stories to the community at large. This is evidenced by the significant success of animated movies and UCCs on the Internet, and the emergence of Machinima—the production technique of computer-generated animated movies using 3D game engine. Therefore, UCC providers such as YouTube supply video authoring tools for their users. However, it is still regarded as complicated and laborious to write interactive narratives, where a story adapts as the user interacts with it. Unlike conventional narrative systems in which one single story is told linearly without intervention from the audience, interactive narrative environments need to arrange multiple plots and unfold one of them as the audience makes a series of choices. For this, interactive story systems require an effective methodology to represent a collection of plots integrated with the audience's choices.

While a number of techniques for authoring interactive narrative generation have been presented by AI researchers and Game developers [1; 3; 4; 7; 9; 11; 12; 16; 21],

U. Spierling and N. Szilas (Eds.): ICIDS 2008, LNCS 5334, pp. 297–308, 2008.

this paper investigates two main approaches to interactive story representation: branching narratives and plan structures. A branching narrative often takes the form of a directed graph containing nodes and arcs between the nodes; a node denotes a series of scripted scenes, and an arc denotes transition from one node to another.

The branching narratives have been favored in a number of practical interactive story systems due to its compelling advantages—intuitiveness and high degree of expressiveness [5; 16; 17; 18]. The interactive digital storytelling system *Scenejo* [16] provides an authoring tool that enables the author to construct a transition graph to represent plot progress where plot points are reached thorough dialogue exchanges between multiple agents. In the plot graph in Scenejo, a scene is represented as a knowledge base that describes the participating agents' utterances using a pattern matching technique. A transitional condition that advances the story to the next scene in Scenejo falls in three categories: elapsed time, a user utterance, or a state change. As exemplified in [16], guessing a right password may result in a scene change.

However, it has been also noted that an interactive story system employing a branching technique which needs frequent interaction with the user would face explosion of conditional branches. In addition, a branching narrative is typically built at the design time and thus its interactivity with the audience is limited by the story author's anticipation. As a result, the player interactions that are not considered at the design time often ignored and unhandled. Furthermore, the story viewers of identical choices would have same experiences with the system [14]. To address these issues, Iurgel [4] incorporates a rule-based technique into the authoring tool Cyranus which is embedded in an interactive story platform Art-E-Fact. In his system, some nodes exist outside of the story graph are labeled terminal nodes and can be activated by a rule-based engine without incoming transitions from other nodes in the graph.

In contrast, a narrative system employing planning formalisms takes as input a story which lacks in the user choice and adapts it as the user interacts with the system [2; 6; 8; 13; 20]. While the use of planning formalism in the interactive narrative system relieves the story author of integrating story contents with the user interaction, the system generated stories often give less immersive experience to the system users.

This paper discusses a framework for authoring interactive narratives that employs a hybrid approach to creating interactive stories. Our approach uses branching narrative structures which contain planning annotations (i.e., preconditions and effects) in order to enable automated association between nodes. This framework is a part of a project implementing a software package which targets non-expert users as well as experts in interactive storytelling. We plan to provide the package with rich 3D graphic models and other story elements such as sound effects and music so that even non-expert users can easily create highly engaging interactive stories. The framework utilizes the TVML (TV Program Making Language) technique [22] for story realization into 3D animation. TVML is a text-based language that can describe all the required elements (e.g., stage set, camera, light, characters, prop, voice, sound, etc.) for a 3D animation production. TVML scripts are realized as 3D animation on any TVML player equipped platforms (e.g., television sets, mobile devices, personal computers, etc.). Thus, we believe that this ubiquitous aspects of our framework will be also beneficial for conventional interactive narrative generation systems which typically need to deal with a large quantity of story contents to create stories with a great deal of variability [7; 15; 20].

2 The System Overview

The system provides a pair of authoring tools in the view of the authoring process of interactive narratives as involving two stages: content creation and interactivity creation. Figure 1 illustrates the system architecture involving one story writer, one interaction engineer, and one story participant, in which they could be a single individual. In the figure, the two components for interactive narrative together are called PRISM whose purpose is to take in the system user's behavior (i.e., input) and present diverse storylines as an analogy to a prism separating the light into different colors.

