A TWO-MODEL FRAMEWORK FOR MULTIMLDAL PRESENTATION WITH LIFE-LIKE CHARACTERS IN FLASH MEDIUM

Zhenglu Yang Nakasone Arturo Adam Jatowt Mitsuru Ishizuka
Department of Information and Communication Engineering
Graduate School of Information Science and Technology
University of Tokyo
7-3-1, Hongo, Bunkyo-ku, Tokyo 113-8656, Japan
{yang, arturo, jatowt, ishizuka}@miv.t.u-tokyo.ac.jp

Abstract

Multimodal presentation using interactive life-like agents is an attractive and effective way to edify, assist and guide the users. However, it is not easy for many people to write such multimodal presentations, due to the complexity of describing various behaviors of character agents and their interaction of particular character system with individual (often low-level) description language. As part of our research, we have developed MPML (Multimodal Presentation Markup Language), which allows the users to write multimodal contents with ease. By extending MPML, we have proposed a new framework for an interactive system that fulfills multimodal presentation through script-based and character-based models in Flash medium.

Key Words

Multimedia Systems, Presentation Markup Language, Life-like Agent

1 Introduction

A presentation, slide show, video clip or a poster may contain all the relevant information, but their success in conveying the information depends mainly on the presence of a human presenter and the way in which the presentation material is used. The presence of a skilled presenter in addition to a well-prepared presentation material usually makes the presentation more appealing.

In the last few years, an increasing number of attempts have been made to develop agent authoring systems that are able to generate agent applications like presentations on the fly. In order for such presentations to be created successfully, agent authoring systems that are feature rich and user-friendly are necessary. Traditional agent authoring systems have been focused on generating one-way presentations that merely "push" information to the user. Such presentations do not take into account customizing of presentation content in response to user needs. More recent developments have led to presentations with increased user interactivity - an advantage that animated agent based systems have over human based presentations. This could take the form of allowing users to interact with the presentation materials, or having the users interact directly with the agent presenter. By allowing such interactions, user interest can be significantly increased by permitting a greater amount of participation. In this paper, we will present our MPML-Flash framework in which we employ two models to create multimodal presentation: the script-centered model and the character-centered model. We use MPML as the script language and Finite State Machine as the technique to control autonomous character, created basing on the pre-rendered images, in the popular Flash medium.

The rest of the paper proceeds as follows. Section 2 describes related works about character agent. Section 3 describes the core components of the system like MPML, character animation, Finite State Machine control mechanism, Flash platform, and chatterbot. Section 4 describes the system structure according to two models, script based model and character based model. Section 5 will show the demo of our system and Section 6 presents a conclusion and future work.

2 Related Works

Similar to our framework, the Improv system [1] is mainly controlled by behavioral scripts designed to be easily translated from a given storyboard. The Jack Presenter [2] has a series of powerful animation engines, also considering authoring issues. The Behavior Expression Animation Toolkit (BEAT) [3] facilitates synchronization of non-verbal behavior and synthetic speech.

The systems described above have the disadvantage of pre-defining the presentation flow. In case of our system, not only MPML is used to script the presentation, but also other techniques are employed like Chatterbot, Non-Deterministic FSM for constructing a character based, in some sense, automatic virtual environment according to the users in real time. Because we fulfill all of these in Flash medium, the framework is especially useful in network environment better than the other systems described.

3 Core Components of the System

3.1 Multimodal Presentation Markup Language (MPML)

3.1.1 Introduction

The Multimodal Presentation Markup Language [4] developed at the Ishizuka Lab. Tokyo University, is an XML styled markup language designed to allow authors to easily script animated agent presentations. At present, MPML can control a talking character head, which is created using physics-based technique, a VRML character, which has a H-ANIM compatible body in a 3D environment, and the popular Microsoft Agent. Not only the character, but also the background objects can be controlled in the virtual reality. In this paper, we propose a new Flash character creating system scripted by MPML.

