Proceedings of the IIEEJ Image Electronics
and Visual Computing Workshop 2012
Kuching, Malaysia, November 21-24, 2012

AN INTERACTIVE 3D AVATAR PLATFORM ON THE OPENSIM SERVER CONTROLLED WITH IRONPYTHON LANGUAGE

Hiroshi Dohi   Mitsuru Ishizuka

Dept. of Information and Communication Engineering,
Graduate School of Information Science and Technology, University of Tokyo, JAPAN
dohi@mi.ci.i.u-tokyo.ac.jp

ABSTRACT

We have developed a new prototype of an interactive 3D avatar interface platform on the OpenSim server. It is redesigned and implemented using IronPython programming language. Our previous prototype was written in C# language, and it was modest to update scripts for avatar interaction. IronPython is an open-source implementation of Python language on the Microsoft .Net framework. We can make use of both expressive scripting ability of Python and various functions of OpenMetaverse access library written in C#. Our scripts for describing dialogue and avatar behavior are also written in IronPython. It can modify part of scripts dynamically. The combination of IronPython and OpenMetaverse library is suitable for developing interactive 3D avatar interface.

1. INTRODUCTION

Avatar interface in 3D virtual space is becoming popular. A user controls his/her own avatar, and communicates with other avatars (users) in common 3D virtual world over network connection. It is also called "3D Internet." This is real-time and interactive (bi-directional) communication style like our daily life.

"Second Life" [12] is a famous network service for providing 3D virtual space with avatars that uses high quality computer graphics. According to the report released by Linden Lab. [16], the Second Life economy remained stable and Q3 2011 web merchandise sales volume grew 30% as compared with Q3 2010. Money supply (Linden $) is equivalent to 29.3M US$.

About 2007, all sorts of companies had entered into Second Life for seeking new business opportunities. They compete to open the Second Life branch of their companies on 3D virtual world, but they shortly closed the branch and withdrew from Second Life quietly. As a result, Second Life is often considered as a failure for business platform.

It is important to recognize that 3D avatar interface has different features from pervasive WWW. It isn't a simple extension from 2D to 3D.

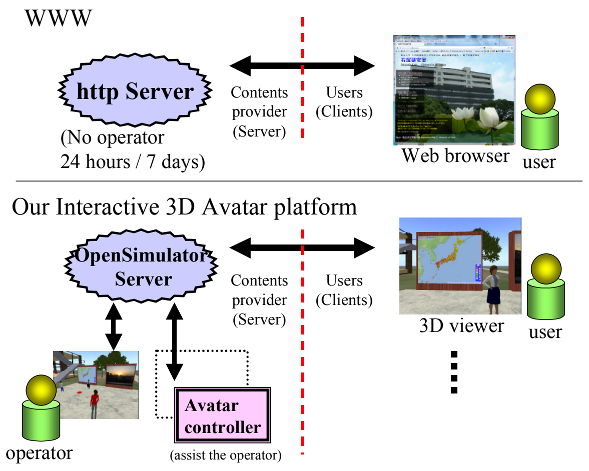

Fig. 1. Interactive 3D Avatar platform

Figure 1 shows WWW and interactive 3D avatar platform.

In WWW, a user searches necessary information unilaterally. Once you set up web pages, the WWW server can accept requests from all over the world 24 hours / 7 days without an operator. Internet technology has broken down barriers of time and distance. But it is difficult for 3D avatar interface. It is because the operator usually controls the avatar manually, and communicates with the user. It spontaneously arises "business hour" and "hot spots" in each place.

When we access the web page, there is generally no way to know how many users are accessing the page at the same time. The WWW server usually closes its network connection immediately after it downloads web data.

In 3D avatar interface, everybody knows easily how many avatars are in the same virtual world. Nobody would like to walk through a desolate ghost town even if it is in the virtual world.

Avatar interface is expected as one of post WWW media. We have developed a new prototype of an interactive 3D avatar interface platform on the OpenSim server. The avatar controller controls the avatar, and it will assist the operator.

In our previous prototype [3], we had a plan to use simple low-level script language for describing avatar interaction. However, we have found that it requires more expressive description ability.

We have redesigned our prototype system, and the new one is implemented with IronPython programming language [5]. Scripts for describing dialogue and avatar behavior in 3D virtual space is also written in IronPython.

