MPML-FLASH: A Multimodal Presentation Markup Language with Character Agent Control in Flash Medium

Zhenglu Yang 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, ishizuka}@miv.t.u-tokyo.ac.jp

Abstract

Nowadays, employing realistic-looking virtual humans as presenters in various situations becomes more effective and important. To make it easier for the author to write such multimodal presentation, we have developed MPML (Multimodal Presentation Markup Language). By extending MPML, we have developed MPML-FLASH (Multimodal Presentation Markup Language for Flash), which facilitates multimodal presentation in Flash medium. In this paper, we present the specification, related techniques and a framework as the testing platform for MPML – FLASH.

Key Words

Multimedia Tool and Systems, Presentation Markup Language, Life-like Agent

1. Introduction

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 these systems, scripting languages play important roles.

Scripting languages are to a certain extent simplified languages which ease the task of computation and reasoning. One of the main advantages of using scripting languages is that the specification of communicative acts can be separated from the programs which specify the agent architecture and mental state reasoning. Thus, changing the specification of communicative acts doesn't require to reprogram the agent [1].

For the purpose that people can write multimodal presentation easily, just as people can build homepage easily using HTML, we have created MPML. MPML is designed to create multimodal presentation contents independent of the character agents and currently it can control a talking character head, a VRML character, and the popular Microsoft Agent. In this paper, we will propose MPML-Flash script specification and the framework which is used as a test bed.

The rest of the paper proceeds as follows. Section 2 describes related works about character agent and script language. Section 3 describes the specification of MPML-FLASH. Section 4 describes the core components of test bed system like character animation, lip synchronization, Finite State Machine control mechanism and Flash platform. Section 5 describes the system structure. Section 6 shows the demo of our system. Section 7 makes an evaluation and Section 8 presents a conclusion and future work.

2. Related Works

Recently, interactive lifelike agents are employed to play key roles in virtual environments. We use these believable characters to act not only as virtual tutors [2], but also as virtual actors [3], virtual sales agents and presenters [4].

Similar to our MPML script, APML – Affective Presentation Markup Language [5] focused on presenting personality and emotions in agents. The presentation is mainly performed by speech synthesis and facial expression. VHML – Virtual Human Markup Language [6] allows interactive Talking Heads to be directed by text marked up in XML. The language is designed to accommodate various aspects of Human-Computer Interaction. As is the case with APML, it does not have a feature to integrate a background. Moreover, they only concern the head of the character, not the entire body.

In case of our MPML-FLASH language, the script is not only used to script the full body characters, but also is employed to control the background objects. Additionally, because we control virtual character and other background objects in Flash medium, the work is especially useful in network environment better than the other script language described above.

3. MPML-FLASH

3.1. Multimodal Presentation Markup Language (MPML)

3.1.1. Introduction. The Multimodal Presentation Markup Language [7] 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. In this paper, we propose a new Flash character creating system scripted by MPML.

3.1.2. MPML System Architecture. Figure 1 shows the overall system architecture of MPML3, which is the newest version of the language. Because of the space limited, the MPML3 specification is not shown here and is available at MPML homepage [7]. 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. Finally, the role of the converter is to convert the graph into executable script code, which directly controls the actual presentation.

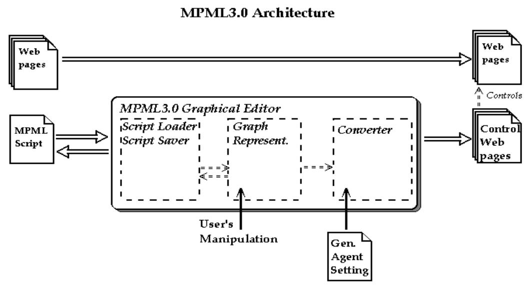
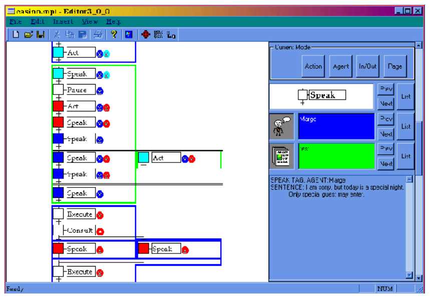


Figure 1. MPML3 architecture

3.1.3. MPML Specifications. The MPML Language consists of 24 tags divided across three main domains. They are described as shown below:

1. 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.
2. Presentation Control
   These tags control the overall flow of the presentation, either by allowing agents to act in sequence or simultaneously.
3. 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 Figure 2.



