MPML3D: Agent Authoring Language for Virtual Worlds

Sebastian Ullrich[1,2], Helmut Prendinger[1], Mitsuru Ishizuka[3]
[1] National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
[2] Virtual Reality Group, RWTH Aachen University, Seffenter Weg 23, 52074 Aachen, Germany
[3] Graduate School of Information Science and Technology, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan
s.ullrich@ieee.org, helmut@nii.ac.jp, ishizuka@i.u-tokyo.ac.jp

ABSTRACT
This paper describes an authoring language for specifying communicative behavior and interaction of agents in virtual worlds. We focus on the popular three-dimensional (3D) multi-user online world "Second Life" and the emerging "OpenSimulator" project. While tools for designing avatars and in-world objects exist, technology to support content creators in scripting (computer-controlled) agents ("bots") is currently missing. Therefore, we have implemented new client software that controls the verbal and non-verbal behavior of bots based on the Multimodal Presentation Markup Language 3D (MPML3D). This paper compares both platforms and discusses the merits and limitations of each from the perspective of adding agents.

Categories and Subject Descriptors
H.5.1 [**Information Interfaces And Presentation**]: Multimedia Information Systems—*Artificial, augmented, and virtual realities*

General Terms
Human Factors

Keywords
agents, authoring language, metaverse, virtual worlds

1. INTRODUCTION
Metaverses as first envisioned in the scifi-novel "Snow Crash" [10] are manifesting themselves nowadays as online virtual worlds and are becoming increasingly popular [9]. Such environments are mostly used for entertainment, research and business purposes. Second Life (SL) is a prominent example of such a virtual online world[1]. SL provides a free networked multi-user three-dimensional (3D) environment and is very popular with an increasing amount of registered users (over 15 million as of October 2008) and about 50,000 users online at any time. Users of SL can design their own objects, such as buildings, vehicles, or even entire eco-systems. OpenSimulator (OpenSim) is an open source project aiming to create and deploy metaverses[2]. Since the beginning of 2007, it is being developed under the Berkeley Software Distribution (BSD) license. The goal of the originators is to provide an open and extensible platform, which can be run on one's own server(s), rather than on servers of Linden Lab, the company behind SL. Otherwise, the motivation, goals and challenges of OpenSim are quite similar to those of Second Life. It is rather surprising that bots are currently almost completely missing from Second Life and OpenSim. We can only speculate about the reason: possibly, since bots have to be programmed in the C# language, common content creators of virtual worlds might lack the skill to specify the behavior of bots. Therefore, in order to support non-computer science professionals, we have developed an XML-based agent authoring language for virtual worlds, based on the Multimodal Presentation Markup Language (MPML) [7]. MPML3D is a scripting language for interaction-rich scenarios with reactive agents, which was recently adapted to SL [11]. Here, in this paper, the system is further extended to OpenSim, and then compared to SL. Both implementations are described in detail and the platforms are evaluated afterwards.

[1]http://www.secondlife.com

Advances in Computer Entertainment Technology 2008, Yokohama, Japan.

2. RELATED WORK
There are only few scripting languages for behavior planning of life-like characters that are directed for use in recent online virtual worlds. One example is an authoring language that integrates of BML [3] into EVE Online, a massively multiplayer online role-playing game (MMORPG). It will allow players to interact with autonomous agents and to automate coordination of nonverbal social behavior. It is an official addition to the game by the developers themselves. Because of the space setting of the game and the closed source game engine, content creators are not able to create their own custom scenarios in this online world. Nakanishi and Ishida developed Freewalk [5], a platform for social interaction between multiple users and agents. Central aspects of this work are a shared environment, an interaction model, and an interaction scenario. The description language Q [2] is used to describe the interaction scenarios and to define the roles of the agents. Although, the Freewalk system has been used for several applications and multi-user experiments, no persistent online presence can be found. Yet, it is essential to have a solid user base for a virtual world. The work of

[2]http://www.opensimulator.org

Friedman et al. [1] does focus on the evaluation of social behavior in SL with a simple agent. The agent was driven by the Linden Scripting Language and was mainly used for data logging and traversing through SL. A very basic type of navigation has been implemented (walking in random directions until either an obstacle or an avatar is found). Furthermore, the agent can greet avatars by their names and perform gestures. In summary, we have not found an authoring language in the literature that supports content creators to easily script agents (bots) in a widely used multi-user 3D environment. Until now, there seems to be only one, very specialized solution for agents in Second Life [1].

