What Would You Do in Their Shoes? Experiencing Different Perspectives in an Interactive Drama for Multiple Users

Birgit Endrass, Michael Boegler, Nikolaus Bee, and Elisabeth André

Augsburg University, Universitätsstr. 6a
86135 Augsburg, Germany
{endrass,bee,andre}@informatik.uni-augsburg.de
http://www.multimedia-werkstatt.org

Abstract. Following the idea of improvisational theater, we built a multi-agent system where multiple users can experience an interactive drama from different perspectives. Depending on the roles that the users are given in advance, the situation is perceived in various ways and interaction possibilities are directed according to the assigned role. The computer-controlled agents' behavior needs to be highly reactive in order to allow for flexibility in the conversational flow of the structured interactive narrative. In this paper, we describe a system architecture that allows multi-user participation as well as distributed behavior planning for multiple agents.

1 Motivation

When discussing interactive storytelling nowadays, most people think of computer games where the user can participate in the story by playing the part of one of the characters. But the idea of interactive storytelling is much older than computers and computer games after all.

Theaters have always been places where people go to enjoy themselves by watching actors in a story. The interactive part arises from a special version of theater - the improvisational theater. In improvisational theater, the audience is actively creating characters that participate in the play, illustrated by actors. Thus, a storyline evolves dynamically from the input of the audience, a facilitator and from spontaneous ideas of the actors. In that manner the improvisational theater does not depend on a predefined script; however, it needs a predefined concept.

The idea of the work described in this paper is based on the concept of improvisational theater. The user finds him/herself in the role of an actor, that is placed into a predetermined concept. Following the ideas of improvisational

theater, he is allocated a certain role with predefined attributes, such as personality, gender or status. To ensure that the user stays within the given role, interaction possibilities are limited by the system, according to the role.

Additionally, the user does not know which characters he is going to meet during the play. These other characters can either be controlled by other users or by the system. Thus, the computer completes the tasks of the audience, the facilitator as well as other actors. In that manner, the user can take an active role in the performance if he wishes to do so. If not, the characters will give a performance on their own and dictate where the scenario is headed.

In this paper we describe an approach where the user can experience a virtual story from different perspectives. Following the ideas of improvisational theater, the user joins a virtual dinner party in a role that he does not know in advance. Depending on the assigned role the same situation can be perceived in various ways. For example, an argument between a couple could either be a pleasant development (from the perspective of a "nasty ex-friend") or an unpleasant one (for the host of the party or a friend of the couple). In this vein, the user can experience different stories, depending on the role he is given, the characters he is meeting as well as the actions he chooses to perform. In the open ended scenario neither the user nor the system can exactly predict how the story is going to evolve.

2 Related Work

Most well-established systems in interactive digital storytelling are based on the following idea: A given story is modeled in a virtual world where a user is in the role of a certain character and performs actions within a structured interactive narrative that allows for a certain amount of flexibility in the conversational flow. For the work described in this paper, we aim at building interactive drama where multiple users can experience a virtual scenario from different perspectives, depending on the role that they are given in advance.

Pizzi and colleagues [1] describe, for example, a system that uses an existing story for storytelling. The French novel Madame Bovary by Gustave Flaubert constitutes the baseline for this interactive digital drama. The user takes part in the role of the character Rodolphe, who tries to encourage Emma (the main character) to cheat on her husband. Through interaction the user is able to influence the characters' feelings which in return affects their behavior. In this vein, the user can experience different outcomes of the story depending on his interactions. The characters' behavior is based on a multi-threaded planner, controlling each character independently. The system described in this paper is also based on a distributed planning approach. However, while we focus on multiple users experiencing the same story from different perspectives, their main contribution is to describe the planning domain in terms of the characters' feelings and emotions.

Another approach is described in [2], where the story of a married couple is narrated that invites an old friend to their house for drinks. The user (in the role

of the friend) finds the couple going into raptures about their lives and weaving around the fact their marriage is falling apart. The story is based on the play "Who's Afraid of Virginia Woolf?", which was chosen because the storyline can be broken apart into story beats that can be resequenced. The characters' sequential and joint behaviors are modeled using ABL (A Behavior Language). In ABL, activities are represented as goals that are supplied with one or more behaviors to accomplish it. The user's interactions influence the behavior selection process of the characters. Consequently, the user can experience different story lines depending on his interactions within the scenario.

Although in the original story of "Who's Afraid of Virginia Woolf?" another couple is invited, Facade is designed for only one user to join the scene. In contrast to them, we enable multiple users to participate in a scenario taking on different roles.