3.1.2 MPML Architecture

Fig.1 shows the overall architecture of MPML3, which is the newest version of the language. The MPML script is first checked by the script loader and then converted into a graph, which is displayed in the graphical user interface. The author can easily modify the graph using this interface. It is also possible to create the graph from scratch and save it as an MPML file. Finally, the role of the converter is to convert the graph into executable code which directly controls the actual presentation.

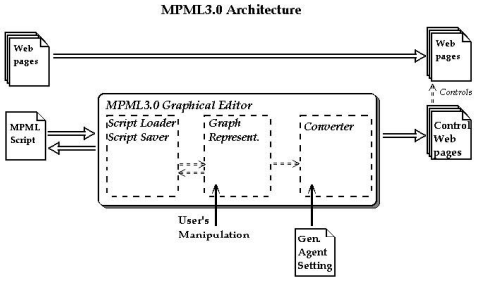

Fig.1 MPML3 Architecture

3.1.3 MPML Specifications

The MPML Language consists of 24 tags divided across three main domains: Agent Control, Presentation Structure Control and Background Control. They are described as shown below:

- Agent Control: These tags are used to determine the actions of the agent, like speaking, thinking, playing and moving. These tags are also able to control the agent's mood and emotions.
- Presentation Control: These tags control the overall flow of the presentation, either by allowing agents to act in sequence or simultaneously.
- Background Control: The background is an important aspect of the presentation material used by the animated agent. The tag names include wait, consult, test and execute.

3.1.4 Graphical User Interface

To make it easier for the author to write MPML script, we have developed a graphical user interface (GUI). A screenshot of the interface is shown in Fig.2. The rectangles denote the background pages, each differentiated by a different color. The tiny faces next to the boxes represent the active agents, which are also differentiated by color.

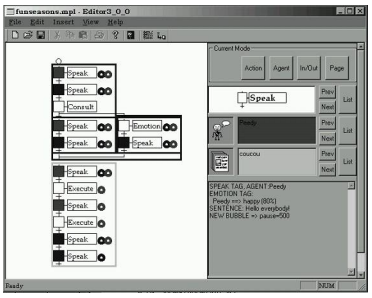

Fig.2 GUI of MPML

3.2 Character Animation

Character animation has a long tradition in computer graphics. The three main techniques used in this area are keyframing [5], procedural methods [6] and the use of data captured from the real world [7].

A disadvantage of the described techniques is the need for accurate 3D models of the characters to animate. Image-based rendering [8] avoids such models by using images available from hand-drawn or pre-rendered. It can utilize a variety of animation styles, e.g., different frame rates, artistic styles and media. In our system, we apply image-based rendering technique.

3.3 Lip Synchronization

To synchronize the character's lips, we use some techniques based on phonemes and visemes. Phonemes are the distinct sounds within a language. Visemes are visually distinct mouth, teeth, and tongue articulations for a language counterpart to phonemes.

In our system, we employ Festival Speech Synthesis System [9] to produce the audio stream that will be subsequently played back in synchronization with the facial animation and the temporized phonemes, which are used to create the corresponding visemes to fulfill the lip synchronization.

3.4 Finite State Machine

To control the character, we use a classic technique called FSM (Finite State Machine), based on the works of MediaSemantics [10].

There are two main types of FSM. The original simple FSM is known as deterministic one, meaning that given an input and the current state, the state transition can be predicted. D-FSM is employed in the pre-script centered model of our system. An extension of the concept at the opposite end is a non-deterministic finite state machine. This is where given the current state, the state transition is not predictable. ND-FSM is used in the character centered model of our system.

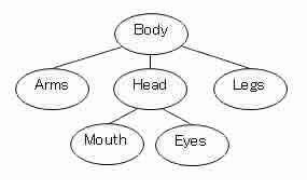

Fig.3: Body FSM structure

Fig.3 illustrates the structure of a Body FSM. The body of character is divided into several parts. The parent image is rendered with a "hole" punched out of it, and then the child image is drawn into that hole. Different child parts, such as the mouth and eyes, can be drawn onto the same head parent part. This work is done manually to define the size of the hole now. We plan to use computer vision technique in the near future to automatically identify the different parts of the character.

State/transition Diagrams

Fig.4 is a simple state machine that defines the turning action of the character. The circles represent states and a character can remain in a state for any period of time. The arrows denote transitions which contain several frames that will bring a character from one state to another.