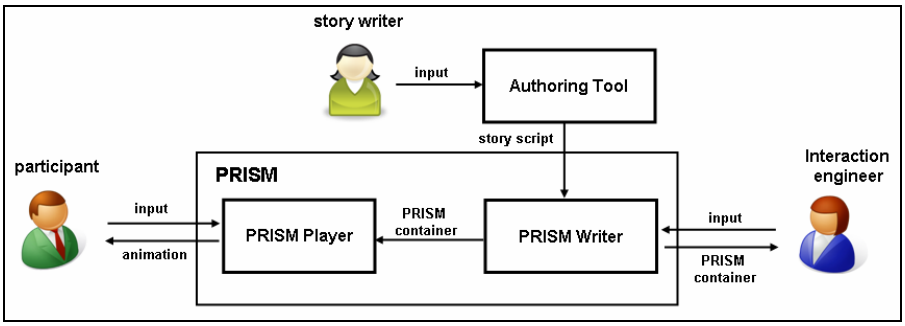

Fig. 1. The interactive story authoring architecture in which PRISM is situated

For content creation, the system provides a graphical user interface (as in Figure 2 in the next page) which employs the TVML technology as its rendering module. The interface allows the user to select rich 3D model items (e.g., stage sets, characters, props) and to control light, camera, music, actor behaviors, dialogues, narration, and captions. Because the content creation authoring tool is stand-alone, it is also available to the users for the exclusive purpose of creating non-interactive stories. The details of the content creating interface are beyond the scope of this paper.

On the completion of authoring story contents in the interface, the authoring tool generates a scene script file, which is an xml document that contains scenes, beats, and shots defined in PRISM as following. The script is then sent to the PRISM writer.

Scenes. A scene describes all the story elements that take place on the same stage.

Beats. A beat is an expressive unit in a story. A beat may include a series of shots that composes a meaningful event. For example, a beat describing a lecture on a specific topic may include shots of teacher explaining the topic followed by a shot of question and answer session.

Shots. A shot describes a set of story elements taken in one camera shot. Depending on the types of camera technique, a shot may span from one second to a few minutes. As an example of a long shot, a camera may track a young girl walking through a winding road in a province. In contrast, in a scene with two actors talking the camera may take short bust-shots of two actors who are facing each other to illustrate their conversation.

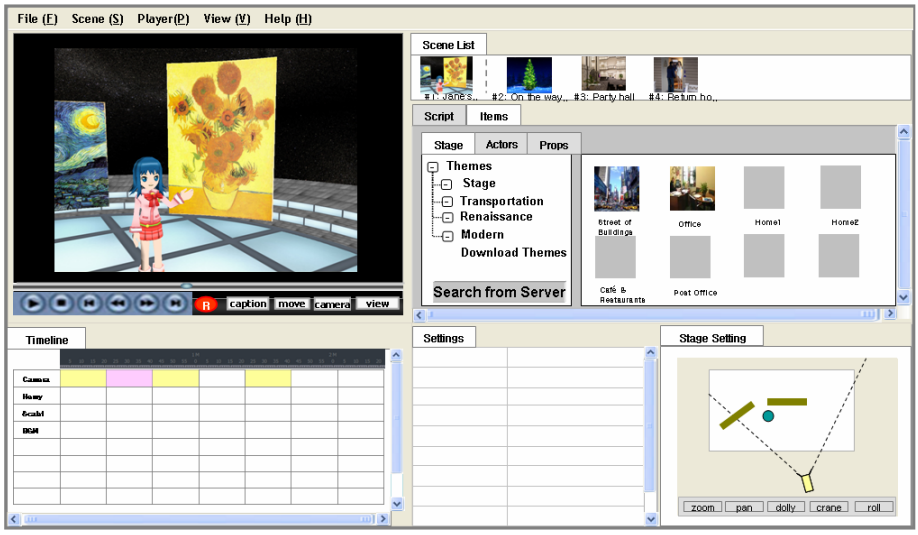

Fig. 2. Screenshot of the prototype User Interface window for content creation

3 The PRISM Writer

The PRISM writer enables the author to fabricate scene scripts into an interactive story utilizing the user interface shown in Figure 3 in the next page. The interface allows the author to construct an interactive story structure and encapsulates it in an XML format—a PRISM container—which is then passed to the PRISM player for presentation to the story participant.