Lankoski and colleagues [3] describe an approach where the user is in the role of a character that wants to seduce a rock star, who has promised to stay a virgin until marriage. To achieve this goal, the user can interact with four different characters. The user's interactions affect the characters' impression, which in turn influences their attitude towards the user character and thus determines their own behavior. To communicate within the virtual scenario, prewritten dialog trees are used, where the branches are based on the current impressions of the non-playing characters.

Aylett and colleagues [4] discuss the role of narrative management in a character-based emergent narrative framework. Instead of building a system that stays on a unique, pre-determined plot, they want to leave more interactive freedom to the user. Following the idea of Game Masters in paper and pen story telling, they introduce an approach using a hierarchical planner and way-points in a story line. In their work, they propose two ways of implementing a story facilitator: one triggers narrative actions taking into account the idea of way-points, the other considers the emotional impact of actions in order to intensify the dramatic effect. Thus, actions are not only dependent on motivations and goals, but also on the characters' emotions.

Unlike the approaches above, our work aims at telling a story from different perspectives allowing an arbitrary number of users to participate. The user (or multiple users) are able to play different roles and meet different characters similar to improvisational theater. By allowing an arbitrary number of participants we can also stay more flexible in alternating the story.

André and colleagues [5] describe the evolution of character-based presentation systems, starting from systems in which a single character simply presents information, over presentation teams to interactive installations. The authors discuss how concepts from improvisational theater may be employed to design interactive presentations with open ended story lines. They also analyze the benefits of various planning approaches for their typology of character-based presentation systems and argue for distributed reactive planners in scenarios where the user takes on an active part.

Following this approach, Rist and colleagues [6] describe a flexible platform that enables the development of applications with virtual characters including user interaction. In a sample application, the user joins a group of three agents that find themselves in the scenario of a car sale. The user is free to define the characters roles within the scenario as well as their personalities, attitudes and initial status. Thus, the dialog varies according to the defined roles and personalities as well as to the user's interactions. Although this platform is highly flexible and allows for different constellations of characters, it does not support multi-user playing. In addition, the user is not able to switch into another role. Following their ideas to control the virtual characters' behavior, we build a multiuser-multiagent scenario based on their approach to distributed behavior planning.

3 Interactive Drama

In order to illustrate our ideas on multiuser-multiagent scenarios which allow users to experience a drama from different perspectives, we designed a virtual dinner party where different characters meet. Figure 1 shows a screenshot of our scenario.

To ensure that an interesting story evolves, some of the characters within the scenario are always computer-controlled. Further details on the implementation of these agent components are provided in Section 4.2. By pre-defining some of the participating characters, parts of the story are controlled. Thus, an amorous couple as well as a jealous ex-friend will meet each other sooner or later at the

Fig. 1. The virtual scene of our interactive dinner party

virtual dinner party. This can lead to an argument or not, depending on the users' roles, their interactions as well as other participating characters.

If the user's assigned role is for example being the ex friend's best friend, he might want to try to separate the couple in order to allow his friend to talk to his/her ex. On the other hand, if the user finds himself in the role of the new lover's best friend the task could be supporting the friend by either keeping the conversation friendly or exposing the dismissed ex friend. Thus, the user's implicit goal is determined by the given role and his / her interactions are limited by the user interface. In order to avoid conflicts of the users' interactions with the agents' behavior, the user interface only offers options that make sense in the context of the story.

Similar to the users' roles, the characters' genders are not fixed. In a two-male one-female constellation the men might poke each other, while in a two-female one-male version the strategy might be more artful, e.g. "bitching" about the rival.

This story is embeddeded in the larger context of a dinner party. Thus, other characters with different roles are joining the party as well. The host, for example, will try to keep the party as friendly as possible, while a drunken guest might poke other guests at the next opportunity. If a user decides to be a passive bystander, the party will develop on its own. Users can also interact with each other while trying to achieve their individual goals. In the beginning of the interactive drama the user is given a particular role that he / she plays during the whole story. By re-entering the virtual dinner party, a new role will be selected for the user. Thus, multiple users can experience the same scenario from different perspectives and influence the storyline depending on their actions.

4 System Architecture

In this section we describe our system architecture that allows for communication between multiple users and multiple agents within our virtual dinner party (see Figure 2).

In the interactive drama, the computer controlled agents' behaviors are determined by their role as well as their predefined set of communication rules, that are designed for their given role. If an action is triggered by the planning component (verbal and nonverbal), it is sent to the application, where it is displayed using body postures, gestures and natural language.

