JOURNAL OF KOREA MULTIMEDIA SOCIETY, VOL. 6, NO. 4, JULY 2003

INTERFACE DEVELOPMENT ENVIRONMENT BASED ON CHARACTER AGENT

YoungMee Choi and MoonWon Choo

ABSTRACT

We describe a scheme for developing character-based interface within the context of an agent-based tutoring system in the Web environment. The ideas in this paper stem from original work representing aspects of human emotion in tutoring computer models, where may provide more natural ways for students to communicate with digital learning materials. The proposed system model is a set of software services that enable developers to incorporate interactive animated characters into their Web pages designed for on-line lectures. The prototypical application is developed and shown for validating the applicability and the effectiveness of this model in real tutoring settings.

Key words: software agent, character agent, Microsoft Agents, tutoring system, conversational interface

1. INTRODUCTION

Software agent provides unprecedented technology to create new conversational interfaces for applications and Web pages. It provides powerful animation capability, interactivity, and versatility.

Many applications have developed computer agents capable of plausibly appraising situations that arise in their world so that they respond with a wide variety of emotion states such as pity, anger, joy, and admiration. The agents are then capable of expressing these emotions through a variety of theoretical paths, resulting in various multimedia manifestations[4].

When proposing to add such an affective components to tutoring systems[3], two primary issues must be addressed[6]. Firstly, the complexity of animated agent may or may not give sufficient and valid coherence. We have current systems which do little or nothing with rich personality models, but which are clear in their objectives. Secondly, there may be lots of questions about the correct balance between entertainment and pedagogical principles.

However, many systems have been proposed the socially complex systems which risk having the social interaction get in the way of the pedagogical tasks, but show promise as interesting, engaging, tutors. Also it is possible for us to obviate the balance issues by integrating the entertainment value into the manipulation of tutoring contents itself using social principles[2,5].

People have built-in capabilities for understanding social and emotional complexity. Agents that are able to capitalize on these capabilities will be able to initiate complex social interactions in the tutoring paradigm without sacrificing coherence. Furthermore, tutoring goals and principles can comprise the fabric from which an agent's emotion reasoning arises, allowing students a high degree of integration between, on the one hand, entertainment through interaction with the agents, and on the other hand, the underlying material in a content domain.

These capabilities advance beyond those of existing traditional tutoring systems, enabling more intelligent and human-like on-line learning

* *The authors are with the Division of Multimedia, Sungkyul Univ. #147-2, anyang-8 Dong, Manan-Gu, Anyang-City, Kyunggi-Do 430-742, Korea.*
E-mail : {choiym, mchoo}@sungkyul.edu

to be realized and facilitating students to work in a more natural and productive fashion.

We propose an Interface Development Environment based on character agents that assists tutors in developing Web-based teaching materials, exploiting emotional agent paradigm supported by Microsoft Agent Services. In this environment, tutors can add affective and customized character agents to their lecture note interactively. The character agent gives appropriate tutoring advices in teaching procedures. We adopt this simple approach, since the full-fledged emotional tutoring agents are still not available. In addition, the conventional interface approaches facilitated by many existing software may not be replaced for a while. In our methodology, character interaction can be easily blended with the conventional interface components, which may enhance the application's interface.

This paper is organized as follows. Section 2 presents the design concepts and characteristics of Microsoft agent, and its related issues of command coding methodology. In section 3, the experimental development environment comprising of character image editor and character command generator (CCG) is proposed. Section 4 presents the architecture of CCG that we have implemented. In section 5, simple experiment is demonstrated by applying the proposed scheme to online lecture note. Then we conclude with some ideas for further development in section 6.

2. MICROSOFT AGENT (MA)

MA is an interface technology that can be utilized as part of Web pages and conventional applications. It enables developers to display and animate an interactive character, and also provides the environment to compile character animations and speech input and output.

Developers can use characters as interactive assistants to introduce, guide, entertain, and enhance Web pages or applications in addition to the conventional use of windows, menus, and controls. MA enables Web authors to incorporate a new form of user interaction, known as conventional interfaces[1,2,5]. The conversational interface approach facilitated by MA services does not replace conventional graphical user interface (GUI) design. Instead, character interaction can be easily blended with the conventional interface components such as windows, menus, and controls to extend and enhance the application's interface.

MA's programming interfaces make it easy to animate a character to respond to user input. Animated characters appear in their own window, providing maximum flexibility for where they can be displayed on the screen. MA includes an ActiveX control that makes its services accessible to programming languages that support ActiveX, including Web scripting languages such as Visual Basic. This means that character interaction can be programmed even from HTML pages.

