Extending MPML3D to Second Life

Sebastian Ullrich[1,2], Klaus Bruegmann[1],
Helmut Prendinger[1], and Mitsuru Ishizuka[3]

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

Abstract. This paper describes an approach how to integrate virtual agents into the 3D multi-user online world Second Life. For this purpose we have implemented a new client software for Second Life that controls virtual agents ("bots") and makes use of the Multimodal Presentation Markup Language 3D (MPML3D) to define their behavior. The technical merits and limitations of Second Life are discussed and solutions are provided. A multi-user scenario serves as an example to illustrate our solutions to technical challenges and advantages of using the virtual environment of Second Life.

1 Introduction

Virtual online worlds are becoming increasingly popular and can be characterized as metaverses [13] as envisioned in the scifi-novel "Snow Crash" [14]. Second Life (SL) is such a virtual online world that provides a free networked multi-user three-dimensional (3D) environment [7]. SL is very popular with an increasing amount of registered users (over 13 million as of April 2008) and about 50,000 users online at any time. The main features of SL are (1) support of social interactions between the avatars of users (SL residents) with customizable representations, (2) support for user-created content (like a '3D wiki space'), and (3) its economy with a marketplace and own currency, called "Linden dollars".

These features allow SL users to design their own objects, including buildings, vehicles, and even eco-systems, on their privately owned virtual locations ("islands"), and to open the island to the SL community. Many institutions, both commercial and academic, have recently opened their presence in Second Life. However, it was soon noticed that simply building a visually impressive place is not sufficient for an attractive presence in an inherently social space like a virtual world. The key to the success of an island is to provide visitors an interactive experience. Some islands demonstrate a high level of interaction because many avatars meet there. Other islands, however, are rather deserted and give an uncomfortable feeling to the visiting avatar.

One suggestion is to populate deserted islands with ‘bots’, i.e. computer-controlled virtual agents, which may play the roles of guides, receptionists, guards, or other visitors of the island. Note that in the gaming world, bots are rather called NPCs (Non-Player Characters), and bots should not be confused with avatars, which are controlled by users. Quite surprisingly, bots are currently almost missing from Second Life.

Hence, in this paper, we propose an XML-based authoring language for SL bots, which can also be used by non-computer science professionals. In particular, we have extended MPML3D [10] to support SL. MPML3D is a successor of the Multimodal Presentation Markup Language (MPML) [11]. MPML3D is a scripting language for interaction-rich scenarios with reactive agents. The proposed novel extension of the MPML3D framework enables content creators to define their own scripted interactive agents for SL.

2 Related Work

There are many different markup languages and systems for behavior planing of Embodied Conversational Agents (ECAs), e.g., APML – a Mark-up Language for Believable Behavior Generation [2], BML – Behavior Markup Language [6], and MPML – Multimodal Presentation Markup Language [11,10], just to name a few. Here we want to focus on reporting solutions that are used for multi-user environments, and also other approaches to adding agents to SL.

BML is being integrated into EVE-Online, a massively multiplayer online role-playing game (MMORPG) [15]. It is an official addition to the game by the developers themselves, and will allow players to interact with autonomous agents and to automate coordination of nonverbal social behavior. 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.

Freewalk [9] is a platform that allows social interaction between multiple users and agents. Much effort has been put into preparing a shared environment, an interaction model, and an interaction scenario. To describe the interaction scenarios and to define the roles of the agents, the description language Q [5] is used. Several applications and multi-user experiments have been conducted with Freewalk. It is unclear, however, how easily new content can be created with Freewalk/Q.

Friedman et al. [3] created a simple agent to evaluate social behavior in SL. The agent was driven by the Linden Script 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 a solution that offers easily scriptable bots in a widely used multi-user 3D environment. Until now, there are only a few and very specialized solutions for agents in Second Life.

3 Second Life Overview

3.1 Infrastructure

Second Life is a client/server application for multiple networked users. In order to use SL, one needs to register a free or professional account for a virtual character/avatar. The client software can be downloaded for free and is available for multiple platforms (Windows 2000/XP/Vista, Mac OS X and Linux). Linden Lab, the company that developed SL, 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 the users that are logged into the system of SL. Similar to HTML, with a web server and browser application, the 3D content is hosted on the cluster network servers and streamed to the client application. However, the crucial difference is that the content in SL is protected by a digital rights management system.