The user is represented by a user avatar that is controlled through a user interface which offers the user various interaction categories. The choice of possible interaction categories is constrained depending on the role that the user is playing. For example, the host of the party that is supposed to keep the party friendly can not start an argument.

To animated the interactive drama, we use the Horde3D GameEngine [7], which was developed at Augsburg University and which is freely available. In the scenario, virtual agents can wander around and engage in a natural language conversation with each other and the user. In our system architecture, the virtual scenario operates as a server where an optional number of users and agents can be registered.

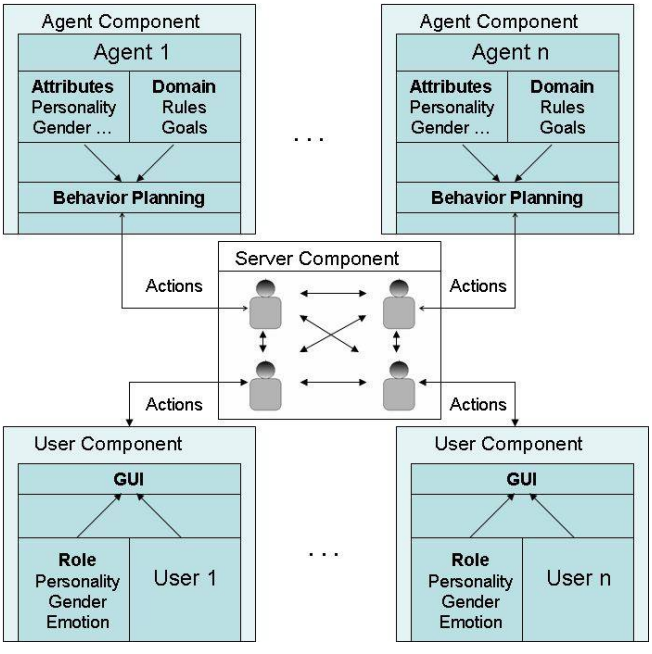

Fig. 2. Overview of system architecture

Our architecture offers several features that are of interest to interactive drama. First, an arbitrary number of characters (either controlled by the user or a computer) can be added to or removed from the interactive drama at runtime. Thus, a highly dynamic story may evolve depending on the constellation of the participating characters.

Another advantage is that the scenario can be expanded by adding new characters and roles. In order to create new system-controlled agents, a new client may be added with a set of rules. Thus a drunken party guest could, for example, be modeled by adding rules that make it poke other party guests. If a new interaction type is defined, appropriate utterances need to be added to the knowledge base.

New user roles can be defined in a similar manner. Existing interaction types can be reused, or new interaction types can be defined to add new possibilities to the scenario. Thus, the clients participating in the scenario can easily be defined at will and new scenarios can be created by defining roles for user characters or computer-controlled agents. In the following, we further describe the components of our system architecture.

4.1 User Component

A novel feature of our approach is that multiple users can join our interactive drama at the same time. In the virtual scenario, they are represented through user avatars. Thus, the users do not know whether they are interacting with

another user or a computer-controlled character. Whenever a new character joins the scene, the server creates a new instance for it. Thus, users and agents can freely interact with each other via the server.

The user's communication is limited to a given set of interaction types, which is a streamlined version of the DAMSL annotation scheme [8]. The DAMSL scheme distinguishes dialog acts into several layers and interaction types, such as statement, info request, influence on future, agreement/disagreement, understanding/misunderstanding or answer. In order to obtain interesting scenarios, social emotional interactions such as poke, comfort or defense are added. Depending on the selected user role, the full set of interaction types is limited. Consequently, only a few possibilities are offered to the user via a drop down menu. The option selected by the user is then sent to the server where an appropriate utterance is generated out of a knowledge base and spoken by the user avatar.

4.2 Agent Component

According to [6] AI-based approaches, and plan-based approaches in particular, are becoming more and more popular to control the behavior of synthetic characters. Using a planner for behavior generation, a complex goal (e.g. getting acquainted with another agent) can be decomposed into smaller subgoals and actions (greet the conversation partner, engage in casual small talk, ...). Depending on character-specific attributes, such as personality or gender, actions can be performed in different ways. Also the selection of planning steps to archive a goal can be varied.

For the realization of our interactive drama, we use the Simple Hierarchical Ordered Planner (JSHOP2), developed at the University of Maryland [9]. This planner meets our requirements in a very satisfying manner as it is completely domain-independent and can easily be used to plan dialogs for virtual characters. However, the software had to be modified, as it does not allow for updates in the knowledge base at runtime in its original version. For our purposes this is a crucial task, as incoming actions (either performed by another agent or a user) need to be recognized and reactions need to be planned accordingly.

