Proceedings of the IIEEJ Image Electronics
and Visual Computing Workshop 2010
Nice, France, March 5-7, 2010

AN INTERACTIVE PRESENTATION SYSTEM IN A 3D VIRTUAL WORLD USING AN OPENSIMULATOR SERVER

Hiroshi Dohi [1] *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
dohi@mi.ci.i.u-tokyo.ac.jp

ABSTRACT

We have developed a prototype of an interactive presentation system with avatars in a 3D virtual world. A 3D avatar can have a presentation to other avatars by simple chat and a variety of actions using various images / 3D objects. In a 3D virtual world, each user has own 3D avatar figure, and can watch the scene from arbitrary own viewpoint. The avatar controller joins (login) to the server as one of client software, and controls the avatar instead of a human. And the avatar can change its behavior depending on the user's avatar behavior and body direction. We make use of an open source 3D application server, the OpenSimulator (OpenSim). It is effective to built up the 3D avatar presentation environment, although we sometimes encounter various problems / bugs at present.

1. INTRODUCTION

A 3D virtual world is rapidly becoming popular. Especially, Linden Lab's Second Life [11] has gathered up more than 15 million registered users all over the world. It simulates our real life in the 3D virtual world. Some researches become to make use of Second Life [3][5].

In the 3D virtual world, each user operates own avatar manually. Hence, it spontaneously arises "business hour" and "hot spots" in each place. As a result, it causes many desolate towns, e.g., shops with no clerk and empty streets. WWW servers can always accept any requests, and return much information automatically. Thus it will become important to create an autonomous avatar that acts as a shop assistant and a secretary for 24 hours [2][12].

We have developed a prototype of an interactive presentation system with 3D avatars. As a 3D environment platform, we make use of an open source 3D application server, OpenSimulator [10], and its access library, OpenMetaverse library [9].

2. OPENSIMULATOR

2.1. A 3D avatar interface system

In avatar interface systems, an avatar is classified broadly into two categories. One is a 2D avatar, and another is a 3D avatar.

The 2D avatar is an animated character. It can express anything if it is possible to draw. Microsoft developed an excellent 2D avatar interface system, called the Microsoft Agent (MS Agent) [7], and distributed it with speech libraries without charge. It had been also included in the old version of the Microsoft Office software. It is high quality and easy-to-use software, and makes a strong impact to other avatar interface systems. (Microsoft has decided to discontinue development of Microsoft Agent technologies.)

The 3D avatar is a 3D CG character. It uses, in general, a deformable 3D wire-frame model and applies texture mapping or rendering techniques to it. In previous times, some 3D avatar interface systems had been developed, and only a few systems survive [4]. One reason is that it is difficult to make attractive 3D avatars and fascinating 3D environments. Most 2D avatar interface systems are built on the MS Agent technologies. The 3D avatar interface systems missed common high-level platform technologies, like the MS Agent for 2D avatars.

2.2. OpenSimulator server vs. Second Life server

Second Life gives us the fascinating 3D virtual world with high quality graphics, and it has already gathered huge number of users. Linden Lab has distributed the free official viewer, and all users can enter the 3D world at no fee.

OpenSimulator is also a platform for operating a 3D virtual world. It can be used to create the virtual environment that can be accessed through a variety of clients, on multiple protocols. The OpenSimulator is not just another implementation of the Second Life server,

although it can be used to simulate a virtual environment similar to Second Life.

Table 1: Second Life vs. OpenSim

	Second Life	OpenSimulator
System type:		
	Commercial	***Open source***
Source code	***No***	***Open***
Extendability	***No***	***Yes***
Expenses:		
Land	***Paid***	***Free***
Upload data	***Paid***	***Free***
Server:		
Standalone mode	***No***	***Yes***
Grid mode	***Yes***	***Yes***
Second Life protocol	***Yes***	***Yes***
System stability:		
	Good	***Poor***

Table 1 shows a comparison of Second Life and the OpenSimulator.

(1) System type
Second Life is basically a commercial system. Linden Lab has contributed a great deal to open source community, and has opened the source codes of an official viewer. Ones of the servers are not open because they include accounting and authentication systems.

The OpenSimulator is open source software, and we can easily get huge source codes and many documentation. These are very helpful for us to analyze complex protocols and develop our own avatar control system. The server is designed to be easily extendable through loadable modules to build completely custom configurations.

(2) Expenses
It needs money (Linden dollars) to do something in the Second Life world like in our real life, although we can enter into the Second Life world at no fee.

It is essential to get (buy or rent) own land for any activities in practice, e.g., events, shops, and my home etc., since it is not allowed to put any private objects in public spaces nor change in public terrain. And in order to keep graphics performance, the size of the own land restricts the number of total objects put on the land. The maximum size of each object is a 10m cube. Hence, when we build a tall building, it requires large land spaces.

