A Life-Like Agent Interface System with Second Life Avatars on the OpenSimulator Server

Hiroshi Dohi[1] and Mitsuru Ishizuka[2]

[1] Dept. Information and Communication Engineering, Graduate School of Information Science and Technology, University of Tokyo
[2] Dept. Creative Informatics, Graduate School of Information Science and Technology, University of Tokyo
7-3-1, Hongo, Bunkyo-ku, Tokyo 113-8656, Japan
dohi@mi.ci.i.u-tokyo.ac.jp

Abstract. This paper describes a design of a life-like agent interface system with Second Life avatars on a 3D virtual world. We have implemented our prototype system on the OpenSimulator server, instead of the Linden Lab's Second Life server. It is open source and a Second Life official viewer can connect it. Although it is still an alpha version and has various problems at present, it has many advantages. Our avatar can be controlled by event-driven. And the script is environment-independent since the other avatars might be changing the world. We have built up our portable experimental environment (our avatar controller, the OpenSimulator server, and the Second Life viewer) on an ordinary laptop PC (Windows Vista). It can run even if it is standalone, without an Internet connection.

1 Introduction

An online 3D virtual world has grown explosively. For example, the Linden Lab's "Second Life" has more than 15 million residents (registered users). [10] And many companies from different industries have entered the Second Life world aiming at major business opportunities. In spite of these situations, their evaluation to the Second Life is modest. We can easily find many "desolate" towns in the Second Life world. There are glittering shops; however no people often exist there. Internet services have allowed removing barriers of time and location. Unmanned Web servers can offer much information for 24 hours / 7 days. On the other hand, a 3D virtual world simulates our real life, and a user operates each avatar. That is, it may cause the simple problem again; each area has own "activity" time and hot spots. It may be helpful to create an autonomous avatar working every time in the 3D World.

Until now many life-like agent interface systems have been developed in various research labs. We developed a life-like agent interface system with a realistic face and speech dialog function, called Visual Software Agent – II (VSA-II) [2], and learned much about agent (avatar) control techniques through this research. It worked well in our Lab, but it was difficult to become popular. It used a special agent model and required some complicated setup. And it didn't have any easy tools to create various contents and environments. In other words, it was missing a common platform.

A.A. Ozok and P. Zaphiris (Eds.): Online Communities, LNCS 5621, pp. 182–190, 2009.

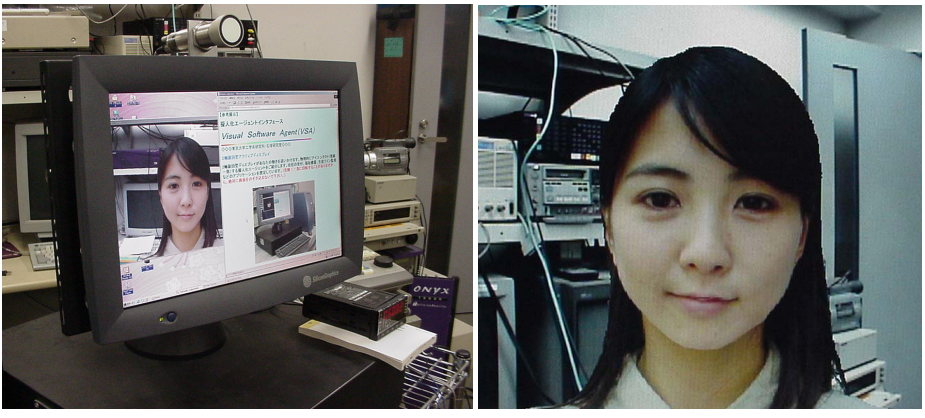

Fig. 1. Visual Software Agent-II Interface System (VSA-II). The girl is a 3D CG virtual agent (avatar) with the realistic face generated from one photograph. The picture is texture-mapped onto a 3D deformable wire-frame model. And then the agent is composed with a live background image (right). A user can talk with her like a videophone.

Our research goal is to realize autonomous agents on 3D virtual world. We think that the Second Life platform and our life-like agent techniques can complement each other to build a useful new system. In the Second Life world, we can create and customize anything with available easy-to-use building tools. And the residents in Second Life can install and use the client viewer on Windows and Linux environment.

Ullrich et al. have developed new client software for Second Life that controls virtual agents. They make use of the Multimodal Presentation Markup Language 3D (MPML3D) to define the behavior of the agents [11].