Figure 3 shows the integration of the planner software into our agent-component. If an interaction takes place in the virtual scenario and is sensed by the agent (either because it is the target of the message or stands close to the sender of the message) it is parsed to meet the specification of the behavior planning component. As we stated above, incoming messages have to be added dynamically to the knowledge base. Thus, we added a dynamic knowledge base to the planner software, that holds information about received interactions whereas the initial knowledge base holds information about the agents' attributes or knowledge about the virtual world. An incoming action will restart the behavior planning. Consequently, the planner within the agent-component decides if and which action should be triggered according to the updated knowledge base. Actions are realized in the same manner as for the user component. Thus, if an action is selected it is parsed back and communicated to the server component, where natural language is generated or appropriate gestures are selected.

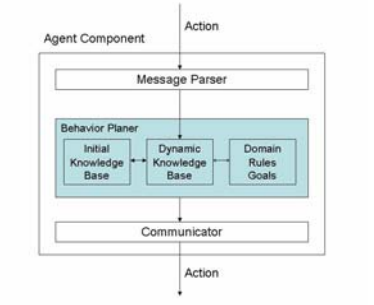

Fig. 3. Overview of agent component

5 Virtual Scenario

In our system architecture, the virtual scenario includes the server for the multiagent communication. Thus, besides rendering the virtual scene holding the agents, it has further tasks to solve, such as maintaining the verbal and nonverbal knowledge base, generating natural language or managing timing issues. In addition, the registration of users and agents needs to be administrated by this component. Figure 4 shows an overview of the server's architecture.

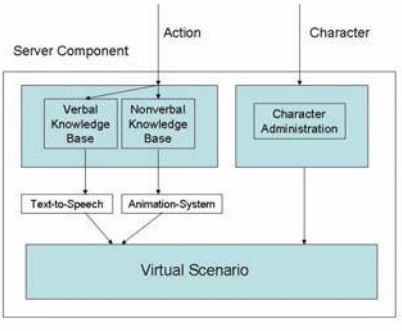

Fig. 4. Overview of server component

As we mentioned above, an arbitrary number of agents and users can interact with each other. Thus, the server needs to add or remove characters dynamically to or from the scene. Within the character administrator, a list of active agents is maintained for that purpose.

Another sector of the server component holds the verbal and nonverbal knowledge base, storing appropriate utterances and gestures for all interaction types. If an action is selected by a virtual character or a human user, an abstract representation is sent to the server component, where a corresponding utterance and/or gesture is selected. Variables for participating characters, discussed topics etc. are replaced using a simple template-based generation approach. For

example, a greeting could either be performed by an utterance, such as 'Hello Sam. Nice to meet you!', a greeting gesture or a combination of both, depending on the characters' role.

In our virtual dinner party, the action is then performed by the corresponding virtual character, using gestures, mimics and synchronized speech. In order to avoid undesirable overlaps in the scenario, the virtual scenario also handles the timing of the utterances.

5.1 Horde3D GameEngine

The design of the Horde3D GameEngine is based on dynamically loadable components. All objects within a game application are represented as an entity in the game world to which a component can be attached. For example, to simulate the behavior of a box that interacts with other objects in a scene, a physics component with appropriate parameters, such as mass, is added. A component is encapsulated from the rest of the GameEngine code as much as possible, which makes it easy to add new functionality without understanding the whole source code. The components interact with each other using an event-based messaging system. The current version of the Horde3D GameEngine includes several basic components for controlling animations, sound and text-to-speech (TTS), physics simulation, inverse kinematics and facial animation control.

5.2 Components

The Animation Component is responsible for controlling animations in the virtual scene. This component attached to an entity (e.g. a virtual character) enables it to play any suitable animation. It provides methods for blending animations on multiple stages. The Scenegraph Component provides an interface to the scene graph of the Horde3D Graphics Engine. It checks the scene graph transformation of all entities with a Horde3D representation and notifies all other components whenever these transformations have changed. The Bullet Physics Component integrates the the Bullet Physics Library into the GameEngine, allowing e.g. collision detection between objects in the game world or simulating physical behavior of objects in the scene. The Sound Component offers methods to play sound, which can be attached to an entity and plays the 3D sounds depending on their respective locations. The Speech Component is a text-to-speech module that makes the virtual characters talk lip-sync (using the Microsoft Speech API), given they provide the necessary morph targets for the corresponding visemes. Further it allows to map pre-recorded voice files to lip-sync with a virtual character. The FACS Control Component controls a virtual characters' facial expressions using the Facial Action Coding System (FACS). Further details can be found in [10]. Finally, there is an Inverse Kinematics Component, which enables pointing gestures and is mainly used to control a virtual character's head and eye direction.