It also requires service charge to upload various materials like texture image, sound, and animation data into the virtual world from our real world.

In the OpenSimulator server, we can use a large land about 16 acres (256 meters square) / server for any activities without charge. And it doesn't require any service charge.

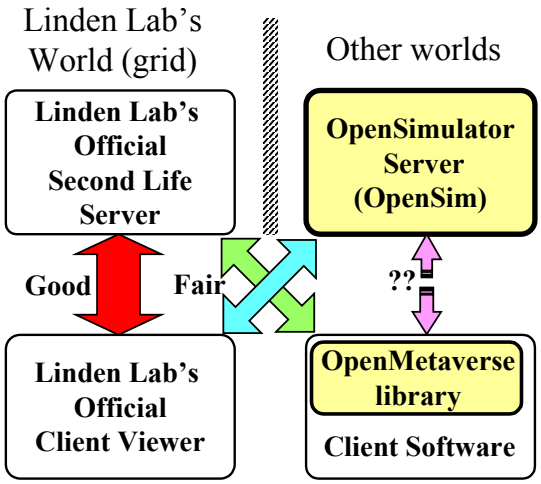

Figure 1: Virtual world servers and clients

(3) Server
Linden Lab hosts all their Second Life servers. We need high-speed network connection to enjoy comfortable virtual life, since many packets are exchanged between the server and the local client through network.

On the other hand, the OpenSimulator server runs on a local PC. Hence we don't need to upload important data with security risk to the outer server. We can keep all private data in our laboratory. The server supports both "standalone" mode and "grid" mode.

In standalone mode, the server runs on a single local PC. It can accept multiple client connections through network. If both the server and the client viewer run on the same PC, it doesn't necessarily network connection.

In grid mode, our server on the local PC can join a virtual world grid through network. Second Life (Linden world) forms another grid because it is a commercial system.

Figure 1 shows relationship between 3D virtual world servers and clients.

Amazingly, the OpenSimulator server has client compatibility with Second Life protocols. It means that we can access the server with a Second Life official viewer. However, the OpenSimulator developer team doesn't promise to implement and support all functions and protocols of the Second Life in the future.

The OpenMetaverse library (its previous name is libsecondlife) [9] is an open source access library for 3D virtual world. It is used for creating clients and automatons in Second Life and the OpenSimulator server.

(4) System stability

At present, unfortunately, the OpenSimulator and the OpenMetaverse library are not stable to satisfactory extent yet. Whereas some functions of the Second Life are not implemented yet, it adds much experimental extension modules that Second Life doesn't have. Many programmers modify and update the huge source codes every day, then many bugs are fixed and new bugs are introduced. Therefore, it may require specific programming skills to make really good use of the software.

As a result, we chose the OpenSimulator server and the OpenMetaverse library. This combination may be very attractive for our research. The source codes are important and helpful for us. We have sometimes to apply our own patches in order to avoid the problems / bugs.

3. SYSTEM CONFIGURATION

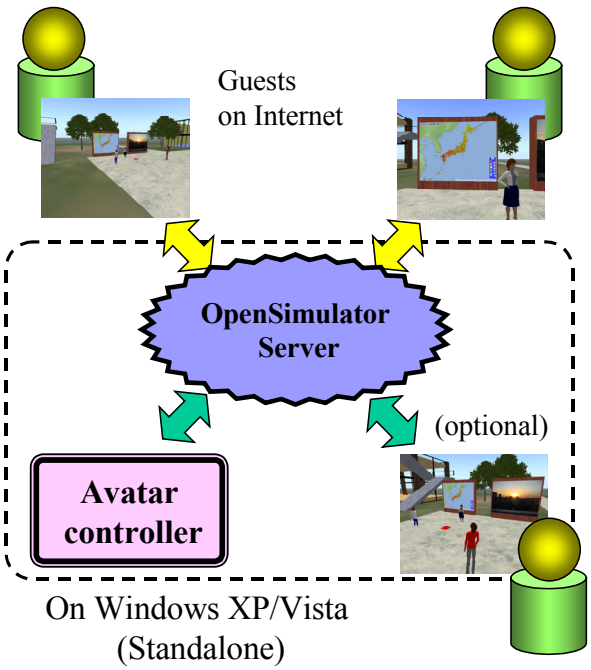

Figure 2: System configuration

3.1. Overview

Figure 2 shows our system configuration.
We have built up one of our experimental environments (the avatar controller, the OpenSimulator server, and the Second Life viewer) on an ordinary laptop PC (Windows XP/Vista).

Each guest user logins the server through network and watches the 3D scene from the arbitrary own viewpoint.