3.1 The PRISM Writer Interface

An interactive narrative structure in PRISM is called a story map (see Figure 3) which represents a story space that the user may navigate and explore. In order to construct a story map, a concept of *situation* is introduced as its basic unit. A situation is a semantic story element that may include a number of beats in disparate scenes. Imagine, for example, a situation where an old man in an asylum recalls an excursion to a state fair with his family decades ago. This situation may contain two beats from two distinct scenes: a beat illustrating him recalling in an asylum scene and a beat illustrating a joyful excursion at a state fair scene. When a situation is created, the author can place the situation onto the story map area by the drag-and-drop style interface.

When given a scene script, the PRISM writer decodes the script to obtain scene information such as title, description, and the first screenshot of the scene. Next, the PRISM writer user interface lays out the scenes contained in the script on the Scene Area located on the right upper area of the window. The author then may click on one of the scenes to lay out the scene's beats on the BeatSequence Area. Then, the author can compose a situation by dragging and dropping beats from the BeatSequence Area

onto the Situation Area. Although the beats and scenes are depicted as labeled rectangles in Figure 3, an implementation of the PRISM writer selects the first shot of each beat and scene as it delegates to show them on the BeatSequence and Scene areas for the author to easily recognize them.

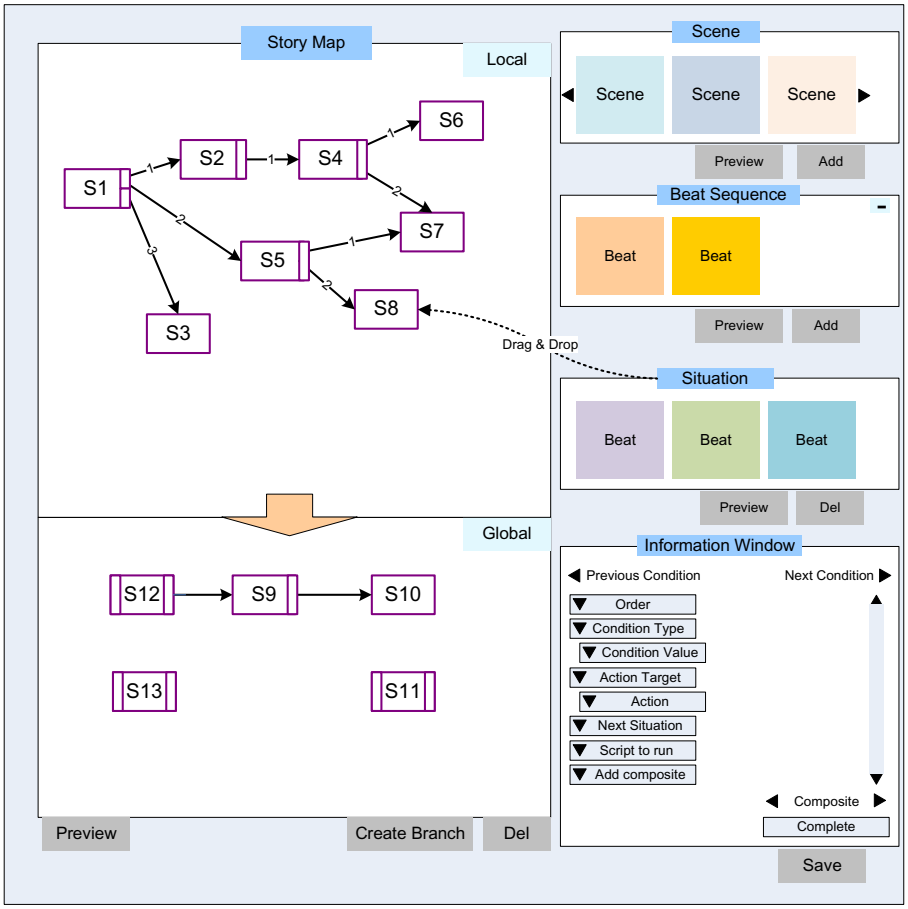

Fig. 3. The PRISM Writer Concept UI Design

The Global situation area on the left lower area of the PRISM writer interface (Figure 3) allows the author to specify a group of situations as global situations; global situations can occur at any point in the presentation of a story when their conditions are met. For instance, the author may produce supplementary situations to be provided when the user requests more information such as the background of the story. The author can easily convert global situations to local situations by dragging them into the local story map area.

3.2 The PRISM Container

The PRISM container is an xml document that describes a narrative structure which fuses branching graphs with planning formalism. As depicted in Figure 3, the story structure in PRISM basically takes a directed graph in which nodes represent situations connected by arcs representing the choice provided by the user or external systems (e.g., weather information, location data). While a conventional branching graph approach provides an intuitive and efficient way for creating interactive stories, it is also disadvantageous in that it restricts the user interaction to a set of actions anticipated by the story author. To compensate this drawback, PRISM also employs a planning approach. In the story graph in Figure 3, rectangles represents situations, where a bar (or a smaller rectangle) on the left denotes its precondition and a bar on the right denotes its effect. When the story graph contains internal nodes that contain no outgoing branches, our planner takes on the job to create links between nodes. As a result, in connecting situations where both approaches are available, PRISM gives higher priority to the branching approach regarding the author created branching as her direct intention. More details on the use of planning techniques in PRISM are discussed in the section 4.1.