3.2 Programming Interfaces

There is an official scripting language in SL, which is called "Linden Script Language" (LSL). 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 script language include time delays for movement of objects (0.2 sec) and memory constraints for scripts (16 KB), i.e., complex algorithms must be either simplified or distributed into several scripts (if the algorithm can be subdivided into smaller tasks). Furthermore, a script cannot be assigned to avatars.

Libsecondlife is an inofficial API for SL that is being developed by an open source community [12]. 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 are: full control of the avatar, responsiveness (i.e., no time delay), and no memory constraints.

3.3 Entities and Resources

The content in SL is mainly created by its users and covers the appearance of avatars (that represent the users), virtual objects, and other resources (animations, sounds, pictures, etc). The appearance is mainly defined by the body shape that can be changed by anthropometrical parameters like height, length of limbs, diameters, and many more. 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.To create new animation files or to re-target existing motion from other systems, reference models for male and female avatars exist [8]. These models contain the hierarchical setup of the skeleton, which is comparable to the reference models of the H-Anim standard [4].

4 MPML3D Framework with SL-Extension

Our new system integrates MPML3D into SL and is based on the MPML3D framework [10,1]. The whole system (see Fig. 1) is divided into three different sections: (1) the MPML server, (2) the "official" server (cluster network) from Linden Lab, and (3) the client side of visitors/users of the system. Basically, the system requires a MPML3D script as input in order to allow users within SL (avatars) to interact with the script-driven agents (bots). To actually host a MPML3D-based scenario within SL, the content creator has to use the services that are provided by the MPML3D server. Potential users or visitors just need to use the free, official SL-client software. The actual implementation is based upon the MPML3D framework, which has been re-factored into the MPML3D backend and the JAVA-based reference frontend that uses OpenGL for local output. Additionally, a second frontend has been implemented to create an interface to the networked environment of SL. In the following, we describe the individual components of the MPML3D server.

4.1 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 (see Section 4.2). 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. The MPML3D backend accepts incoming network connections from frontends. After initialization, the backend waits for the beginning of the scene plot, which is triggered by the frontend. Then the backend issues 'start' respectively 'stop' commands for those

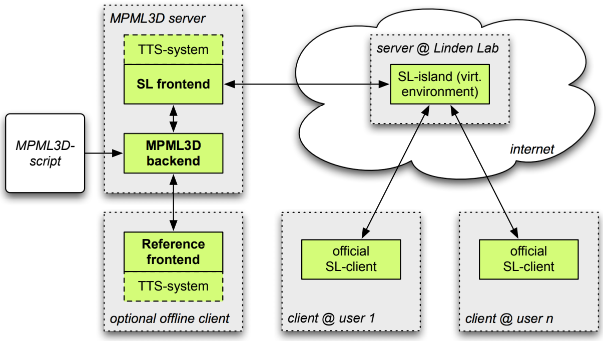

Fig. 1. Overview of the system architecture with the different components distributed over several servers

actions, as defined by the scene plot of the MPML3D script. The frontend in turn performs the output for each action according to its specification and notifies the backend when an action is completed.

In our example (see below), the MPML3D script file for the presentation of a traditional Japanese interior design is parsed, the entities (bots) 'Ken' and 'Yuuki' are loaded, and so is the activity network. The backend waits for connections from the frontends.

4.2 Frontends

Each frontend realizes a user interface – comprising multimodal output as well as user feedback channels – for interactive content scripted via MPML3D. A frontend provides a concrete implementation for supported entity types along with their actions and entity states.