Our avatar controller is one of client software. It logins to the server, and then it pretends a human and controls the avatar. The avatar controller itself doesn't have a scene viewer, however other guest users can watch the avatar figure. Thus, it can run on an old PC with modest graphics performance. The optional viewer is used for monitoring the 3D scene if necessary.

3.2. Avatar controller

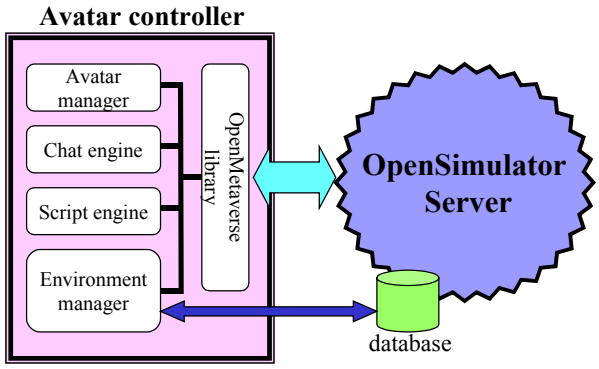

Figure 3: Avatar controller

The avatar controller consists of 5 modules.
Figure 3 shows these modules in the avatar controller.

1. Avatar manager
   It decides and controls avatar's behavior in cooperation with other modules.
2. Chat engine
   It receives a chat text from other avatars through a chat channel, and replies it. It supports both English and Japanese.
3. Script engine
   It manages various presentation scripts. In addition to presentation texts, it controls the avatar's position, action, and images displayed on a screen along the scripts.
4. Environment manager
   It has two important roles.
   - It tracks the behavior of all avatars in the world.
   - It converts object names to UUIDs (Universal Unique ID).
5. OpenMetaverse library
   All modules above access the OpenSimulator server through this library. The environment manager has also another communication channel with the server for the efficiency.

Figure 4: A screen shot of our prototype system

Figure 4 is a screen shot of our prototype system.

There are two avatars. The avatar controller controls the avatar in front of a screen, and a guest user controls another one (foreground) manually. In order to distinguish two avatars, we call the avatar that the "**user**" controls manually, the "**guest**" avatar; and one that our system controls with software, the "**cast**" avatar. Two avatars can communicate through chat channel. The cast avatar can change the images of screens by chat or along with the presentation script.

4. DISCUSSION

4.1 User's avatar behavior and body direction

Many of avatar interface systems, in general, assume that a single user takes a seat in front of the PC display over the presentation. Once the user clicks a start button for the presentation, the avatar proceeds a presentation till the end of a script, even if the user gets bored and walks off in the middle. It is because there is no way of getting the user's state.

In the 3D virtual world, a user's avatar behavior and body direction gives us much information. On the other hand, in the pervasive 2D avatar system like the Microsoft Agent and 3D avatar system with fixed viewpoint, the user can't have own avatar figure. That is, the avatar system can't observe user's behavior.

The guest avatar can walk freely to anywhere at any time in the 3D world. When the guest avatar comes to the cast avatar, we guess that the user may want to have communication with the cast avatar. If the guest avatar suddenly turns back and walks away during the presentation, we know that the user gets bored or has no interest in the contents of the presentation.

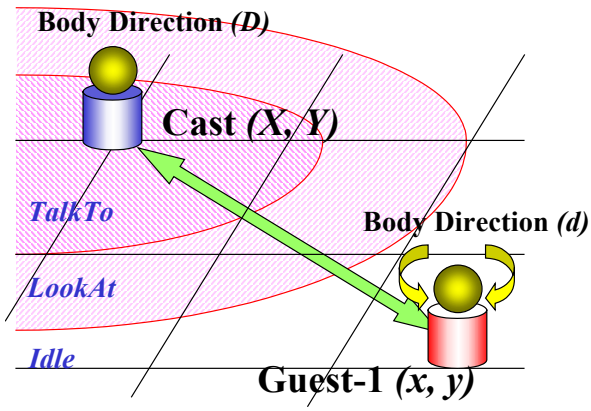

Figure 5: Avatar position and body direction

When the user logouts the server, the guest avatar also disappears. If the user keeps login but leaves the seat, the guest avatar becomes "away" state.

In our system, the environment manager periodically accesses the OpenSimulator server and gathers the position and direction information of all avatars in the world.

Figure 5 shows avatar position and body direction.

1. If the guest avatar is far away, the cast avatar takes random idle action and walks around the presentation stage.
2. When the guest avatar is approaching to the cast avatar and comes within the predefined distance, the cast avatar turns the body to the guest avatar. The user may find out the cast avatar looks at the guest avatar.
3. If the guest avatar moves closer to, the cast avatar says to the guest avatar, "hello", "hi", "how are you", and so on.
4. If the guest avatar moves closer to but steps back, the cast avatar says, "excuse me".
5. If the guest avatar ignores the word from the cast avatar and looks another direction, the user has another interest.
6. If the guest avatar turns back and walks away, the cast avatar breaks the presentation, and say, "thank you" and "good bye".