In order to incorporate planning formalism into branching narratives, we devise a data structure named *condition* to be shared by both approaches. A condition is a logical sentence which describes a type of user interaction (e.g., text, speech, gesture, facial expression, mouse, brain signal, etc.), external information (e.g., real-time web data, GPS, etc.), and states of the virtual environment in which the story is situated (e.g., scores, characters' locations, the story participant's cognitive and emotional model, story progress, etc.). In the story graph, a condition can serve as a branching decision point that connects a single situation to another. Suppose, for instance, an educational application that displays a pair of fruit props, an apple and an orange, on the screen and waits for the user choice. The node that describes this example may include a condition that links her mouse-click event on the apple prop in a situation to another situation which explains planting apples in South Korea. In this example, the current situation contains a condition represented as a logical sentence (e.g., *User-Mouse-Clicked(apple)*) and embeds a branch that links to the explanation situation.

In the planning mode in PRISM, conditions can serve to represent preconditions and effects. Preconditions are logical statements that must be achieved for a situation to occur. Effects are logical statements denoting significant states that are altered in the virtual story world due to the situation execution. For this, a situation data structure holds places for preconditions and effects, which are optional. As an illustration, suppose that a penniless man who steals a bank with a shotgun. For this situation to take place, the action requires him holding a shotgun as its precondition (i.e., Holding(man, shotgun)) and describes him being rich as its effect (i.e., Rich(man)). To assist the user with constructing conditions, the interface provides the user with a predefined list of conditions and their combinations as well as a text editing box for writing customized conditions.

This hybrid representational approach makes it feasible for the user to reach disjoint sub-graphs as long as the graphs contain appropriate planning annotations—achievable preconditions from execution of other situations. If these two different approaches mutually inconsistent—a branch of a node points to *A node* while its

planning annotation leads to *B node*, the branching approach takes precedence over the planning, in the respect of the author's intention. However, such conflicts would not happen in the current design since the planner is only active when narrative branches are not available. More details on the PRISM container can be found in [10].