SPEAK TAG, AGENT:Marge
SENTENCE: I am sorry, but today is a special night.
Only special guest may enter.

Figure 2. GUI of MPML

3.2. MPML for Flash Medium

Based on the works of MPML before, we have developed MPML-FLASH, which is used in Flash medium to create multimodal presentations with life-like agents.

3.2.1. MPML-FLASH Design. The MPML before mainly uses Microsoft Agent as the character to present the scenario. Yet it has a drawback that the character file and the TTS engine should be first downloaded into the user's machine. This will take some time especially if the user has a slow network connection.

Based on the streaming technique, Flash can be viewed while downloading. So the user can see the Flash presentation even there is nothing but a Flash Player, which has already been used by 98% web users according to the Macromedia Company's report. Table 1 shows the difference of Flash and MS Agent.

Table 1. Comparison of MS Agent and Flash

	Main Function	Player	Platform	Speed	Character Creation	Size
Flash	Vector graphics & showy animation	98% web users already have	PC,Mac, Unix,PDA,m phone	Fast, streaming technique based	Difficult	Small
MS Agent	Character create tool	Need download	PC	Slow download Fast Execution	Easy	Big

The MPML system before is not clear that it is whether or not built based on Client-Server model. In case of MPML-FLASH, it creates all presentation Flash files at the server and users can request it from the client. This clear separation can take good use of the server as a powerful pool such as employing database to implement a dynamic presentation.

The MPML parser before is mainly converting MPML script to javascript and HTML code, which is used to control the character stored in user's machine. The MPML-FLASH parser is built based on DOM XML at the server. So no MPML-Javascript converting is needed.

3.2.2. Specification of MPML-FLASH

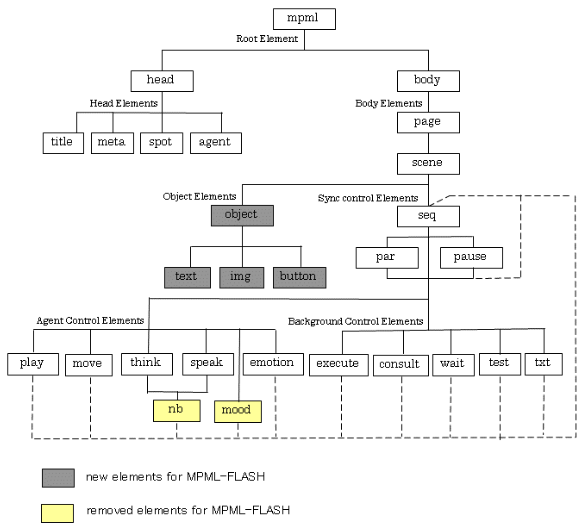


Figure 3. MPML-FLASH tag tree

The XML tag tree of MPML-FLASH is shown (Figure 3). The new elements in MPML-FLASH are described in the gray box. And the yellow box shows the removed elements in MPML-FLASH. Since the detailed specification is available at MPML-FLASH homepage [8], we note here the main modification from MPML3 to MPML-FLASH.

The new object elements define and control the text, image and button objects in Flash medium. The reason why we add these new tags is that we can control more Flash objects in the background so that we make the presentation more attractive. In the future we can also extend to other objects such as audio and video. The example codes of object elements are shown as follows:

```
<object type="text" id="Title1" x="0" y="100" height="50" src="Multimodal Presentation Markup Language" />
<object type="img" id="MPML_GUI" x="100" y="200" width="500" height="400" src="mpml3_gui.jpg" />
<object type="button" id="Stop" value="Stop" x="780" y="680" act="Stop"/>
```

The reason why we removed the tag <nb> is that it can be replaced by tag <pause> for the similar function of them. And we also removed tag <mood> because at present we don't focus on developing this part of the character. We only simplify the character by defining the tag <emotion> such as the following example code:

```
<emotion id="happy" name="happy" act="smile" />
```

The other tags in MPML-FLASH are the same as in MPML3, which is available at MPML homepage [7].

