RCEI: An API for Remote Control of Narrative Environments

Federico Peinado and Álvaro Navarro

Departamento de Ingeniería del Software e Inteligencia Artificial
Universidad Complutense de Madrid
c/ Profesor García Santesmases s/n, 28040 – Madrid, Spain
email@federicopeinado.com, alvaro.nav@gmail.com

Abstract. The proposal of this research project is to develop a standard connection mechanism to make narrative environments and the external systems that control them interoperable. Thanks to a new communication interface called RCEI, different knowledge-based storytelling systems will be able to perform interactive drama using different narrative environments that were available. The controller has a reasonable level of granularity, being the only additional requirement the development of valid adapters for both extremes of the connection. Open source implementation of this protocol and language is provided in order to save extra parsing and serializing efforts and make software more interoperable.

Keywords: Game Based Interfaces, Interactive Digital Storytelling, Virtual Environments and Intelligent Medias.

1 Introduction

Interactive Digital Storytelling (IDS) applications differ very much in their presentation to the user. Most of them use some kind of virtual environment with narrative or dramatic qualities, such as visual capabilities for showing text dialogs, representing 2D or 3D characters, objects and locations, giving feedback to the user through game-like HUDs, etc. These virtual environments are, of course, interactive, so their states are continuously changing, frequently in real time, depending on the interactors' actions and the behaviours implemented in the environment (logical or physical reactions, autonomous and proactive agents, hard-coded events, etc.). These environments also differ between applications and usually the integration with the final application is an expensive task in terms of time and effort.

This paper describes the proposal of a standard communication protocol and language designed for connecting bidirectionally a narrative virtual environment and a remote controller system in a generic way, abstracting low-level details relative to the composition of the multimedia effects of the virtual environment and the internal representation of the whole model that the controller uses privately. This toolkit receives the name of Remote-Controlled Environments Interface (RCEI [1]).

The development of this interface is part of a bigger project called Knowledge-Intensive Interactive Digital Storytelling system (KIIDS [2]), a framework and a

M. Cavazza and S. Donikian (Eds.): ICVS 2007, LNCS 4871, pp. 181–186, 2007.

library of domain-specific components for developing directed-IDS applications with the possibility of representing the drama in a RCEI-compatible environment.

The paper is divided in these sections: Section 2 presents a short overview of the related work on this topic. Section 3 presents the communication protocol and language of RCEI. Section 4 adds some implementation details, and finally Section 5 explains what are the conclusions of this research.

2 Background

The need of communication between narrative environments and controllers have been in the field since the very first projects on IDS. In most cases communication is just performed using a proprietary format, with no special intention of establishing a general or reusable language. Three projects with communication mechanisms comparable to our proposal have been chosen for its consideration in this section.

The first one is the Interactive Drama Architecture [3], an IDS proposal that includes an automatic director of the interactive experience implemented as a rule-based system. This architecture is interesting because it has features of a mixed approach, in which both a centralized director (or drama manager) and semi-autonomous (directed) characters collaborate in order to develop an expressive, variable and interesting plot. The controller of the story basically sends messages that are basic commands for a non-player character available at the environment for the player to interact with, messages such as "explore the environment", "get this item", "go to that room", "say something to the player", etc.

The second one is Zocalo service-oriented architecture [4, 5], designed for creating interactive narratives within game-based environments in which direction by planning is performed by Web services running on different machines. It is one of the first projects that take seriously scalability, security and interoperability issues of IDS applications' software components. Its communication formats are well documented, so they can be reused when developing extensions to the architecture. Content of the messages (mainly operators) have been found relatively coupled with the semantic specification of Longbow's input, a main planner that this research group uses, and more clearly coupled with the planning paradigm.

Finally the third one is Shadow Door agent interface [6], an agent control interface for the game engine of Neverwinter Nights (NWN [7]) and the starting point of the code of our project. It allows external applications to control one single character in the game. Using this interface developers can implement external controllers, coding them in arbitrary languages (as Lisp or Java), for the game play. An extension to the game module called Neverwinter Nights Extender (NWNX [8]) is necessary, and also special scripts for receiving commands and passing them to the character, and sending facts (observations) that happen inside the world to the external controller.

3 Communication Protocol and Language

The RCEI communication protocol is blocking and synchronous, at the same time the sending of messages must be strictly alternate. First, the remote controller must begin the interchange between both extremes because, generally, the answers of the

environment will be *expected facts* -that will confirm the system's *commands*- or *unexpected facts* as consequence of long-term actions produced by the execution of the commands combined with the actions performed by the interactors and other autonomous entities of the fictional world. Second, the first message of the virtual environment must be an identification message, listing every condition referred to the initial state that the environment must know. And finally, the remote-controlled environment must be who finish the communications sending an ending message to the environment, although this also could be an unexpected error message when something goes wrong at that side of the connection or if the blocking of the controller delays too much.

The communication language uses a similar syntax for both commands and facts. All of them have a subject, that it may be an agent of the environment or the environment itself, and also all of them has a process in order to notify the kind of action, which can be relative to the environment or any object or agent. The other fields in the command (or fact) are variable according depending on the parameters needed for each type of instruction.

There are three special pairs of meta-commands and meta-facts with a different structure; these are *begin*, *end* and *synchronize*. These commands have a subject and a predicate with a process, and two new parameters: a list of concepts and a list of relations. The functionality of these instructions -when they are commands- is starting or ending a new game performing changes (creating, deleting, moving…) the available elements of the environment. When these instructions are facts, they are useful to report the state of the environment at the beginning, at the end or at any significant moment –determined by the AI system- during the game session.