3.3 The PRISM Writer Architecture

This subsection describes the PRISM writer architecture underlying the PRISM writer interface. The PRISM writer takes a scene script and generates a PRISM container corresponding to the story map on the PRISM writer interface. As illustrated in Figure 4, the PRISM architecture consists of four components: a story map creator, a post processor, an xml decoder, and an xml encoder. When the PRISM writer interface loads a scene script, the xml decoder analyzes the script and extracts scene and beat information. As the user makes modifications to the graphical representation of the story map on the PRISM writer interface, the story map creator constructs and keeps track of its data structure. The post processor conducts various tasks: it estimates the minimum and maximum play time, and determines the validity of the story map structure. For instance, the processor determines a story map is invalid if it contains multiple cyclic paths. Finally, the xml encoder converts the story map structure into an xml document—the PRISM container.

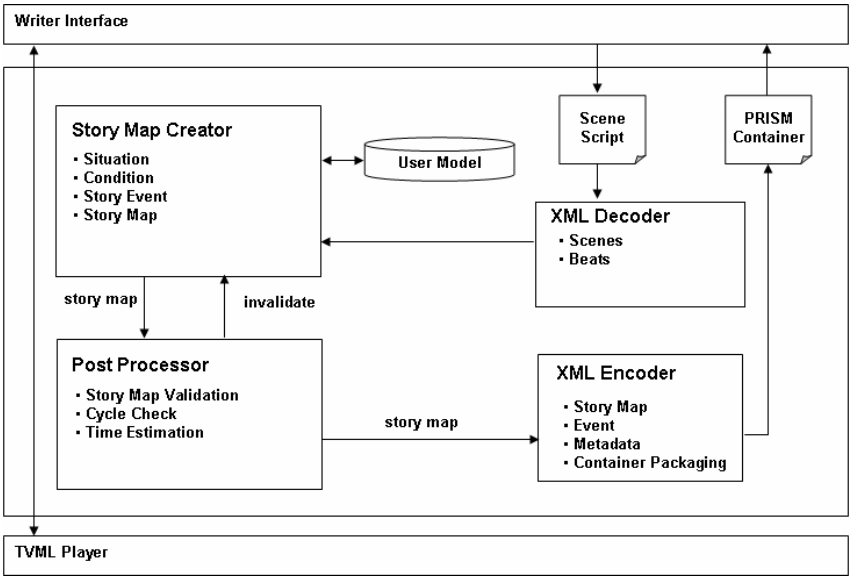

Fig. 4. The PRISM Writer Architecture

4 The PRISM Player

The PRISM Player takes in the PRISM container to realize its TVML scripts on the screen, interacting with the participant. This section discusses the player architecture

and its components, as illustrated in Figure 4. The architecture of the PRISM player focuses on the provision of diverse scenarios to the story participant in real time through her interaction with the story. For this purpose, the PRISM player provides three functionalities: a) decoding a PRISM container, b) managing the interaction between the user and the story, and c) visualizing the story representation.

Upon receiving a PRISM container which is constructed by the PRISM writer, the XML decoder in PRISM parses it to obtain the story information (i.e. story map, story event and metadata) for the story director. The story director then delivers the first situation of the story map to the TVML player, which renders the TVML scripts contained in the situation for the user. The following subsections describe the PRISM player components focusing on the story director and the TVML player, which are two primary modules responsible for interaction management and visual realization respectively.