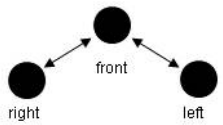

Fig.4: Turning FSM

Within a given body state, we can use more state machines to describe other parts of the character to do some animation. For example, we can define a "looking" state machine and a "speaking" state machine for addition. The Fig.5 has three state machines.

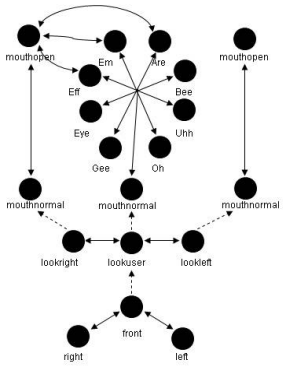

Fig.5: Speaking FSM

Implementing FSM

The character's animation library is an XML version file in which are defined all the states and transitions of the character. For example, our simple turning model is expressed as follows:

<state id="Right" />

```
<state id="Front" />
<state id="Left" />
<transition fromstate="Right" tostate="Front" />
<transition fromstate="Front" tostate="Left" />
<transition fromstate="Left" tostate="Front" />
<transition fromstate="Front" tostate="Right" />
```

When attaching child states to parent states, we must connect the parent state to the default child state, and connect all child states back to the parent state. Our "Looking" hierarchy would be expressed as follows:

```
<state id="LookUser"
parentstate="Front,Right,Left"    part="head"
mouthstate="FrontUserMouthNormal"
eyestate="FrontUserEyesNormal" />
<state id="LookRight"
parentstate="Front,Right,Left"    part="head"
mouthstate="RightUserMouthNormal"
eyestate="RightUserEyesNormal" />
<state id="LookLeft"
parentstate="Front,Right,Left"    part="head"
mouthstate="LeftUserMouthNormal"
eyestate="LeftUserEyesNormal" />
```

3.5 Dynamic Flash Creation

To create Flash, we employ the client-server model that Flash scenario is created at the server side by the author's script or dynamically according to the user's request.

Among the server-side Flash creation toolkits available, we use Ming toolkit [11]. Ming is a sourceForge project, C library for generating SWF ("Flash") format movies. It can be used with PHP, Python, Ruby and C++ languages. Using the Ming libraries, we can create complete SWF files from scratch with server side scripting on the fly.

3.6 Chatterbot

For making a live presentation like it happens in a real life, the presenter needs not only to write the whole plot of the presentation (script-based), but also to prepare the answers according to the different questions which may be asked by the audiences (character based). To simulate a real scene, we should build a speech based chat system, but unfortunately, because of the inherent difficulties of speech recognition, which by itself is a highly complex problem, we are now only using a textual interface (chatterbot) to fulfill the free question-answer scenario.

Chatterbots are conversational agents engaging in a natural language-based interaction with user. In our system, we use AIML [12] as our chatterbot language.

3.6.1 Introduction of AIML

AIML is a XML-like language, designed for ALICE (Artificial Linguistic Internet Computer Entity) chatterbot based on the pattern-matching and Case-Based Reasoning techniques.

A simple example of random responses of AIML in our system is as follows:

```
<category>
<pattern> HELLO</pattern>
<template>
<random>
<li>Well hello there!</li>
<li>Hi there!</li>
<li>Hi there. I was just wanting to talk to you.</li>
<li>Hello there!</li>
</random>
</template>
</category>
```

The category tag contains following tags: a pattern tag that specifies the pattern to be matched, a template tag, which defines a response or set of responses to be given, and a random tag, in which <li> tag is used to define random answers to the same question.

3.6.2 Domain Model

As is true of nearly any computer program, the narrower the context, the higher the probabilities that the project can be completed successfully. Particularly in the field of AI it has been found that it is extremely difficult to produce a program to solve general problems [13]. The notion of domains and domain knowledge has allowed computer scientists to address more reasonable problems.

In our system, we have implemented a two-domain model [14], which determines whether the user is conversing within the domain of the presentation topic or whether he is speaking out of topic. By this way, we have rectified the embedded improper characteristic of Chatterbot, from diverse topic to specific presentation topic.