3. SECOND LIFE AND OPENSIMULATOR

3.1 System Setup

Second Life is a client/server application for multiple networked users. The client software is available for multiple platforms. Linden Lab maintains a cluster network to host regions of 3D virtual environments, the "islands". These islands contain user-created 3D content and can be interactively explored by users logged into SL. The 3D content is hosted on the cluster network servers and streamed in an encrypted, protected format to the client application. This encourages users to have their own virtual property, to create new 3D models and to participate in the economic system of SL. OpenSim is a server system for virtual worlds implemented in C# and has several modes of operation. It can be used in stand-alone mode to host environments similar to an intranet. Additionally, a grid-mode allows several OpenSim servers to host a scale-able environment. The system is partially compatible to the Second Life protocol. For that reason, the official client of Second Life can be used to connect to an OpenSim server. Although, OpenSim is still in alpha stage, the system is growing rapidly and even big companies like IBM and 3Di are supporting the development.

3.2 Scripting Languages

The "Linden Scripting Language" (LSL) allows to assign scripts to in-world objects. With over 300 library functions and different data and message types, scripts can control the behavior of virtual objects and communicate with other objects and avatars. Limitations of the scripting language include time delays for movement of objects (0.2 sec) and memory constraints for scripts (16 KB). Furthermore, a script cannot be assigned to avatars. A solution to circumvent this problem is to attach a virtual object (which includes a script) to an avatar. In [1], for example, the avatar wears a ring which contains a script that can apply animations to the avatar. Due to indirect control and slow script execution the result is not convincing. In OpenSim there are several languages for in-world scripting: LSL, OpenSim Scripting Language (OSSL) and C# scripts. While LSL is being re-implemented, the crucial difference to SL is the compilation to .Net code which results in a faster execution time. OSSL is meant as an extension to LSL and C# scripts allow the developer to create new methods.

3.3 Programming Interfaces

Libsecondlife[3] is an inofficial API for SL and can be used for OpenSim as well. It connects to SL as an alternative client and has access to some of the data that is provided to the client. Among many functions for controlling the avatar, it contains methods for communication, navigation, and object manipulation. Thus, compared to the Linden Script Language, its main advantages include (1) full control of the avatar, (2) responsiveness (i.e., no time delay), and (3) no memory constraints. Most importantly, because OpenSim is open source, algorithms can be extended or replaced very easily. A modular architecture ensures extensibility by implementing new plugins. There are different types of plugins: OpenSim Database Plugins, OpenSim Application Plugins and OpenSim Region Modules. As the names suggest these plugins cover different domains within OpenSim and allow for additions on specific layers.

[3]http://www.libsecondlife.org

3.4 Avatars and Resources

The appearance of avatars in SL is defined by the standard male or female body shape. This shape can be changed by parameters to create individual body types. Virtual cloths, skin textures, hair styles and accessories can be either modeled by the user or purchased within the virtual economy of SL. Animation files, which can be applied to the avatars, must be uploaded in the BVH-format. SL supports spatial 3D sound (i.e., sound panning and attenuation) and also allows for streaming audio. Audio sample files can be uploaded by the users and are hosted on the server. Each file is restricted to a maximum length of 10s and must have a sample rate of 44.1Hz. The avatar appearance modeling in OpenSim is similar to SL. But in combination with an alternative Viewer from realXtend [8], OpenSim allows the usage of arbitrary 3D meshes that are rendered by the open source 3D engine OGRE. This leads to a higher flexibility as opposed to SL. Direct joint manipulation enables new algorithms for skeletal and facial animations. For example, this allows to parameterize animations, implement inverse kinematics and to do accurate lip synchronization and blend it with different facial expressions reflecting emotions.

3.5 Comparison

In summary, both SL and OpenSim are promising platforms for virtual worlds. Although they have a lot of comparable features, both have advantages and shortcomings (see Table 1). On the one side, SL has a big community, much content and is relatively stable. On the other side, OpenSim is highly extensible but still in alpha stage and sometimes unstable. Very recently, IBM and Linden Labs announced interoperability between SL and OpenSim by allowing avatars to teleport from one world to the other by proposing a new open grid protocol [4]. Because of these reasons, we support both platforms.

4. MPML3D FOR VIRTUAL WORLDS

4.1 System Architecture

Our system is divided into three different modules: the MPML3D server, the virtual worlds server and an according client. As input, the system requires a MPML3D script, which enables users within the virtual world (avatars) to interact with the script-driven agents (bots). In order to host a MPML3D-based scenario within SL or OpenSim, the content creator has to use the services that are provided by the MPML3D server. Potential users or visitors just need

	Second Life	OpenSim
Registered users	∼15 million	∼50,000
Regions/islands	∼25,000	∼4,000
Economy	linden dollar	none yet
Hosting location	only at Linden Labs	anywhere
Hosting costs	annual fee	free
Server	closed source	open source
Scripting	LSL	LSL, OSSL, C#
API	libsl	libsl, plugins
3D models	prims only	prims, 3D meshes
Animation	BVH only	joint manipulation

Table 1: Comparison between Second Life and OpenSim (as of october 2008).

to use the free, official SL-client software. The actual implementation of the server module is based upon the MPML3D framework [6], which has been re-factored into the MPML3D backend and the reference frontend. Additionally, two frontends have been implemented to provide an interface to the environment of SL and OpenSim respectively. Subsequently, the individual components of the MPML3D server module are described.