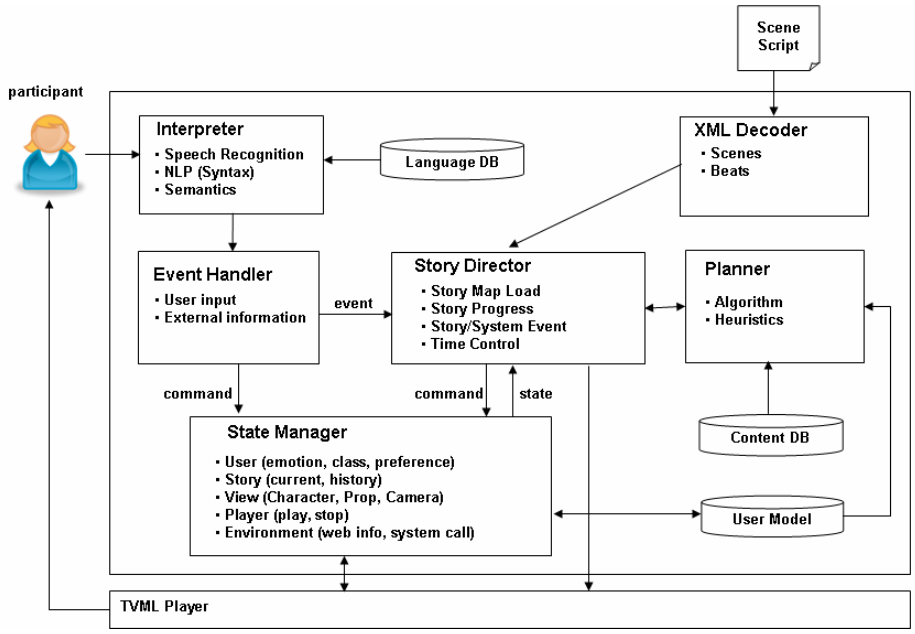

Fig. 5. The PRISM Player Engine Architecture

4.1 The Story Director

The story director takes an interpreted event from the event handler and determines if the event is issued in order to manipulate an item (e.g., characters, props) or to advance the story, where an event can describe the user input, external information, and story world states. For example, the story author makes a character simply wave his hand when the user mouse-clicks on him in a situation, which would not affect the storyline. On the other hand, the user's decision to take a red pill may lead to another situation, unfolding the story. Events and their treatments are described in the PRISM

container and registered in PRISM during the XML container decoding step so that the story director can refer it to generate reactions to an event.

The user can interact with the system through various input medium such as text, voice, and mouse pointers. To handle the user's natural language interaction, the interpreter processes the following steps: speech recognition, natural language syntactic processing (NLP), and the semantic processing. First, the speech recognition phase converts the user speech input into text. As the second step, our NLP processes the pipeline of tokenization, word tagging, pronoun resolution, named entity tagging, and noun phrase & verb phrase analysis. The final step, the semantic processing, conduct a shallow parsing task that extracts a list of logical sentences from the syntactically analyzed text produced in the second stage, which are then sent to the event handler. When the user input is non-text, the interpreter bypasses the input to the event handler. The event handler encodes the given event as a form of a logical sentence (e.g., *User-TextInput("Help")*, *User-Mouse-Clicked(apple)*) using a simple template matching technique, and then it dispatches the sentence to the story director and to the state manager.

In case where the interpreted event is related to story progression, the story director retrieves the next situation by employing two methods: conditional branching and planning. First, the story director matches the input to the conditions of the current situation. If a matching condition and its branch are found, the story director replaces the current situation with the branch's destination. If no such branches are found, the director delegates its situation retrieving role to the planner. The planner utilizes a planning algorithm—currently a simple forward planning at a depth of one—to find a situation which contains a set of preconditions that are satisfied at the point. This approach has been used as an efficient content selection method in Façade, a real-time drama application [8]. However, since content selection based on matching preconditions does not assure the provision of dramatic experiences for the user, the system would require heuristic functions that guide the selection of story elements. For instance, Weyhrauch [19] devises an evaluation function that rates the aesthetic value of the user experiences in an interactive storytelling environment. As such guidance in our system, we plan to devise a generic heuristic function for content selection based on syntactic information of the story graph such as branching factors. On the completion of selecting the next situation, the planner then passes it to the story director.

The state manger maintains various states regarding the virtual story environment and the user. The manager updates the states when an event is input from the event handler or the TVML player. The manager reports a state in response to a query sent from the story director. When events regarding the user model occur, the state manager modifies diverse fields in the user model (e.g. cultural level, educational level, preference, personality, etc.). The user model plays an important role in constructing a personalized story that considers the user's cognitive and emotional states. The user model in the current design employs a planning-based reasoning: a reasoning algorithm, reasoning capacity, knowledge, and preference. As a reasoning algorithm we plan to utilize a partial-order causal planner. To represent the user's knowledge, we will use a set of plan operators which contains preconditions and effects. The user's preference includes heuristic function that controls the content selection processes as described above.