2.1. Issues of MA command generator

Commands represent agent's animation services to manage the animation and movement of a character's image in its own window on the screen. An animation is defined as a sequence of timed and optionally branched frames, composed of one or more image[7-10].

To animate a character, MA supports several commands.

The **Load** is used to load the character's data. The **Show** method makes the character's frame visible and plays the animation assigned to the character's showing state. The **Play** method can be used when the character's frame is visible. The **Get** (or **Prepare**) method is for firstle retrieving the animation data. This will cause Agent to request the animation file from the server and store it in the browser's buffer on the local machine. The **Speak** method is for programming the character to speak, automatically lip-syncing the output. The **MoveTo** method is used to position the character

at a new location. When it is called, MA automatically plays the appropriate animation based on the character's current location, then moves the character's frame. The **GestureAt** is called to make agent play the appropriate gesturing animation based on the character's location and the location specified in the call. The **Hide** method is called to hide the character. This automatically plays the character associated with the character's hiding state, then hides the character's frame. However, by setting the character's **Visible** property, a character also can be hidden or shown.

MA's enable you to animate characters independently or use the **Wait**, **Interrupt**, or **Stop** methods to synchronize their animation with each other. MA also plays other animations automatically for you. for several seconds, Agent begins playing animations assigned to the character's idling animations. Similarly, when speech input is enabled, Agent plays the character's listening animations and then hearing animations when an utterance is detected. These server-managed animations are called *states*, and are defined when a character is created.

If multiple clients are sharing the same character, the server will designate its *activeclient* as input-active client. You can set your client to be the active or not-active client using the **Activate** method. You can also use the **Activate** method to explicitly make your client input-active; but to avoid disrupting other clients of the character, you should do so only when your client application is active.

Many other commands are available in MA services such as SAYNOCAP(to say without text), AUDIO(to sound), AUDIONOCAP(to sound without text), THINK, LOOP, SLIDE(to shift to next slide), SUSPEND(to rest), END(to end), COMMENT, RUN, NEXT, SOUNDFXON(to turn on sound effect), SOUNDFXOFF(to turn off sound effect).

The **SIZE** controls the size of character, which is added to the MA list of commands. In our work, several commands are modified and created.

For example, an agent is saying that I can act as a sales representative or lead you through a series of questions to arrive at a final decision, while pointing to right-hand side and moving to the coordinate (30,20) with doubling its size. Then the list of commands could be as follows (Fig.1):

The MA commands are very useful services to develop the conversational interfaces. There are several shortcomings in MA environment(see Fig. 2). But in real settings, it is not easy to properly verify the agent's behavior. The repeated modification is required to locate the agent's exact positions on the screen. Also the internal code should be modified when multiple foreign languages are used in same session.

```
PLAY GestureRight
SAY I can act as a sales representative or lead
you through a series of questions to arrive at a
final decision.
MOVE 30,20
SIZE 200%
```

Fig.1. List of commands.

Function	Description
animation	hard to verify the resultant behaviors of agent in develping session
movement	repeated modification is required to fix the character positions on the screen
language	very hard to modify internal codes for beginners to support multiple languages in same session

Fig. 2. The relative shortcomings of MA environment

3. THE PROPOSED MODEL

The conversational interfaces supporting character agent can not be easily developed without providing WYSIWYG environment that can help developers design and develop characters, and generate the proper commands. The proposed model(see Fig. 3) includes conceptual and technical information on character, image, and animation

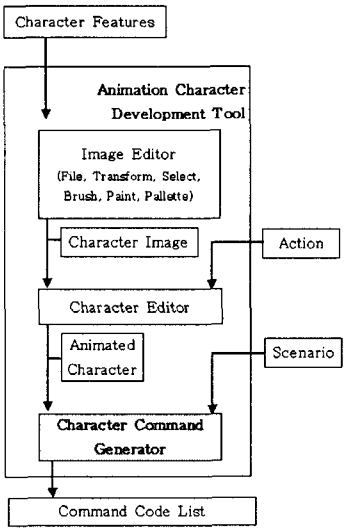

Fig. 3. The proposed model for interface development environment

design; the size, use of color, and types of images you need to create; suggested animations; speaking animations; and agent states, and menu-driven developing tools.

One of the most important issues of this model is how support the way to design visually convincing animated characters to fit for given teaching materials and educational goals. Human communication is fundamentally social, so students will expect a character to conform to the same social, though not necessarily physical, rules they use when interacting with other people, even when they understand that the character is synthetic.

When designing a character, the profile of target audience may be considered first, and what appeals to them as well as what tasks they do. Hence the model should provide easy way to solve this problem. One solution is to provide many optional characters to be selected, otherwise image editor should be imbedded for creating customized agent.