2. A 3D VIRTUAL SPACE

2.1. OpenSimulator (OpenSim) server

We have built our new prototype system on the OpenSimulator (OpenSim) server [10].

OpenSim is an open source 3D application server. The source code of OpenSim is more than 500k lines (September 2012).

It has client compatibility with Second Life protocol. It means that we can access the server with a free Second Life official viewer that has already been distributed widely for the Second Life server.

In Second Life, a large number of servers are connected to each other in a network and it forms huge 3D virtual space. It is called "grid mode."

OpenSim supports "standalone mode" too in addition to grid mode.

In standalone mode, it runs single server process on a local computer. We can get independent 3D virtual space. Network connection is available.

A computer-controlled avatar is often prohibited in the virtual world of game. (Currently Second Life doesn't prohibit the computer-controlled avatar unless it put extra load on the server.) We don't have such restrictions in standalone mode.

On the other hand, OpenSim is yet alpha version. The stability of the OpenSim server is modest. Both bug fixes and adding new features are made in parallel, and the source code is modified almost everyday. It may occasionally require some patches.

2.2. Scene update

OpenSim uses a server-client model. Many avatars (users) share common 3D virtual space. Each avatar can have arbitrary viewpoint.

When a user logins to the OpenSim server, an avatar emerges in 3D space. One client viewer corresponds to one avatar. Each viewer has the template of the basic 3D avatar model, and receives visual parameter packets from the server for updating the scene. Then each viewer generates the scene image from own viewpoint at the client-side.

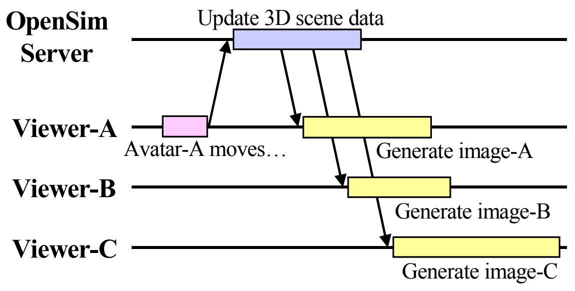

Fig. 2. Packet transfer for scene update

Figure 2 shows packet transfer for scene update.

We can't expect that the avatar actions be synchronized on all viewers. The exact timing specification may be meaningless. For example, when the user moves the avatar position on the viewer-A, its information packet is sent to the OpenSim server. Next, the server sends back new avatar position information to all clients one by one. It is not a broadcast packet from the server. All clients didn't get information simultaneously. Then, each client viewer generates and updates the scene image independently. The complexity of the scene image and graphics performance is different on each viewer.

2.3. Avatar control without manual operation

There are mainly two ways to control avatars and objects without manual operation.

1. Embedded scripting language
2. Original client software

Second Life has embedded scripting language, called "LSL (Linden Scripting Language)". LSL script is attached to a prim (primitive object) on 3D space, not to the avatar directly. It is run by event-driven. There are more than 300 functions.

In the OpenSim server, it supports "OSSL (OpenSim Scripting Language)" in addition to the LSL.

The embedded scripting language is easy to use, however the code size is restricted. Some LSL functions delay script execution when they are called. It is convenient to add a little event-driven feature, but it is not suitable for describing complex dialogue and avatar behavior.

Another method is creating original client software that controls the avatar without manual operation. It needs to connect the server with Second Life protocol. The client is not necessarily a viewer. We chose this method. OpenMetaverse access library [9] is useful for creating own original client. It is open source, and the source code is about 340k lines (September 2012).

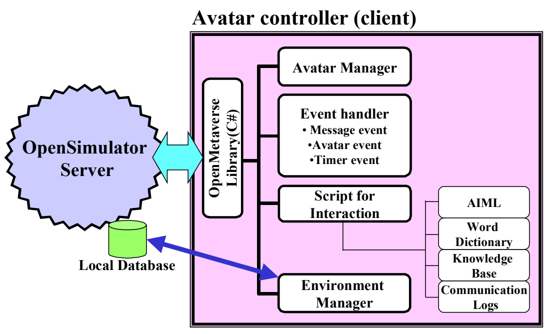

Fig. 3 System configuration

3. A DESIGN AND IMPLEMENTATION

Our avatar controller is one of client application for the OpenSim server. (Figure 1) It doesn't have viewer function.