4.2 MPML3D Backend

The MPML3D backend implements the scenario (scene setup and scene plot/content) as defined in an MPML3D script, and acts as a host for one or more frontends. It integrates the parser for the XML-based script source with the dynamic runtime representation of all interdependencies between and the hierarchy of agents' conversational activities and perceptions, the so-called "activity network". In order to keep this part of the MPML3D system flexible, the backend handles scene entities and activities in an abstract way, leaving the actual implementation to the frontends.

4.3 SL & OpenSim Frontends

Each frontend realizes a user interface – comprising multimodal output as well as user feedback channels – for interactive content scripted via MPML3D. The SL frontend is responsible for the presentation of the MPML3D content in SL. It is implemented in C# and makes use of the libsecondlife API (as described in Section 3.3) to connect to SL. For OpenSim we have ported the SL Frontend. To show the purpose of the frontends, we explain the initialization phase and presentation phase (shown in Fig. 1). After registering at the MPML3D backend, the scene data is received. Text-based sentences that are specified in the MPML3D script must be synthesized with the help of a TTS-system and are then uploaded to SL. On a further execution of the script this step is obsolete. The backend sends the scene setup for each humanoid entity (bot), and logs the used bots, according to user accounts. I.e., bots also require user accounts, which have to be registered/provided by the content creator beforehand. After the initialization is finished, the SL frontend monitors the virtual environment in SL for incoming events. In the example the two agents are waiting in idle mode in a lecture hall (see Fig. 2). As soon as visitors appear and step on the sensor badge ("NII Open House 2008") once, a perception is triggered which creates a plot activation event in the SL frontend. This event in turn sends a

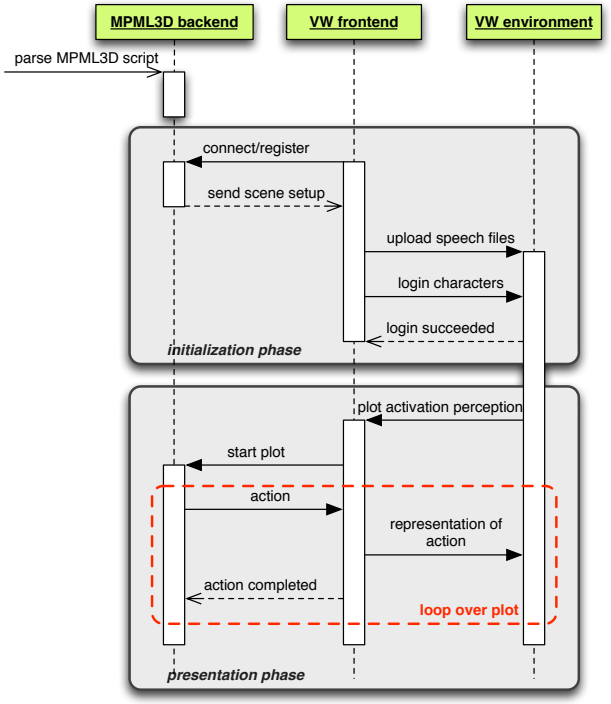

Figure 1: UML sequence diagram to illustrate the runtime behavior of the modules to control agents in virtual worlds with a MPML3D script.

'start plot command' to the backend, which then begins the presentation. During the presentation the bots show verbal and non-verbal behavior by use of timed speech output, gesture playback and a basic lip synchronization which have been implemented as functions in the frontend.

5. COMPARISON AND DISCUSSION

In the following, we will discuss the functionality and limitations of the methods that we have implemented for the representation in SL and how they can be addressed in OpenSim:

Gesture playback. To animate the virtual characters we have implemented basic support to start and stop previously uploaded or officially provided gestures in SL. Due to technical restrictions in SL, it is not possible to change the playback speed or to blend animations. Moreover, one cannot perform low-level animation, because it is impossible to manipulate the joints of SL character directly. In OpenSim, however, it is possible to change the playback speed and to blend animations. Additionally, the OpenSim developer team is currently working on support for low-level animation. This will allow for parameterized gestures and individual pointing gestures.