4 System Structure

We will present system structure according to the two models of the system, pre-script based model and character based model.

4.1 Pre-script Based Model

As shown in Fig.6, the author first creates MPML at the client side and then transfers it to the server, where the corresponding Flash is created based on the techniques presented earlier. When different users request from their own client side some time later, the server will send the same pre-script Flash to them.

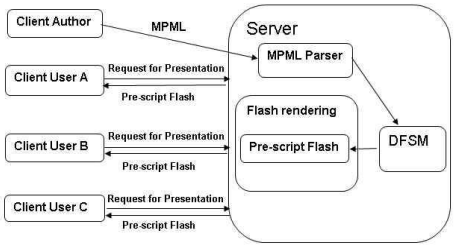

Fig.6: Script based system structure

The detail work flow of the Flash rendering is shown as Fig.7. The MPML parser assigns the work to different parts such as the background rendering, the character body rendering and the Text-To-Speech according to MPML script. Based on the phonemes, which are created from input texts, the system synchronizes the lip of the character and uses FSM to model the character. At the end, it combines the three rendering parts to get the final rendered Flash presentation.

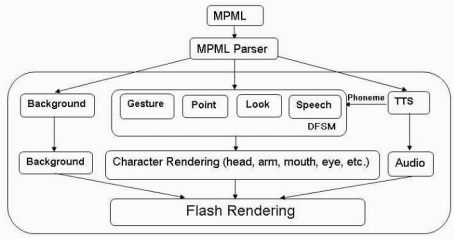

Fig.7: Script based system work flow

4.2 Character Based Model

In this modal, we hope the character acts like a real person based on his own judge in the virtual circumstance not on predefined plot. In other words, it means an autonomous character behaves properly to the users. We are now or will employ several methods to achieve this attractive but difficult goal.

First we employ the natural language processing technique, say chatterbot, as the "brain" of the character. Second we use non-deterministic finite state machine method, random actions like blinking or flexing hand, to the character to achieve the autonomous goal. The third technique we want to bind into our system is SCREAM[15], which is developed in our Lab, for generating emotion, regulating emotion and expressing emotion of the character. This part is under construct at present. Another technique we will employ to simulate life like autonomy is to make a link between gesture and text.

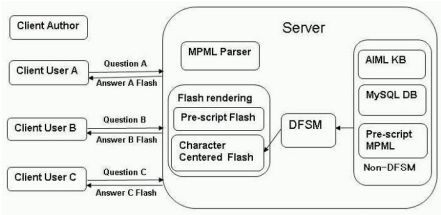

Fig.8: Character based system structure

As shown in Fig.8, different users should ask different questions to the server. Based on the domain knowledge provided by the author using AIML and general knowledge provided by system, corresponding answers will be created dynamically and sent back to the users. The detail work flow of the Flash rendering is shown as Fig.9.

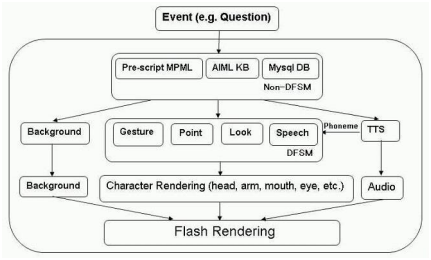

Fig.9: Character based system work flow

5 Demo

We present a live demo (Fig.10) on the web [4] to show our system.

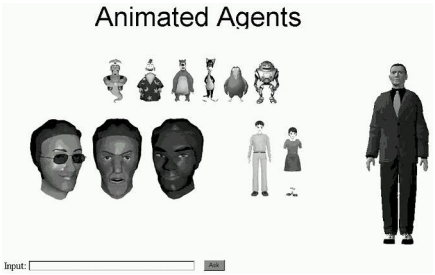

Fig.10 Demo snapshot

In this demo, we first define some texts and graphs in the <head> tag and then describe the representation flow in the <scene> tag.