Also the model considers the way to generate animation according to character's basic personality type: dominant or submissive, emotional or reserved, sophisticated or down-to-earth. So an agent's emotion could be implemented more realistic and life-like.

In addition, the model provides the way to synthesize the voice(using a text-to-speech engine) and record the tutor's voice(.WAV file). This module may depend on the type of character to be used, the languages to be supported. The model can provide the way to use the word balloon for output and the default settings for the balloon's font and color.

4. THE ARCHITECTURE OF CCG

The Character Command Generator(CCG) that we have implement is explained in detail. This module provides the way to generate commands for animation and preview for editing with given character and scenario which is the contents for teaching. The output of this module is the command code list, which animates the character. This CCG is more detailed as Fig. 4.

The CCG generates the commands by following procedure.

① accept the character selected
② load the selected character
③ select the components of behaviors according to scenario
④ preview the selected behavioral components
⑤ add/delete the components
⑥ preview the command list
⑦ edit the final behavior (delete/modify)
⑧ output the command list

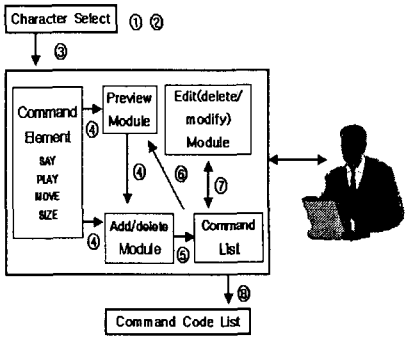

Fig. 4. The conceptual structure of CCG

Each component of user interface is commented briefly. Fig. 5 shows the user interface, comprising of 6 sections for editing and 1 command viewer. The scenario is assumed MS PowerPoint slides here. The resultant command list can be copied and pasted to the slide note of each slide, then can be checked character animation in real-time.

The SAY window(Fig. 6) accepts(1) the text to be used for balloon tips and speech(4). The ADD(2) is clicked to add the speech command to command list(3), which will be pasted to slide note concerned.

This window(Fig. 7) lists all commands for animation/play. The selected command are added to command list(3) and resultant animation(4) is previewed on this interface when the Insert button(2) is clicked.

The MOVE window(Fig. 8) sets the coordinates of start(1) and destination(2) positions within the maximum dimension of screen. If the character moves, the coordinates should be changed. The Add button(3) is for adding the coordinates to the command list(4). The character's positions are