4.2 Visualization

The TVML player [22] visually realizes the story representation into 3D animation. TVML is a text-based language that can describe all the required elements (e.g., stage set, camera, light, characters, prop, voice, sound, etc.) for a 3D animation production. In TVML script, one line of script corresponds to one primitive event. For example, with given a TVML script "character: talk(name = BOB, text = "My name is BOB")," the TVML Player generates the 3D animation that the character Bob appears on the screen and talks "My name is BOB" with synthetic voice in real time. The TVML script is device independent; thus, it can be thus realized on any platforms (e.g., television sets, mobile devices, personal computers, etc.) where a TVML player is equipped. Our TVML player provides more advanced functionalities than the publicly open source TVML player [22]. The TVML player in our system supplies characters with human-like motions and one-skin model for high-polygon 3D characters. The system also provides a variety of fonts, CG effects such as fade-in and fade-out, and lighting. Furthermore, we plan to provide relative coordination and intelligent interaction among multiple characters and props.

5 Conclusion and Future Work

In authoring interactive narratives, two primary approaches—branching graphs and AI planning—have been significantly used to augment interactivity into conventional linear narratives. Although each approach offers its own possibilities and limitations, few efforts have been made to blend these approaches.

Thus, we have presented a hybrid framework that adapts the conventional branching graphs by integrating AI planning formalism. As a result, the use of branching graphs provides the story author with an intuitive illustration of the story and the story participants with coherent and logical story experience when properly authored. In addition, the use of the planning technique offers the possibility of automated story generation when no predefined branches are available for a given node in the story graph.

Although our initial intention to develop a practical authoring tool for non-experts who need to create interactive stories, it is also expected that this work will be valuable for the interactive storytelling environment systems [7; 15; 20] in which a large volume of high-quality story contents is essential for the provision of varied endings for a given story.

The system is currently under development; we have implemented the content authoring tool and our own TVML player, and designed the xml schema to represent the interactive story structure [10]. We will implement the PRISM player by the end of this year and the PRISM writer by the next year.

Our future work includes the formal evaluation of our system and provision of intelligent assistance for professional knowledge needed work such virtual camera control. Furthermore, we consider employing statistical user models such as Bayesian network. We also plan to supply interface for a variety of user input devices such as the Emotive EPOC^{TM} neuroheadset which detects the human thoughts, expressions, and emotion, in order to maximize the participant's immersive experiences.