Daden Ltd. has developed Second Life based avatar chatbots, “Abi Carver” – a virtual receptionist and “Halo Rossini” – an autonomous research bot. These are commercial products. Daden Ltd. has long experience of chatbots and applies it within the Second Life [1].

We have developed a prototype of a life-like agent interface system with Second Life avatars. The Second Life official server is a commercial product. It is hosted at the Linden lab, and many users (Second Life residents) access it simultaneously through an Internet connection. We adopt the OpenSimulator server [8] instead of the Linden Lab’s server. We can connect it as well using the Second Life official viewer. Although the server is still an alpha version and has various problems at present, we think it has many advantages for our research purpose. Ullrich has also attempted to integrate MPML3D with the OpenSimulator [12]. Our prototype system is clearly another implementation.

2 OpenSimulator Server and OpenMetaverse Library

The OpenSimulator [8] is an open source 3D application server. It can be used to create a 3D virtual world, and we can access it with the Linden Lab’s official client viewer. The OpenSimulator is not just another implementation of the Second Life server. It includes much experimental extension.

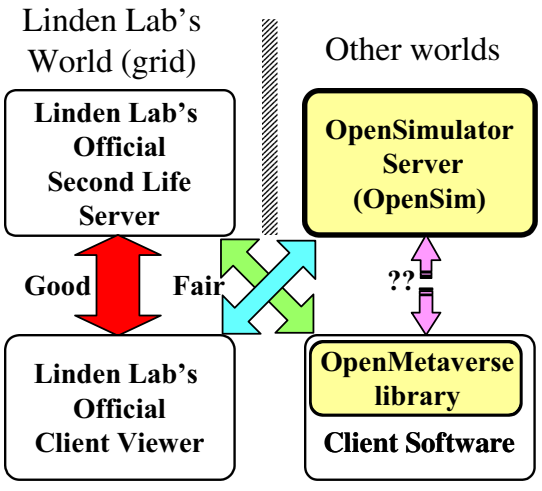

Fig. 2. OpenSimulator server and OpenMetaverse library

The OpenMetaverse library [7] (its former name is libsecondlife [5]) is another open source project in order to study how Second Life works. It is also used to develop original client software for the Second Life official server.

Since both the OpenSimulator and the OpenMetaverse library are still on alpha version at this time and include many problems, it is not necessarily recommended to combine these softwares. We think, however, this combination may be very attractive for us.

They have many advantages, especially for research purposes, such as shown below.

- Open sources.
  Hence we can access and investigate internal codes.
- It can run in standalone mode on a local PC.
  It does not necessarily require high-speed network connection. We can easily build private environment.
- Free of charge.

We can use a free land about 16 acres for any activities on the OpenSimulator server. It also requires no service charges to upload some materials.

On the other hand, we sometimes encounter various problems when using the OpenSimulator. Therefore we have sometimes to apply our own patches in order to avoid the problems / bugs.

3 Life-Like Agent Interface System on 3D Virtual World

3.1 System Configuration

Fig. 3 shows our prototype system configuration. It consists of an avatar controller module, the OpenSimulator server, and at least one Second Life viewer.

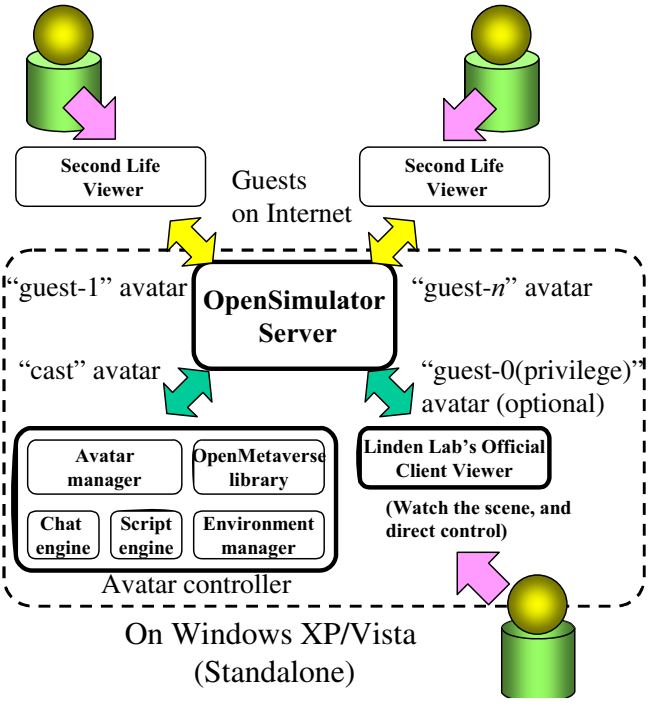

Fig. 3. System configuration

In this paper, we call the computer-controlled avatar the "cast avatar", and other avatars that controlled directly by humans the "guest avatar".