5.3 Agent Control

The component-based approach of the Horde3D GameEngine allows us to attach an agent component (see Section 4.2) to each virtual character entity. Thus, the server is able to control the virtual character directly while the details for animating the virtual characters or text-to-speech are encapsulated in the other components of the GameEngine. The agent component plans the following four actions: *act*, *speak*, *gesture* and *goto*, whereas each actions holds subactions, such as *greet*, *ask* or *answer* for the *speak* action for example.

To control the virtual characters, we transform the abstract actions broadcasted by the agent component into direct commands for the GameEngine. For example, if an agent component sends the action `agent1 speak greet agent2`, agent1 looks up the appropriate greeting utterance and speaks it via the TTS component. The necessary knowledge base is attached to the virtual scene as a Horde3D GameEngine component. In this manner, each agent component of a virtual character is able to easily access the knowledge base and translate the actions for example to utterances that shall be spoken.

6 Conclusion and Future Work

Every situation is judged differently depending on the observer. Following this idea, we described a system architecture where multiple users can experience the same story from different perspectives depending on their assumed role. In our interactive drama the user finds himself in the role of a guest at a dinner party, where virtual agents as well as other users interact with each other. The virtual agents' behavior is controlled by a distributed planning system. Thus, they are independent from each other and highly reactive to interactions either performed by other agents or human users.

The virtual scenario is visualized by the Horde3D GameEngine, which is based on dynamically loadable components. Users represented through user avatars can join and leave the interactive scenario at any time. Their interactions are limited to their assigned role, thus they need to deal with a given situation and the implicit goal that is determined by their given role.

In this paper, we focused on multiple user participation. However, the users' interaction possibilities were limited to interaction categories for which the system generated an utterance and the corresponding nonverbal behavior. In our future work, we will allow for natural language input in order to build a more interesting scenario for the user. In addition, we plan on expanding our system by defining more system-controlled characters as well as additional user roles.

Acknowledgments

The first author of this paper was supported by a grant from the Elitenetzwerk Bayern (Elite Network Bavaria). This work was also funded by the European Commission under grant agreement IRIS (FP7-ICT-231824) and the German

Research Foundation (DFG) through its funding for the Cube-G project (RE 2619/2-1). In addition, the authors wanted to thank Volker Wiendl for initiating the Horde3D GameEngine.

References

1. Pizzi, D., Charles, F., Lugrin, J.-L., Cavazza, M.: Interactive Storytelling with Literary Feelings. In: Paiva, A., Prada, R., Picard, R.W. (eds.) ACII 2007. LNCS, vol. 4738, pp. 630–641. Springer, Heidelberg (2007)
2. Mateas, M., Stern, A.: Facade, An Experiment in Building a Fully-Realized Interactive Drama. In: Game Developers Conference, Game Design track (2003)
3. Lankoski, P., Horttana, T.: Lies and seductions. In: Spierling, U., Szilas, N. (eds.) ICIDS 2008. LNCS, vol. 5334, pp. 44–47. Springer, Heidelberg (2008)
4. Aylett, R., Louchart, S., Tychsen, A., Hitchens, M., Figueiredo, R., Mata, C.D.: Managing emergent character-based narrative. In: Second International Conference on INtelligent TEchnologies for interactive enterTAINment (INTETAIN 2008). ICST, Brussels (2008)
5. André, A., Rist, T.: Controlling the Behaviour of Animated Presentation Agents in the Interface: Scripting versus Instructing. AI Magazine, Special Issue on Intelligent User Interfaces 22(4), 53–66 (2001)
6. Rist, T., André, A., Baldes, S.: A Flexible Platform for Building Applications with Life-Like Characters. In: Johnson, W.L., André, A., Domingue, J. (eds.) IUI 2003, pp. 158–165. ACM, New York (2003)
7. Augsburg University, `http://mm-werkstatt.informatik.uniaugsburg.de/projects/GameEngine`
8. Allen, J., Core, M.: Draft of DAMSL: Dialog Act Markup in Several Layers, `http://www.cs.rochester.edu/research/speech/damsl/RevisedManual.html`
9. University of Maryland, `http://www.cs.umd.edu/projects/shop/index.html`
10. Bee, N., Falk, B., André, A.: Simplifed Facial Animation Control Utilizing Novel Input Devices: A Comparative Study. In: Conati, C., Bauer, M., Oliver, N., Weld, D.S. (eds.) IUI 2009, pp. 197–206. ACM, New York (2009)