References

1. Carbonaro, M., Cutumisu, M., McNaughton, M., Onuczko, C., Roy, T., Schaeffer, J., Szafron, D., Gillis, S., Kratchmer, S.: Interactive Story Writing in the Classroom: Using Computer Games. In: 2nd International Digital Games Research Conference, Vancouver, Canada, pp. 323–338 (2005)
2. Cavazza, M., Charles, F., Mead, S.: Planning Characters' Behaviour in Interactive Storytelling. Journal of Visualization and Computer Animation 13, 121–131 (2002)
3. Donikian, S., Portugal, J.N.: Writing Interactive Fiction Scenarii with DraMachina. In: Göbel, S., Spierling, U., Hoffmann, A., Iurgel, I., Schneider, O., Dechau, J., Feix, A. (eds.) TIDSE 2004. LNCS, vol. 3105, pp. 101–112. Springer, Heidelberg (2004)
4. Iurgel, I.: From Another Point of View: ArtEFact. In: Göbel, S., Spierling, U., Hoffmann, A., Iurgel, I., Schneider, O., Dechau, J., Feix, A. (eds.) TIDSE 2004. LNCS, vol. 3105, pp. 26–35. Springer, Heidelberg (2004)
5. Kelso, M., Weyhrauch, P., Bates, J.: Dramatic Presence. Journal of Teleoperators and Virtual Environments 2(1) (2005)
6. Magerko, B.: Story Representation and Interactive Drama. In: The 1st AIIDE, Marina del Rey, CA, USA (2005)
7. Mateas, M., Stern, A.: Façade: An Experiment in Building a Fully-Realized Interactive Drama. In: Game Developer's Conference: Game Design Track, San Jose, CA (2003)
8. Mateas, M., Stern, A.: Structuring Content in the Facade Interactive Drama Architecture. In: The 1st AIIDE, Marina del Rey, CA (2005)
9. Medler, B., Magerko, B.: Scribe: A General Tool for Authoring Interactive Drama. In: Göbel, S., Malkewitz, R., Iurgel, I. (eds.) TIDSE 2006. LNCS, vol. 4326, pp. 139–150. Springer, Heidelberg (2006)
10. Min, W., Shim, E., Kim, Y., Cheong, Y.: Planning-Integrated Story Graph for Interactive Narratives. In: ACM Multimedia 2008 SRMC Workshop, Vancouver, BC (2008)
11. Onuczko, C., Cutumisu, M., Szafron, D., Schaeffer, J., McNaughton, M., Roy, T., Waugh, K., Carbonaro, M., Siegel, J.: A Pattern Catalog For Computer Role Playing Games. In: 5th GameOn North America, Montreal, Canada, pp. 30–38 (2005)
12. Riedl, M., Rowe, P., David, K.E.: Toward Intelligent Support of Authoring Machinima Media Content: Story and Visualization. In: 2nd international conference on INtelligent TEchnologies for interactive entertainment (2008)
13. Riedl, M.O., Stern, A.: Believable Agents and Intelligent Story Adaptation for Interactive Storytelling. In: Göbel, S., Malkewitz, R., Iurgel, I. (eds.) TIDSE 2006. LNCS, vol. 4326, pp. 1–12. Springer, Heidelberg (2006)
14. Riedl, M., Young, R.M.: From Linear Story Generation to Branching Story Graphs. The IEEE Journal of Computer Graphics and Applications, 23–31 (2006)
15. Riedl, M.O., Stern, A.: Believable Agents and Intelligent Scenario Direction for Social and Cultural Leadership Training. In: Behavior Representation in Modeling and Simulation Conference, Baltimore, Maryland (2006)
16. Spierling, U., Weiß, S.A., Müller, W.: Towards Accessible Authoring Tools for Interactive Storytelling. In: Göbel, S., Malkewitz, R., Iurgel, I. (eds.) TIDSE 2006. LNCS, vol. 4326, pp. 169–180. Springer, Heidelberg (2006)
17. Swartout, W., Hill, R., Gratch, J., Johnson, W.L., Kyriakakis, C., LaBore, C., Lindheim, R., Marsella, S., Miraglia, D., Moore, B., Morie, J., Rickel, J., Thiébaux, M., Tuch, L., Whitney, R., Douglas, J.: Toward the Holodeck: Integrating Graphics, Sound, Character and Story. In: 5th International Conference on Autonomous Agents, Montreal, Quebec, Canada, pp. 409–416 (2001)

18. Tully, G., Turner, S.: Integrated Decision Points for Interactive Movies. In: Göbel, S., Spierling, U., Hoffmann, A., Iurgel, I., Schneider, O., Dechau, J., Feix, A. (eds.) TIDSE 2004. LNCS, vol. 3105, pp. 61–67. Springer, Heidelberg (2004)
19. Weyhrauch, P.: Guiding Interactive Ficiton. Ph.D. Dissertation, Carnegie Mellon University, Pittsburgh, PA (1997)
20. Young, R.M., Riedl, M.O., Branly, M., Jhala, A., Martin, R.J., Saretto, C.J.: An Architecture for Integrating Plan-Based Behavior Generation with Interactive Game Environments. Journal of Game Development 1(1), 51–70 (2004)
21. INSCAPE Storytelling, http://www.inscapers.com/
22. TVML, http://www.nhk.or.jp/strl/tvml/english/player2/index.html