4.2. Chat engine

The chat engine receives a chat text from other avatars, and then replies it.

As the chat engine, we adopt two programs. One is a simple pattern-matching program with regular expressions written in C# language, and another is an AIML (Artificial Intelligence Markup Language) engine [1]. The AIML is an XML dialect for creating natural language software agents, and its performance has been highly appreciated. It is also free software, and

we can get sample implementations written in various programming languages and some AIML sets (AIML "brain" files).

In our system, received texts from other avatars are sent to the pattern-matching program first, and then if matching is failed it passes to the AIML engine.

AIML works very well for English texts, however, it is not necessarily efficient for Japanese linguistically. Since Japanese texts are spelt in several different ways, the number of matching patterns increases rapidly. Japanese usually use a mixture of different types of characters, kanji characters, hiragana and katakana phonetic scripts, and sometimes the Roman alphabet. And in Japanese sentences, subjects are often missing.

The AIML set affects the performance of AIML. Some groups apply AIML to Japanese, however the public AIML set for Japanese are very few.

4.3. Mapping from name to UUID

The environment manager converts the name appeared in the presentation script to UUID (Universal Unique ID).

UUID is a 16-byte unique number (32 hexadecimal digits). All objects in the 3D world, e.g. prim (primitive object), avatar, texture, etc, have own UUID. The OpenSimulator server manages all objects using UUID internally, however it is inconvenient and frustrating for us to write a presentation script that includes UUID.

For example, in Figure 4, in order to show the image "whether_map" on the display "left_screen", our low level script is simple,

```
Display  whether_map, left_screen
```

The environment manager converts the name "whether_map" and "left_screen" to those UUIDs, and then sends the command to the server.

The name is not unique in the virtual world. If two objects have the same name, our system chooses one randomly.

We have prepared a direct access mechanism to the database of the server. Client software always gets all information from the OpenSimulator server with Second Life protocol. In some cases, however, it may not be efficient because of the packet structure. For example, UUID-to-name conversion is easy. Since UUID is unique, the number of the return value is only one. However, name-to-UUID conversion may take much time. It retrieves the name properties of all objects first since many objects may have just the same name.

We can select the data from the database by the SQL statement, and get result data only. It is tricky way, but it is efficient and enables to provide fast response to the user. And it is independent of the server implementation that is changing fast every day.

5. CONCLUSION

In this paper, we presented our design and implementation of our interactive presentation system with 3D avatars. Using the OpenSimulator server for a 3D virtual environment is an attractive approach, although it is not stable yet. The 3D avatar interface systems missed common high-level platform technologies, like the Microsoft Agent for 2D avatars. The OpenSimulator will encourage making fascinating 3D environments. The 3D avatar interface system simulates our real life in the 3D virtual world, and then we can get user's state by observing the user's avatar behavior.

Future work will focus on creating autonomous agent with human-like behavior and natural communication ability in the 3D virtual world.

6. REFERENCES

[1] "Artificial Intelligence Markup Language (AIML) Version 1.0.1", *A.L.I.C.E. AI Foundation Working Draft*, 8 August 2005 (rev 008), 2005

[2] Daden limited, http://www.daden.co.uk

[3] Friedman, D., Steed, A., Slater, M.: "Spatial Social Behavior in Second Life", *International Conference on Intelligent Virtual Agents 2007 (IVA-2007)*, LNCS (LNAI), vol. 4722, pp.252–263, Springer, 2007

[4] Hayashi, M., Ueda, H. and Kurihara, T,: "TVML (TV program Making Language) - Automatic TV Program Generation from Text-based Script –", *ACM Multimedia'97 State of the Art Demos*, 1997

[5] 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*, vol. 24, Issue 4, pp. 233–245, 2007

[6] Maes, P.: "Agents that Reduce Work Overload and Information Overload", *Communications ACM*, pp. 31-40, 1994.

[7] Microsoft Agent, http://msdn.microsoft.com/en-us/library/ms695784(VS.85).aspx
(See also, "Microsoft Agent Software Development Kit and Design tools", *Microsoft Press*, 1997)

[8] 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*, vol. 49, pp. 1–15, 2004

[9] OpenMetaverse Foundation, http://www.openmetaverse.org

[10] OpenSimulator Main Page, http://opensimulator.org/wiki/Main_Page

[11] Second Life official site, http://secondlife.com

[12] Ullrich, S., Bruegmann, K., Prendinger, H., Ishizuka, M.: "Extending MPML3D to Second Life", *International Conference on Intelligent Virtual Agents 2008 (IVA-2008)*, LNAI, vol. 5208, pp. 281–288, Sprinter (2008)