4. Character Agents for MPML-FLASH

4.1. Character Animation

Character animation has a long tradition in computer graphics. The three main techniques used in this area are keyframing [9], procedural methods [10] and the use of data captured from the real world [11].

A disadvantage of the described techniques is the need for accurate 3D models of the characters to animate. Image-based rendering [12] 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.

4.2. 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.

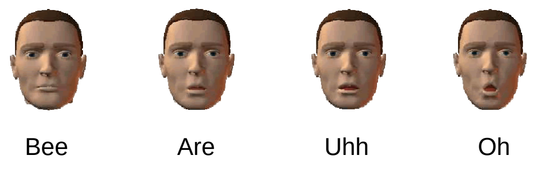


Figure 4. Phoneme & Viseme

In our system, we employ Festival Speech Synthesis System [13] 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 (Figure 4).

4.3. Finite State Machine

To control the character, we use a classic technique called FSM (Finite State Machine), based on the works of MediaSemantics [14].

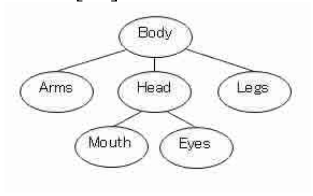


Figure 5. Body FSM structure

Figure 5 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

Figure 6 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.

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 Figure 7 has three state machines (Turning, Looking and Speaking states).