While the messages sent by the environment are interpreted as facts that have been produced recently in the environment, the messages sent by the remote controller are interpreted as "events that should happen in the environment as soon as possible", with no guarantees due to real time and non-deterministic constraints.

The basic repertory of RCEI vocabulary contains instructions relative to the simulation domain of the environment, being its expressiveness inspired on the Conceptual Dependency Theory (CDT [9]).

Firstly, there is an instruction called *speak* to perform conversations between two agents. If we want to perform more advanced conversations, we also can use the instruction *change* to modify the state of characters involved in the dialog (emotion, attitude, mood, etc.).

The instruction *move* represents values of action, gestures or grimaces of one agent with respect to a determined location. Also it represents values of expression of an agent's action with a determined meaning. For example, make gestures of anger or happiness, execute animations of social or complex activities, etc. The values of agent's movements towards a location are represented by the instruction *go;* if this instruction has no parameters it means the agent must perform a random walk.

Create and *destroy* instructions allow both creation or destruction of a location in a determined situation (linked to other locations), an agent or object, links between locations, etc.

The *change* instruction mentioned before has a great versatility for the simulation. This instruction is used for change the climate or ambient conditions of the virtual environment. It is also used to change the objects' state of things like doors, chests, weapons, lights sources, etc.

Other instructions as *attack*, *take*, *drop*, *give*, *equip* and *unequip* have been included for practical reasons and due to the game-like orientation of many stories.

4 Implementation

The RCEI language have been implemented using Java and XML technologies in an API we called jRCEI. This distribution is available on the Internet as an open source project composed by different packages such as the parser, the serializer, the API itself, a GUI for manual control and some additional testing code, including the NWN to RCEI adapter and an example scenario.

Fig. 1. Sequence diagram and screenshot of an example execution. The KIIDS system is monitoring and controlling what is happening in the RCEI test environment.

The example scenario has a set of models ready to be used in the story. After establishing the connection with the RCEI-compatible environment running in a NWN server, the KIIDS system sends the begin command. In this command, the player (dressed as a monk), a goblin, an elk and many objects (chests, flags, doors...) are described.

The environment creates everything without problems, so it returns a successful begin fact. After a short time reasoning, KIIDS sends a go command for the elk to move in random directions.

Nothing interesting happens until the player decides to go to the chest. When KIIDS incorporates that fact to its internal world model, the system reasons about it and decide that the goblin must avoid that, so it orders the goblin to attack the player.

Immediately the goblin attacks the player (note that it implies low-level decisions taken by the NWN engine, as path planning and choosing the best attack form), and the development of the story continues for KIIDS, receiving (expected or unexpected) facts and sending commands until it decides to send the end command.

5 Conclusions

Looking for a connection mechanism to make narrative environments and intelligent controllers interoperable, the KIIDS project includes a development related to a standard interface called RCEI. The objective of this interface is to provide a general engineering toolkit for building IDS applications offering flexible and efficient communication between the main system and the dramatic scenario.

The language proposed offers a basic but powerful vocabulary to communicate with the environment. While the interface is valid for both plot-oriented or character-oriented projects, now the tests have been done with the plot-oriented KIIDS director in mind. This only means that there are no special high-level commands designed for autonomous actors. Every instruction is simple, so characters, objects and the environment itself receive the same kind of information.

Extensions to the basic repertory of RCEI vocabulary are possible and simple to add. RCEI adapters at both sides of the connection must include those additions and must use them propertly. With respect to the proposal of extensions to the language, there is an important restriction: no addition of new simulation instructions can become RCEI dependent on too specific environments' or systems' specifications.

As future work we are designing new adapters for other environments based on the Neverwinter Nights 2 and Unreal Tournament 3 engines, at the same time we deals with distributed architectures and performance issues. RCEI is planned to be included in the final distribution of KIIDS, being completely open to the contributions of the research community.

Acknowledgments

This research is funded by Ministerio de Educación y Ciencia (TIN2006-14433-C02-01 project), Universidad Complutense de Madrid and Dirección General de Universidades e Investigación de la Comunidad de Madrid (UCM-CAM-910494 research group grant). Second author held a Beca-Colaboración 2006-2007 grant from Ministerio de Educación y Ciencia.

References

1. Peinado, F., Navarro, A.: RCEI, A Remote-Controlled Environments Interface (2007), http://federicopeinado.com/projects/rcei
2. Peinado, F.: Knowledge-Intensive Interactive Digital Storytelling system (2007), http://federicopeinado.com/projects/kiids/
3. Magerko, B., Laird, J.E.: Building an Interactive Drama Architecture. International Conference on Technologies for Interactive Digital Storytelling and Entertainment. Darmstadt, Germany (2003)
4. Young, M., et al.: Zocalo (2007), http://zocalo.csc.ncsu.edu/
5. Vernieri, T.M.: Web Services Approach to Generating and Using Plans in Configurable Execution Environments. M.Sc. thesis, Liquid Narrative Group. North Carolina State University. Raleigh, NC, USA (2006)

6. Zubek, R.: Shadow Door: Neverwinter Nights NPC Control Interface (2003), http://www.zubek.net/robert/software/shadow-door/
7. BioWare: Neverwinter Nights: Diamond Compilation Pack (DVD-ROM). Atari (2005)
8. Stieger, I.: Neverwinter Nights Extender (2004), http://www.nwnx.org/
9. Lytinen, S.L.: Conceptual Dependency and its Descendants. Computers and Mathematics with Applications 23(2-5), 51–73 (1992)