We assume that the avatar is in contents-provider side. The avatar acts as an assistant or a guide for the user. In Second Life / OpenSim, the avatar must have own land for any activities since it is not allowed to put any private objects in public space nor change in public terrain.

Figure 3 shows our new prototype system configuration.

3.1. IronPython

Our new prototype system is written in IronPython programming language [5].

(Original) Python is one of famous scripting languages (e.g. Ruby, Perl, and PHP etc). Some large application programs that handle big data (e.g. the Google App Engine and the Facebook etc.) use it. It is also used for system configuration scripts on some Linux distribution, Fedora and Ubuntu.

IronPython is an implementation of Python language on the Microsoft .Net framework. Microsoft had maintained IronPython, but it moved to open-source community in 2010.

The salient feature of IronPython is that it can access .Net library directly. That is, we can make use of both expressive scripting ability of Python and various functions of OpenMetaverse library written in C#.

3.2. System Configuration

The avatar controller consists of five modules. Because of the expressive scripting ability of IronPython, most of avatar control functions move to both the event handler module and the scripts for interaction module in our new prototype.

Avatar Manager The avatar manager has interface to OpenMetaverse library. It defines basic avatar behavior. It encapsulates and hides the raw function of OpenMetaverse library API from scripts.

Event Handler When any changes in 3D virtual world (e.g. the avatar speaks or moves) are informed to all clients, the event handler invokes an appropriate script.

Scripts for Interaction It is script pool for interaction. The scripts are also written in IronPython.

Environment Manager The environment manager manages avatars and objects in 3D virtual world. In addition to OpenMetaverse channel, it has another direct access channel to the local database of the OpenSim server. It can get properties of the object using SQL commands.

OpenMetaverse library Second Life protocol is very complex. We use OpenMetaverse library in order to control avatars. It is open source, and written in C#.

3.3. Event-handler

Our avatar operations are basically event driven. We handle three types of events.

1. Message events
2. Avatar events
3. Timer events

In this paper, we call the avatar that is controlled manually by the user "guest (avatar)", and the computer-controlled avatar is called "cast (avatar)."

3.3.1. Message events

When the guest speaks something (chat-typing), it invokes an appropriate reply script. The script is an IronPython function with two parameters, the guest name "*name*" and its message text "*mesg*" in this example.

This is a very simple example.

```
def simple_reply(name, mesg):
  if "hello" == mesg.lower():
    cast.say("Hi, %s" % (name))
```

The *"%s"* above will be replaced with the guest name. For examples, the guest "miv" says (types) "Hello", the cast will reply "Hi, miv".

In practice, we don't usually use the simple "==" operator but more sophisticated way. Regular expressions are available for complex string matching. It is easy to write random selection method.

3.3.2 Avatar events

When the guest moves, for example, avatar-event-handler is invoked with two parameters, position and direction. It calculates distance between the guest and the cast. Avatar direction is also recorded. Both distance and direction give much information.

If the guest approaches to the cast, we find that the guest may want to have communication with the cast. On the contrary, if the guest walks away from the cast, the rest of the presentation will be canceled immediately.

The guest may get bored with the conversation or find another interesting thing, if the guest looks another direction during chat.

If the guest approaches too close to the cast, the cast will shout and step back.

3.3.3. Timer Event

The cast responds to guest behavior. If the guest is near the cast and has no action, the cast may not be able to act properly.

Timer event is raised at regular time intervals. The cast begins action independent of the guest behavior when the cast doesn't receive any event in a little while.

3.4. Script for Interaction

The script pool has 4 sub module / databases.

- AIML
- Word dictionary
- Knowledge base
- Communication logs

AIML (Artificial Intelligence Markup Language) [1] is an XML dialect for creating natural language software agents. Most AIML interpreters are released as open source software. When the message text doesn't match any dialogue scripts, it will pass to the AIML module. It will return the plausible reply text. Its performance depends on the data set.

Others (Word dictionary, Knowledge base, and Communication logs) are simple text databases that are helpful to generate reply text.