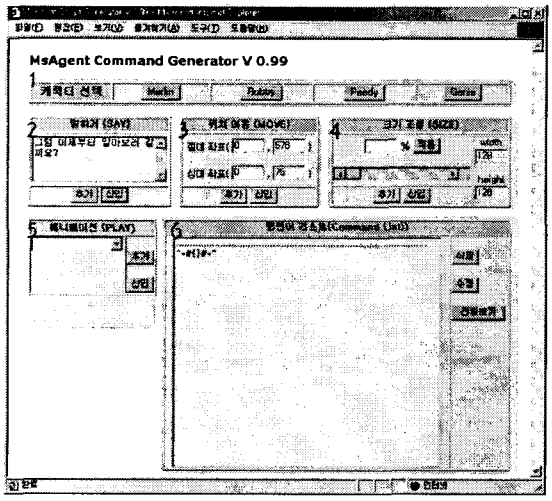

Fig. 5. User Interface of CCG

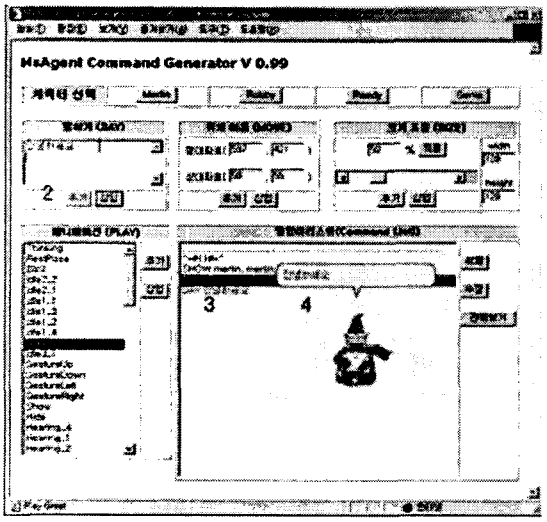

Fig. 6. SAY window

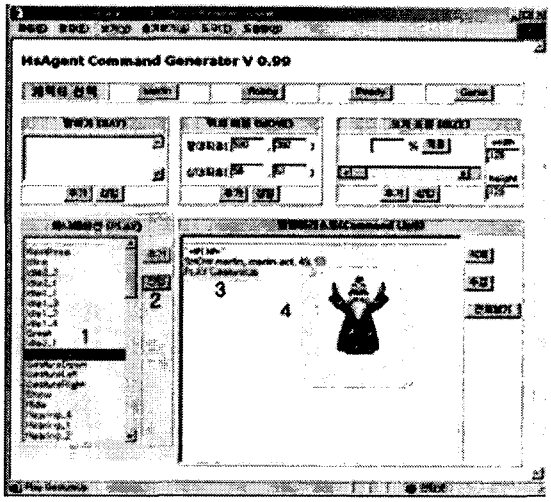

Fig. 7. ANIMATION(PLAY) window

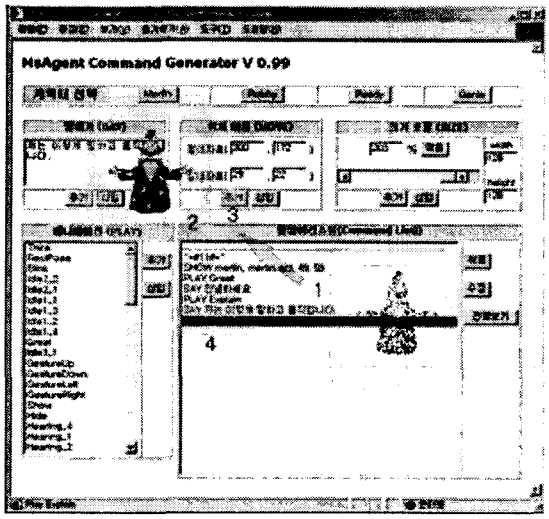

Fig. 8. MOVE window

described by the ratio of screen size, so the screen resolutions should be set consistently.

This window(Fig. 9) controls the size of agent image. The size is set by using horizontal scroll bar(2). After checking the ratio of expansion, click the Add button. Then the command are generated and added to the command list(7).

This window(Fig. 10) has standard editing functions such as add, delete, modify, and etc.

5. IMPLEMENTATION

The proposed model is applied to the lecture notes on History on Western Art. Fig. 11 shows the main interface for prototyping this example.

For example, the following command list is generated and applied to one of PowerPoint slides (see Fig. 12).

SHOW *bear, bear.acs, 83, 80*

SAY *now, I will introduce Leonardo da Vinci.*

PLAY *RestPose*

SAY *Leonardo da Vinci is a great scientist, artist and engineer representing the Italy in Renaissance Age.*

MOVE *40, 60*

SAY *His genius talent is shown in a lot of his great works. Among them, Mona Lisa and Baptist John, are well known.*