6 Conclusion and Future Work

In this paper, we have proposed a new agent authoring framework in Flash medium. There are two models in our system. The first one is a script centralized approach that uses MPML which is created by authors to render the Flash presentation. Such a scripting approach facilitates the generation of well-structured and coherent presentations yet need to be created at first. And there are also situations that need to take into account customizing of presentation content according to different users. Therefore, the presentations dynamically change during display time. For this, we propose a character-based approach in which a dynamic presentation is rendered using underlying mechanism. Through this two-model system, we can simulate a virtual presentation scenario.

We are now combining another system developed in our Laboratory, named SCREAM, with our framework to let the character has his "emotion". Moreover, to act as a real human-like autonomous character, ND-FSM needs to be improved. Additionally, we plan to promote MPML-FLASH that uses MPML to control Flash which is created by Macromedia Flash Tool. We hope that by these methods we can create a virtual world that models the real one in which we live in.

Acknowledgments

This research is supported by the Research Grant (1999-2003) for the Future Program ("MiraiKaitaku") from the Japanese Society for the Promotion of Science (JSPS).

Appendix A: Detail of Character Creation

The prototype character is created by Poser software, then rendered to a raster format. To divide the character into parts, we use GD Graphics Library. For controlling it we employ FSM technique. Lip synchronization is achieved by using Festival Speech Synthesis System. Lastly, we use Ming/PHP to create the Flash scenario combining the character and background together on an Apache server.

References

[1] K. Perlin, and A. Goldberg, Improv: A System for Scripting Interactive Actors in Virtual Worlds, *Proceedings of SIGGRAPH'96*, New Orleans, USA, 1996, 205-216.

[2] N. Badler, R. Bindiganavale, J. Bourne, J. Allbeck, J. Shi, and M. Palmer, Real Time Virtual Humans, In *International Conference on Digital Media Futures*, Bradford, UK, 1999.

[3] J. Cassell, H. Vilhjalmsson, and T. Bickmore, BEAT: the Behavior Expression Animation Toolkit, In *Computer Graphics Proceedings*, Annual Conference Series. ACM SIGGRAPH, LA, California, USA, 2001, 477-486.

[4] MPML project, Retrieved March 8, 2003 from http://www.miv.t.u-tokyo.ac.jp/MPML

[5] D. Paul, Issues and techniques for keyframing transformations, In *Graphics Gems III*. Academic Press, Boston, USA, 1992, 121-123.

[6] J.K. Hodgins, W.L. Wooten, D.C. Brogan, and J.F.O'Brien, Animating human athletics, In *Proceedings of SIGGRAPH 95*, LA, California, USA, 1995, 71-78.

[7] Z. Popovic, and A. Witkin, Physically based motion transformation, In *Proceedings of SIGGRAPH 99*, Anaheim, California, USA, 1999, 11-22.

[8] I. Buck, A. Finkelstein, C. Jacobs, A. Klein, D.H. Salesin, J. Seims, R. Szeliski, and K. Toyama, Performance-Driven Hand-Drawn Animation, In *Non-Photorealistic Animation and Rendering Symposium*. Annecy, France, 2000, 101-108.

[9] A. Black and P. Taylor, Festival Speech Synthesis System: system documentation(1.1.1), Human Communication Research Centre Technical Report HCRC/TR-83, UK, 1997.

[10] Media Semantics, Retrieved March 3, 2003 from http://www.mediasemantics.com

[11] Ming sourceforge project, Retrieved April 6, 2003 from http://ming.sourceforge.net

[12] R. S. Wallace, Don't read me – A.L.I.C.E. and AIML documentation, Retrieved March 8, 2003 from http://alicebot.org/articles/wallace/dont.html

[13] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach(Upper Saddle River, NJ: Prentice Hall, 1995).

[14] K. Mori, A. Jatowt, M. Ishizuka, Enhancing Conversational Flexibility in Multimodal Interactions with Embodied Lifelike Agents, In *IUI'03*, Miami, Florida, USA, 2003, 270-272.

[15] H. Prendinger and M. Ishizuka, SCREAM: Scripting Emotion-based Agent Minds, In *Proceedings First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-02)*, Bologna, Italy, 2002, 350-351.