Avatar Controller. The avatar controller controls the cast avatar automatically instead of a human. It is one of Second Life client software, and can log in to the server like the Second Life viewer. The server cannot distinguish the avatar controller with a human.

The avatar controller consists of following functional blocks.

1. OpenMetaverse library
   The avatar controller can access the OpenSimulator server through the OpenMetaverse library.
2. Avatar manager
   The avatar manager controls the avatar in cooperation with both the chat engine and the script engine.
3. Environment manager
   The environment manager accesses the server periodically and gets avatars information in the world. It also gets the inventory list and resolves the object name and UUID.
4. Chat engine
   The chat engine receives a text from other avatars through a chat channel, and replies to it. It also invokes scripts for a presentation.

5. Script engine
   The script engine manages various predefined scripts.

Second Life Viewer(s). Our avatar controller module doesn't have an own viewer. In order to watch the cast avatar in the scene, we use a Second Life official viewer. The viewer joins the Second Life world as a guest avatar with another login name. That is, we need at least two login accounts. Then we can watch the cast avatar and the scene from the back of the guest avatar.

The viewer is optional. The cast avatar appears on other client viewers when the avatar controller joins (logged-in) the Second Life world even if it doesn't have own viewer. Thus, the avatar controller can work on an old PC with modest graphics performance.

Privilege Avatar. A privilege avatar is one of guest avatars, and uses a special login name. This avatar can also control the chat avatar and objects directly through a chat channel. It is mainly used for debugging purpose. Other guest avatars ask some actions to the cast avatar through the chat channel, but their requests are not necessarily accepted.

3.2 Implementation

The OpenSimulator server supports multiple platforms, i.e. Windows, Linux, and Mac OS. We have implemented and tested our prototype system on Windows XP/Vista.

Second Life has excellent 3D graphics. Although it is said that the Linden lab's official viewer requires both high-performance graphics card and high-speed networks, a recent mid-performance PC may be able to run it since Windows Vista has advanced graphics interface.

We have built up one of our experimental environments (the avatar controller, the OpenSimulator server, and the official viewer) on an ordinary laptop PC. Its specification is, Windows Vista, Core2 Duo CPU 1.6GHz, GM965 express chipset, and 2GB memory. It uses an integrated graphics controller on the GM965 chipset and has no high-performance GPU.

It can work even if it doesn't have Internet connection.

Fig. 4 shows a screen snapshot of our experimental environment on Windows XP/Vista.

3.3 Environment-Independent Script

It is important that scripts for the avatar control are environment-independent and reusable.

These are simple examples of our "greeter" scripts. (Low-level scripts.)

```
Turnto    $guest          ; turn to the guest avatar.
Moveto    $guest          ; step forward to the guest avatar.
Anim      HELLO           ; HELLO action.
Say       "Hi, $guest"    ; say "Hi, (guest name)"
                          ;      through the chat channel.
Display   img1,screen1    ; display "img1" on "screen1".
```