SAY *This picture is titled as Baptist John.*

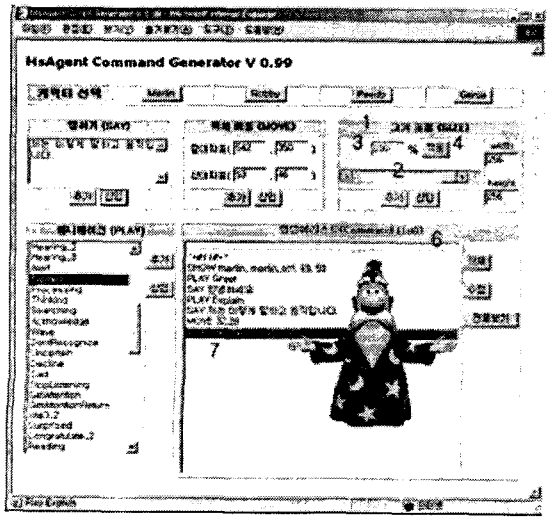

Fig. 9. SIZE window

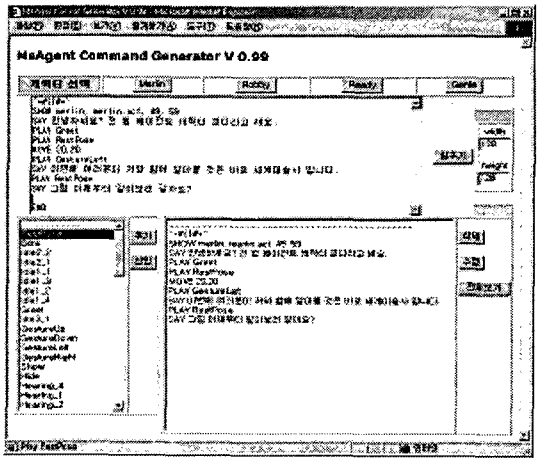

Fig. 10. Command List Editor window

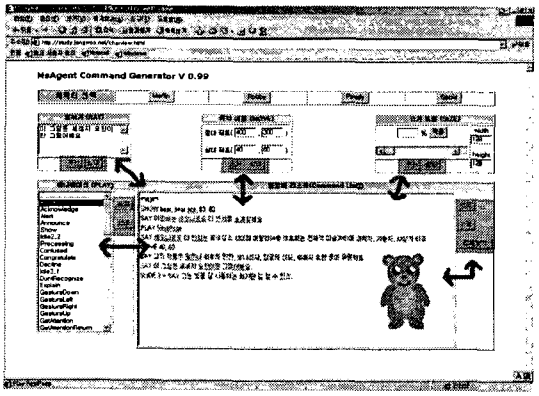

Fig. 11. The interface for implementation of example case

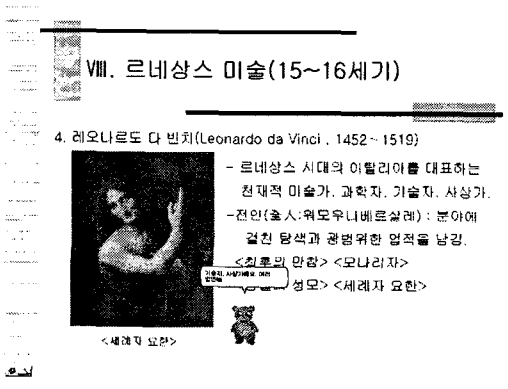

Fig. 12. The resultant slide with character agent

6. CONCLUSION

Software agent provides powerful technology to create new conversational interfaces for applications and Web pages. When proposing to add such an affective component to tutoring systems, two primary issues must be addressed: complexity versus coherence and the balance between entertainment and pedagogical principles. We believe that such issues are not fully addressed and resolved. In addition, the conventional interface approaches facilitated by many existing software may not be replaced for a while.

We propose an Interface Development Environment based on character agents. This model assists tutors in developing Web-based teaching materials, exploiting emotional agent paradigm. In this environment, tutors can add affective and customized character agents to their lecture note interactively. The character agent gives appropriate tutoring advices in teaching procedures. In this model, character interaction can be easily blended with the conventional interface components, which may enhance the application's interface.

The prototypical application is developed using this environment. We realize that this system can give the online digital contents quickly and efficiently. Also the conversational interface may enhance the learning motivation more than conventional teaching materials.

However, the proposed model had many limitations. First of all, this is just test-bed, not fully-developed package, so the comparative study on performance is not possible. Secondly, the multimedia components, such as video, audio, background music and special effect, should be dealt with properly. Most of all, the emotion and its related behaviors are fully studied in this work. These issues will be main topics of our future research.

10. REFERENCES

[1] Arthur C. Graesser, *AutoTutor: A Simulation of a Human Tutor, Journal of Cognitive Systems Research,* 1, pp. 35-51, 1999.

[2] Brian A. Stone and James C. Lester, *"Dynamically Sequencing an Animated Pedagogical Agent",* Readings in agents, pp156-163, 1998.

[3] Burton, R. and Brown, J., *"An investigation of computer coaching for informal learning activities,"* International Journal for Man-Machine Studies, Vol.11, pp.5-24, 1979.

[4] Hyacinth S. Nwana, *"Software Agents: An Overview," Knowledge Engineering Review,* Vol. **11**, No 3, pp.1-40, Sept 1996.

[5] Justine Cassell, *"Animated conversation: Rule based generation of facial expression, gesture & spoken intonation for multiple conversational agents",* Readings in agents, pp148-155, 1998.

[6] Rosalind W. Picard, *Affective Computing,* The MIT press, 1997.

[7] http://www.bellcraft.com/mash

[8] http://www.extempo.com

[9] http://www.microsoft.com/msagent

[10] http://www.msagentring.org

YoungMee Choi

1979 Ewha Womans University B.S. (Mathematics)
1981 Ewha Womans University M.S. (Computer Science)
1993 Ajou University Ph.D. (Computer Engineering)
1994~Present Associate Professor, Division of Multimedia Sungkyul University

MoonWon Choo

1986 San Jose State University B.S. (Mathematics)
1987 New York Institute of Technology M.S.(Computer Science)
1996 Stevens Institute of Technology Ph.D. (Computer Engineering)
1997~Present Assistant Professor, Division of Multimedia Sungkyul University

For information of this article, please send e-mail to:choiym@sungkyul.edu(YoungMee Choi)