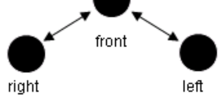


Figure 6. Turning FSM

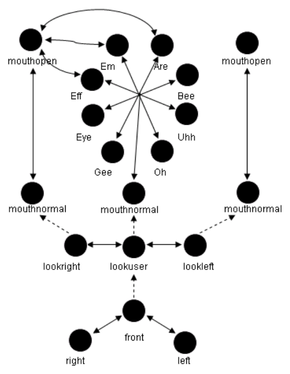


Figure 7. Speech FSM

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 "Look" 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" />
```

4.4. Dynamic Flash Creation

To create Flash, we employ the client-server model that Flash scenario is created at server by the author's script.

Among the server-side Flash creation toolkits available, we use Ming toolkit [15]. 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.

5. System Structure

We have constructed a test bed system which is built based on the MPML script to simulate a virtual presentation scenario.

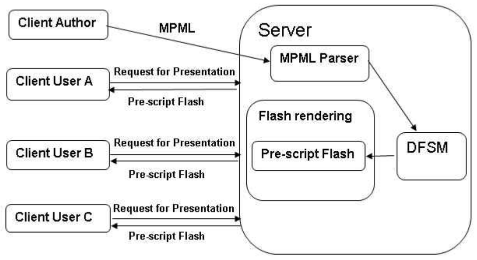


Figure 8. Script based system structure

As shown in Figure 8, 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 pre-script Flash to them.

The detail work flow of the Flash rendering is shown as Figure 9. 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.

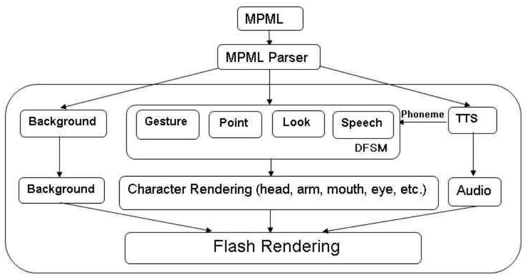
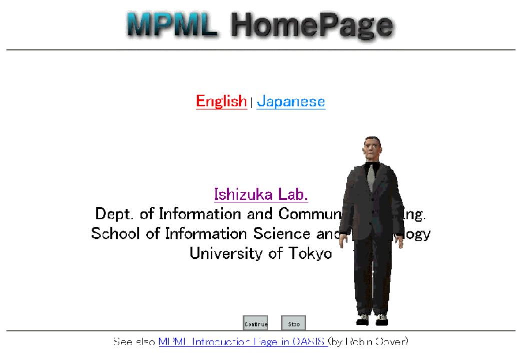


Figure 9. Script based system work flow

6. Demo

We present a live demo (Figure 10) on the web [8] to show our system. In this demo, we present some scenarios to introduce our MPML project.


MPML HomePage

English | Japanese

Ishizuka Lab.
Dept. of Information and Commun[illegible]ng.
School of Information Science an[illegible]ogy
University of Tokyo

See also MPML Introduction Page in OASIS (by Robin Cover)

Figure 10. Demo snapshot

7. Evaluation

Because the existing character systems are few so that it is difficult to make a precise comparison between MPML-FLASH system and others. Here we make an evaluation based on qualitative analysis, not on quantitative analysis.

MPML-FLASH employs Client-Server model that the Flash file is created at the server side. By this way we reduce the burden of the user's side while increasing the burden of the server's side, which can be solved by setting a powerful configuration such as more memory, higher CPU and even multiple CPUs.

When using the MPML3, which mainly employs Microsoft Agent as the character, the user should first download the character file. It needs some time especially under the slow network connection condition. In case of MPML-FLASH, the time consumed process is creating Flash scenario at the server side. The user can see it upon the creating process finishes, even the whole Flash file has not been downloaded.

At present, the Flash character has 22 animations. If users want to get some new actions, the character designers such as us should do this job at the server side so that all users can use it and do not need to download the character file, which is necessary for the Microsoft Agent. We are also considering developing some tool so that users can use it to design their own custom characters.

8. Conclusion and Future Work

In this paper, we have proposed our new MPML-FLASH script language for multimodal presentation with character agent control. Such a scripting approach facilitates the generation of well-structured and coherent presentations. Moreover, we have presented a testing system in which uses MPML to control the flow of the presentation. Through this framework, we can simulate a virtual presentation scenario.

We intend to control Flash templates that are created by Macromedia Flash Tool. By this way, we can reuse many already existing Flash files. Moreover, we are now combining another system developed in our Laboratory, named SCREAM [16], with our framework to let the character has his "emotion". We hope that by these methods we can create a virtual world that models the real one in which we live.

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] Z. Huang, A. Eliëns, and C. Visser, "STEP: a Scripting Language for Embodied Agents", in: Helmut Prendinger and Mitsuru Ishizuka (eds.), *Life-like Characters, Tools, Affective Functions and Applications*, Springer-Verlag, 2003.

[2] W.L. Johnson, and J. Rickel, "Integrating pedagogical capabilities in a virtual environment agent", In *Proceedings of the First International Conference on Autonomous Agents*, pages 30-38, Marina del Rey, CA, February 1997. ACM Press.

[3] B. Hayes Roth, R. van Gent, and D.F. Huber, "Acting in character", In *Creating Personalities for Synthetic Actors*, editors, Robert Trappl and Paolo Petta, eds., Springer-Verlag, 1997.

[4] T. Rist, E.Andre, and S.Baldes, "A Flexible Platform for Building Applications with Life-Like Characters", In *IUI'03*, Miami, Florida, USA, January 12-15, 2003.

[5] Berardina De Carolis, Fiorella De Rosis, Valeria Carofiglio, Catherine Pela-chaud, and Isabella Poggi, "Interactive Information Presentation by an Embodied Animated Agent", *International Workshop on Information Presentation and Natural Multimodal Dialogue*, 2001.

[6] Andrew Marriott, Simon Beard, John Stallo, and Quoc Huynh, "VHML – Directing a Talking Head. In Active Media Technology", In *Proc. The Sixth International Computer Science Conference,* Vol. LNCS2252. Springer. Hong Kong, 2001.

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

[8] MPML-FLASH homepage, Retrieved September 15, 2003 from http://www.miv.t.u-tokyo.ac.jp/MPML/en/MPML_FLASH/index.html

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

[10] 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.

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

[12] 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.

[13] 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.

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

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

[16] 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.