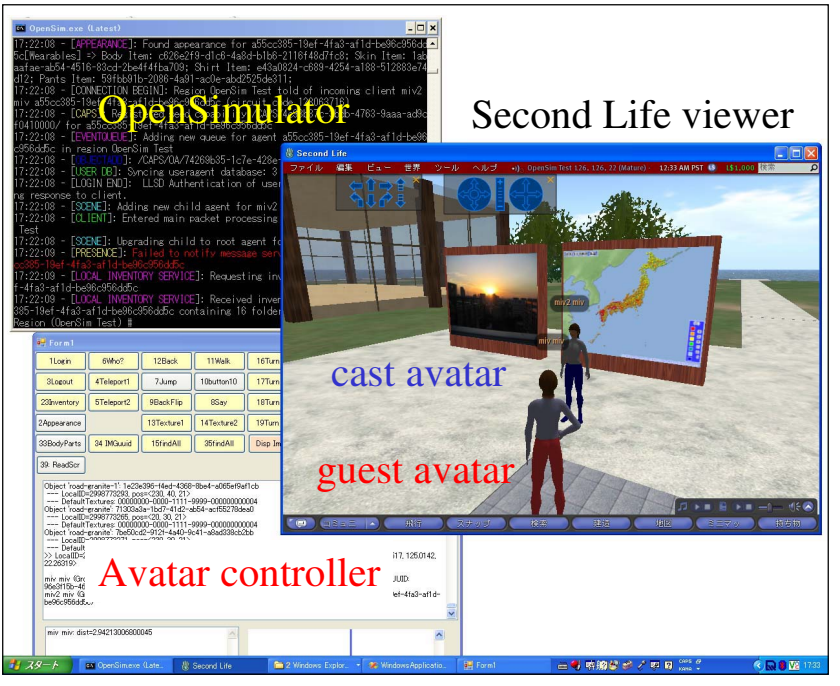

Fig. 4. A screen snapshot of our experimental environment on Windows XP/Vista. There are three windows, the avatar controller (bottom-left), OpenSimulator (top-left), and the Second Life viewer (right). In the Second Life viewer, there are two large displays behind the cast avatar, and their images are changed along the presentation.

The argument "$guest" is a variable. The environment manager binds guest avatar information to "$guest" at the execution time. The cast avatar cannot know who comes in advance. The environment changes as time goes on. This binding may change dynamically.

The "$guest" has information of one avatar at most, however it has multiple meanings. In case of "Turnto / Moveto" instructions, "$guest" means the location of the guest avatar. It is replaced with the guest avatar name in "Say" instruction.

The "Display" instruction displays an image [img1] on an object [screen1]. It is actually a texture mapping to the object. The words "img1" and "screen1" are just names of a texture image and a primitive object, respectively. Because of the restriction of the Second Life server, the image should be uploaded beforehand.

The "screen1" is mapped to the object on the 3D world at the execution time of the script. The script doesn't know the shape of the "screen1". When the object is box-type, it looks like a projector screen or a large display. If it is cylinder-type, it may look like an advertisement pillar.

4 Discussion

4.1 Life-Like Agent Interface System on the Second Life World

Lasting World. The life-like agent interface system on the Second Life world is different from conventional one in several respects. One of the remarkable differences is that the Second Life world is lasting beyond the server down.

A conventional life-like agent interface system, in general, starts with the new (empty) state every time. The first task is to set up the environment using an initialize script, e.g. avatar's start position, avatar's appearance, background, lighting, small objects around avatars, etc. It can replicate avatar interaction since it "resets" the environment at every startup time.

In the Second Life world, once an object is created and put on the world, it remains there until someone deletes it explicitly. The OpenSimulator server has a link with a database engine, and restores the previous environment data from the database at the startup time. It can take over the environment even if the server restarts.

In addition, someone (avatars) might be changing the world, e.g. put on a new object or take away an old one, while your avatar is logged off. It may cause some problems for avatar interaction that the avatar cannot assume the environment exactly. Therefore, the script should be environment-independent.

All objects, e.g. prim (primitive object), avatar, texture, etc, have own Universally Unique Identifier (UUID) in the Second Life world. Many of the OpenMetaverse libraries APIs use UUID directly in order to identify objects. This is, however, inconvenient and frustrating to write a script because UUID consists of a 16-byte number (32 hexadecimal digits).

In our script, we use a name instead of UUID. Both the OpenSimulator server and the OpenMetaverse library use UUID. If the script includes UUID, it cannot be reusable. The copied object has another UUID different from original one.

The environment manager has an inventory list on the world, and resolves the name and UUID. The name is not necessarily unique in the 3D world since the owner of the object can give arbitrarily one. If some objects have the same name on the inventory list, the system chooses one object randomly. If it cannot find the named object, the script will be ignored.

Avatar Information. In the Second Life world, we can get various types of avatar information from the server, e.g. avatar's location and direction, etc. It seems slightly tricky; however it is useful information to establish natural communication. While it is easy for a human to watch the avatar's behavior, the avatar cannot get any information from the scene image.

In a conventional interface system, it assumes that a user takes a seat in front of a computer display. The avatar waits a start cue from the user for the interaction, e.g. clicking a mouse, or typing a keyboard. It usually has no way to know that the user gets up and walks away. Once the avatar starts a presentation, it will carry out until the end even if nobody watches it.

In our system, the cast avatar is controlled by event-driven. All avatars can walk around freely in the world. Its communication starts in three ways as follows.

1. The guest avatar talks to the cast avatar.
2. The cast avatar searches the guest avatar explicitly.
3. The guest avatar is approaching to the cast avatar.

In case 1, the guest avatar initiates the communication. This is an ordinary way. When the guest avatar talks to the cast avatar through a chat channel, it raises a "chat" event. And it invokes the chat-event-handler with both the guest name and text strings. The cast avatar turns to the guest avatar, and then replies.