The following excerpts from the example MPML3D-script demonstrates two very important features of MPML3D that are supported by each frontend: (1) gestures synchronized to utterances, and (2) perceptions to create interruptions. To synchronize gestures to utterances, the timing of gestures is specified in square brackets and refers to the count of words of the sentence that is defined in the speak action. This means that `kenSpeak[1].begin` will be performed at the beginning of the first word "Me", and `kenSpeak[4].begin` at the beginning of "and", respectively.

```
<Parallel>
 <Action name="act">
   ken.speak("Me, Ken Kobayashi, and my colleague...")</Action>
 <Action startOn="act[1].begin">ken.gesture("MYSELF")</Action>
 <Action startOn="act[4].begin">ken.gesture("POINT_LEFT")</Action>
</Parallel>
```

Here is an example for the perception feature. The action to be interrupted is the following sentence by the female bot Yuuki.

```
<Action name="yuukiSpeak">
  yuuki.speak("What you can see here is a washitsu.")</Action>
```

A perception is set up with a unique identifier (here, "interruption") and the activating event is specified.

```
<Perception name="interruption">
  onEvent(yuuki, "saysPhrase", "you can see")</Perception>
```

The following task is immediately started when the perception is triggered, which causes the termination of the previous action.

```
<Task name="explain" startOn="interruption">
<Parallel>
  <Action name="kenSpeak">
    ken.speak("Yuki! Wait! We should first explain...")</Action>
```

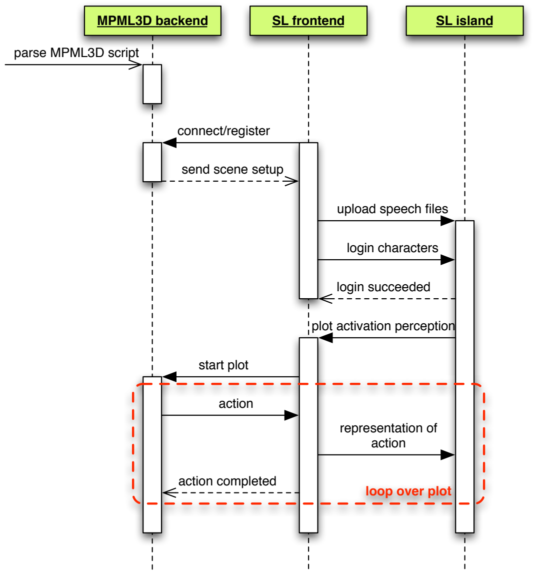

Fig. 2. left: UML sequence diagram to illustrate the initialization and run-time behavior of the modules to control agents in SL with a MPML3D script. **right:** Example of a visitor listening to the presentation of the agents in SL.

4.3 SL Frontend

The SL frontend is responsible for the presentation of MPML3D content in SL. It is implemented in C# and makes use of the libsecondlife API (as described in section 3.2). During the initialization process (shown in Figure 2, left) the frontend is registered at the MPML3D backend, the scene data is received and the agents are logged into SL. In the example the two agents are giving a presentation in front of a japanese building (see Figure 2, right).

In the following, we will describe the functionality and limitations of the methods that we have implemented for the representation in SL:

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.

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 output that is intercepted on the client-side and synthesized by locally installed TTS software. For the most accurate timing under the given circumstances, we favor the first approach.

Pseudo-lipsync. At the time of writing this paper, lip synchronization is not supported in SL and no third-party solutions exists. 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.

4.4 Challenges in SL

Timing issues in SL. Due to the distributed server/client architecture of SL, it is difficult to achieve accurate timing. Because of the many ways speed on the internet can be affected, synchronization between server and client should be avoided. The most feasible solution is to synchronize the events on the server-side.

Multiple users. Another interesting challenge is to handle the unpredictability of visitors (users in the form of avatars) in SL. A presenter bot has to consider (1) when to start a presentation, (2) when to abort a presentation, and (3) where to look in case of multiple visitor avatars. We provide a few heuristics to cope with these problems. The agents continuously scan their environment and if a visitor (avatar) enters within a predefined radius, an event is triggered.

5 Conclusions

In this paper we have presented a novel solution, that enables life-like agents in Second Life with behavior controlled by the Multimodal Presentation Markup Language MPML3D. We have evaluated the technical details of SL and provide solutions to ensure responsive behavior and smooth interactions of the agents. The resulting system has been successfully implemented and tested with existing and new MPML3D scripts, without compromising key features of MPML3D. In summary, the system combines the easy-to-use authoring multimodal content language for life-like characters MPML3D with the feature and content rich multi-user online environment of SL. There are numerous applications and scenarios where the agents can be used. Possible areas of usage are in education, research or entertainment, e.g. to give dialogues, instructions or speeches, to be part of user studies, or for testing within simulation environments.