Speech output. We have identified three different approaches for speech output in SL: (1) playback of previously uploaded audio-clips, (2) streaming audio that is generated on the server-side in nearly real-time, and (3) text (input) that is intercepted on the client-side and synthesized by lo-

Figure 2: Example of an audience of visitors attending to a presentation given by two MPML3D scripted agents in SL.

cally installed TTS software. For the most accurate timing, we favor the first approach. To organize speech-clips in a meaningful way, we have created a speech-cube in SL. This cube is a small transparent object that is attached to the agent and serves as a 'container' for the speech-clips. It has an associated LSL script that manages these speech-clips, and can start, pause, resume, and stop these files by commands from the SL frontend. In OpenSim, speech output can either be realized by uploading audio-clips and sending playback commands (as in SL), or, alternatively, by streaming of audio files. The latter can be achieved more easily than in SL (currently evaluated).

Pseudo-lipsync. At the time of writing this paper, lip synchronization is not supported in SL and no third-party solutions exist. The two programming interfaces of SL provide no access to the internal data-structures, which would enable the implementation of a new facial animation system. As a tentative method, we implemented pseudo lip synchronization, using randomized "mouth open" animations. To achieve a good lip-synchronization, work on a client-side solution is required. OpenViewer[4] is an alternative to the official client of Second Life and could be used for such an approach in future work. Alternatively, the rendering (visual) quality should also improve greatly by the contributions of realXtend [8] which will be integrated as modules to OpenSim.

6. CONCLUSIONS

In this paper we have presented the Multimodal Presentation Markup Language MPML3D as an authoring language to easily control multiple virtual agents in virtual worlds. The system is based on the MPML3D framework, which has been re-designed and extended to accommodate interfaces to virtual worlds like Second Life and OpenSim. While the extension for SL has been attempted before [11], this is the first time to integrate MPML3D with OpenSim. Hence, we have described, discussed and compared the technical features and capabilities of SL and OpenSim. The resulting system has been successfully implemented and tested with existing and new scripts, without compromising key features of MPML3D. In summary, the developed system combines the easy-to-use multimodal content authoring language for life-like characters, MPML3D, with the feature and content rich multi-user online environments of SL and OpenSim.

[4]http://www.openviewer.org

Acknowledgements

The first author was supported by a "Strategic Project" Grant from the National Institute of Informatics. Furthermore, the authors would like to thank Jeff Ames and Adam Johnson, two core developers of OpenSim, for their valuable information and fruitful discussions.

7. REFERENCES

[1] D. Friedman, A. Steed, and M. Slater. Spatial social behavior in second life. In P. et al., editor, *Proceedings Intelligent Virtual Agents LNAI 4722*, pages 252–264, 2007.

[2] T. Ishida. Q: A scenario description language for interactive agents. *IEEE Computer*, 35(11):54–59, 2002.

[3] S. Kopp, B. Krenn, S. Marsella, A. N. Marshall, C. Pelachaud, H. Pirker, K. R. Thórisson, and H. Vilhjálmsson. Towards a common framework for multimodal generation: The behavior markup language. In *Proceedings of 6th International Conference on Intelligent Virtual Agents*, pages 205–217. Springer, 2006.

[4] H. Linden. IBM and Linden Lab Interoperability Announcement. http://blog.secondlife.com/2008/07/08/ibm-linden-lab-interoperability-announcement/, last visited: 2008/07/14.

[5] H. Nakanishi and T. Ishida. FreeWalk/Q: Social Interaction Platform in Virtual Space. In *ACM Symposium on Virtual Reality Software and Technology (VRST-04)*, pages 97–104, 2004.

[6] M. Nischt, H. Prendinger, E. André, and M. Ishizuka. MPML3D: a reactive framework for the Multimodal Presentation Markup Language. In *Proceedings 6th International Conference on Intelligent Virtual Agents (IVA-06)*, Springer LNAI 4133, pages 218–229, 2006.

[7] H. Prendinger, S. Descamps, and M. Ishizuka. MPML: A markup language for controlling the behavior of life-like characters. *Journal of Visual Languages and Computing*, 15(2):183–203, 2004.

[8] realXtend Developer Community. realXtend - Open source platform for interconnected virtual worlds. http://www.realxtend.org, last visited: 2008/07/14.

[9] J. M. Smart, J. Cascio, and J. Paffendorf. Metaverse Roadmap: Pathways to the 3D Web. http://www.metaverseroadmap.org, 2007.

[10] N. Stephenson. *Snow Crash*. Spectra, 1992.

[11] S. Ullrich, K. Brügmann, H. Prendinger, and M. Ishizuka. Extending MPML3D to Second Life. In *Proceedings 8th Int'l Conf on Intelligent Virtual Agents (IVA'08)*, Tokyo, Japan, September 2008. Springer Verlag, in press.