In case 2, in contrast, the cast avatar initiates the communication. First, the cast avatar searches the guest avatar explicitly. If it finds the guest avatar, the cast avatar talks to the guest avatar. If the guest avatar is far away, the cast avatar may step forward to before talking to.

If both avatars don't initiate the communication and the guest avatar is approaching to the cast avatar within a predefined distance, the event manager raises an "avatar location" event once. This is case 3, in which the event invokes the avatar-location-event-handler with the location, the distance, and the direction of the guest avatar.

It is important to know the direction of the guest avatar. When the guest avatar steps forward and within the distance, the cast avatar says "hi". But the cast avatar says "excuse me," if the guest avatar steps back.

The direction also shows avatar's interest. If the cast avatar talks to but the guest avatar looks another direction, the guest avatar has another interest.

The "avatar location" event is also raised when the guest avatar walks away. Then the cast avatar will abort the presentation and say "good-bye".

4.2 OpenSimulator vs. Second Life Official Server

The OpenSimulator is suitable especially for research purpose because it is open source and we can get free land. Since it is the private server, we can have any experiences without bad influence to others.

In the Second Life world, having own land is essential for any activities in practice. In the official server, it costs to get and maintain own land, although a free membership account is available. We cannot build any building on a public space.

In addition, the official server restricts the number of objects and the size of scripts. They depend on the area size of the owned land. If we built a complex building, it may require a broad land.

On the other hand, we have to manage the OpenSimulator server. The number of the OpenSimulator source codes files written in C# is more than 1,000, and the number of the OpenMetaverse library is about 500. Some functions are not implemented yet and some bugs remain. The development version is updated every day. Some problems are fixed and new experimental functions are added daily, and sometimes they bring new bugs. It requires a certain level of skills to fix bugs.

5 Conclusion

In this paper, we have described our life-like agent interface system with Second Life avatars on the OpenSimulator server. Although now the OpenSimulator currently has

some problems, it has many advantages compared with the Second Life official server, especially for research purpose. A prototype system is working on a laptop PC with Windows Vista. We hope it will be contribute as a test bed for developing an autonomous agent on the virtual 3D space.

References

1. Daden limited, http://www.daden.co.uk
2. Dohi, H., Ishizuka, M.: Life-like Agent Interface on a User-tracking Active Display. In: Smith, M.J., Salvendy, G., Harris, D., Koubek, R.J. (eds.) Usability Evaluation and Interface Design: Cognitive Engineering, Intelligent Agents and Virtual Reality, vol. 1, pp. 534–538 (2001)
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. 252–263. Springer, Heidelberg (2007)
4. Kamel Boulos, M.N., Hetherington, L.: Wheeler. S.: Second Life: an overview of the potential of 3-D virtual worlds in medical and health education. Health Information and Libraries Journal 24(4), 233–245 (2007)
5. libsecondlife www page, http://www.libsecondlife.org
6. Nagaoka, T., Watanabe, S., Sakurai, K., Kunieda, E., Watanabe, S., Taki, M., Yamanaka, Y.: Development of Realistic High-Resolution Whole-Body Voxel Models of Japanese Adult Male and Female of Average Height and Weight, and Application of Models to Radio-Frequency Electromagnetic-Field Dosimetry. Physics in Medicine and Biology 49, 1–15 (2004)
7. OpenMetaverse Foundation, http://www.openmetaverse.org
8. OpenSimulator www page, http://opensimulator.org
9. Quax, P., Monsieurs, P., Jehaes, T., Lamotte, W.: Using Autonomous Avatars to Simulate a Large-Scale Multi-User Networked Virtual Environment. In: International Conference on Virtual-Reality Continuum and its Application in Industry (VRCAI 2004), pp. 88–94 (2004)
10. Second Life official site, http://secondlife.com
11. Ullrich, S., Bruegmann, K., Prendinger, H., Ishizuka, M.: Extending MPML3D to Second Life. In: Prendinger, H., Lester, J.C., Ishizuka, M. (eds.) IVA 2008. LNCS (LNAI), vol. 5208, pp. 281–288. Springer, Heidelberg (2008)
12. Ullrich, S., Prendinger, H., Ishizuka, M.: MPML3D: Agent Authoring Language for Virtual Worlds. In: International Conference on Advances in Computer Entertainment Technology (ACE 2008), pp. 134–137. ACM Press, New York (2008)