Future work will focus on the addition of new features that are possible due to SL, like basic movement of the agents to navigate to specific target location, to walk towards or to follow other virtual character (agents or avatars), and to react on additional levels to user input (e.g., talking, transaction of objects, etc).

References

1. Brügmann, K., Dohrn, H., Prendinger, H., Stamminger, M., Ishizuka, M.: Phase-based gesture motion parametrization and transitions for conversational agents with mpml3d. In: INTETAIN 2008: Proceedings of the 2nd international conference on Intelligent Technologies for interactive Entertainment. ACM Digitial Library (to appear, 2008)

2. Carolis, B.D., Pelauchaud, C., Poggi, I., Steedman, M.: Life-Like Characters. Tools, Affective Functions, and Applications. Cognitive Technologies, chapter APML: Mark-up language for communicative character expressions. Springer, Heidelberg (2004)
3. Friedman, D., Steed, A., Slater, M.: Spatial social behavior in second life. In: Pelachaud, C., Martin, J.-C., André, E., Chollet, G., Karpouzis, K., Pelé, D. (eds.) IVA 2007. LNCS (LNAI), vol. 4722, pp. 252–264. Springer, Heidelberg (2007)
4. H-Anim Working Group. Information technology — computer graphics and image processing — humanoid animation (h-anim) (last visited: 2008/04/11), `http://www.h-anim.org`
5. Toru Ishida, Q.: A scenario description language for interactive agents. IEEE Computer 35(11), 54–59 (2002)
6. Kopp, S., Krenn, B., Marsella, S., Marshall, A.N., Pelachaud, C., Pirker, H., Thórisson, K.R., Kopp, H.V.S., Krenn, B., Marsella, S., Marshall, A.N., Pelachaud, C., Pirker, H., Thórisson, K.R., Vilhjálmsson, H.: Towards a common framework for multimodal generation: The behavior markup language. In: Proceedings of 6th International Conference on Intelligent Virtual Agents, pp. 205–217. Springer, Heidelberg (2006)
7. Linden Lab. Offical website of second life (last visited: 2008/04/11), `http://www.secondlife.com`
8. Linden Lab. Reference character models for second life (last visited: 2008/04/11), `https://secondlife.com/downloads/avatar.php`
9. Nakanishi, H., Ishida, T.: Freewalk/q: Social interaction platform in virtual space. In: ACM Symposium on Virtual Reality Software and Technology (VRST 2004), pp. 97–104 (2004)
10. Nischt, M., Prendinger, H., André, E., Ishizuka, M.: MPML3D: a reactive framework for the Multimodal Presentation Markup Language. In: Gratch, J., Young, M., Aylett, R.S., Ballin, D., Olivier, P. (eds.) IVA 2006. LNCS (LNAI), vol. 4133, pp. 218–229. Springer, Heidelberg (2006)
11. Prendinger, H., Descamps, S., Ishizuka, M.: MPML: A markup language for controlling the behavior of life-like characters. Journal of Visual Languages and Computing 15(2), 183–203 (2004)
12. Second Life Reverse Engineering Team. libsecondlife API (last visited: 2008/04/11), `http://www.libsecondlife.org`
13. Smart, J.M., Cascio, J., Paffendorf, J.: Metaverse roadmap: Pathways to the 3rd web (2007), `http://www.metaverseroadmap.org`
14. Stephenson, N.: Snow Crash. Spectra (1992)
15. Vilhjálmsson, H., Cantelmo, N., Cassell, J., Chafai, N.E., Kipp, M., Kopp, S., Mancini, M., Marsella, S., Marshall, A.N., Pelachaud, C., Ruttkay, Z., Thórisson, K.R., van Welbergen, H., van der Werf, R.J.: The behavior markup language: Recent developments and challenges. In: Pélachaud, C., Martin, J.-C., André, E., Chollet, G., Karpouzis, K., Pelé, D. (eds.) IVA 2007. LNCS (LNAI), vol. 4722, pp. 99–111. Springer, Heidelberg (2007)