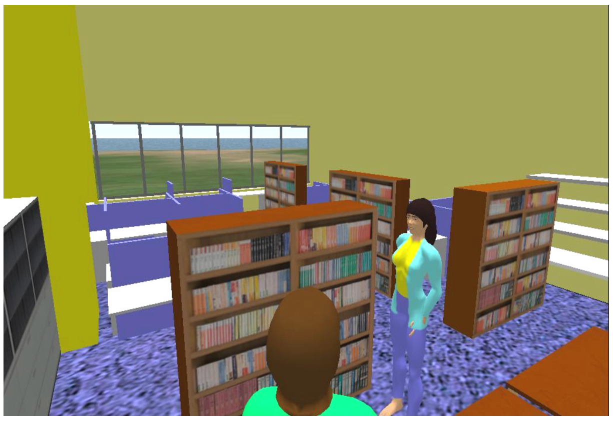

Fig. 4. Snapshot of our prototype system

Figure 4 shows a snapshot of our prototype system. We replicated our lab room in the 3D virtual world. This virtual room is made about 100 primitive (cube) objects with textures in total, including surrounding walls, glass windows, some bookracks, cabinets, more than 20 desks with partitions, and so on. There are two avatars in the room. One is a computer-controlled avatar.

4. DISCUSSION

4.1. Scripting in IronPython

Our previous prototype system is written in C# programming language. It is because related software, both the OpenSimulator server and OpenMetaverse access library, are implemented in C#. It was modest to modify and update dialogue scripts.

IronPython is powerful script language, and it unites Python and Microsoft .Net.

In our new prototype, scripts for interaction are also written in IronPython. That is, scripts are part of our system. It can simplify the script parser. Scripts can make use of just the same control flow and data structure of Python directly.

Besides, IronPython supports an interactive console with fully dynamic compilation. Hence we can modify and update part of script code dynamically while our software is running.

Both natural language processing and superior behavior control are indispensable elements for developing the autonomous avatar.

Python is often used for natural language processing, and it has already made many useful tools and support libraries. We can make use of such effective resources too.

Some agent interface systems (e.g. MPML3D [11][14], VHML [7] etc.) adopt XML-based scripting language. And some languages for making interactive application (e.g. AIML[1], VoiceXML[17]) also use XML style format. We think that XML-based scripting language is not necessarily easy to write scripts for non-computer science professionals. In practice, avatar interaction doesn't usually run according to script. It requires computer-programming knowledge to handle flexible control flow and complex regular expressions in any way.

XML-based language may be sometimes inefficient to describe dialogue. AIML is developed for the A.L.I.C.E. (Artificial Linguistic Internet Computer Entity) system, which won the annual Loebner Prize Competition in Artificial Intelligence three times. However, AIML has weak pattern matching ability. It doesn't permit more than one matching pattern per each basic block <category> even if the reply is the same. It may often accumulate huge numbers of simple pattern. It is not easy to maintain the AIML data set.

AIML standard description examples:

```
<category>
<pattern>HELLO</pattern>
<template>Hi, there</template>
</category>

<category>
<pattern>HOWDY</pattern>
<template>
<srai>HELLO</srai>
  (it means the same response with “HELLO”)
</template>
</category>
```

Python descriptipn examples:

```
import re        # once in a file
m = re.compile(“hello|howdy”, re.IGNORECASE)
if m.search(mesg):
    cast.say(“Hi, there”)
```

IronPython / Python support more complex regular expressions. They have also interface to useful parser library and database access library.

4.2. IronPython and OpenMetaverse

The combination of IronPython and OpenMetaverse library is flexible and effective.

Since IronPython seamlessly integrates with .Net framework, it can make use of various functions of OpenMetaverse library without any modification.

The “DevoBot” project [13] uses IronPython and OpenMetaverse library. It provides a base framework for writing Second Life bots that are controlled via commands sent through instant messages. The main goal of the DevoBot project is to allow extremely rapid development by providing the ability to modify source code without having to recompile it. Its IronPython code is just less than 0.4k lines. It may require the knowledge about OpenMetaverse library if you want to add original function.

The “PyOGP” [15] is another open source project to explore a suite of python based Second Life client libraries. The code aims to produce an automated testing framework. “Py-” is a prefix that means Python library, and “OGP” is an abbreviation of Open Grid Protocol.

It is written in Python language, not IronPython. PyOGP has known problems with various environments. The packages that make up PyOGP have some dependencies on python modules not included in a standard install.

5. CONCLUSION

We have presented our new prototype of the interactive 3D avatar interface platform on the OpenSim server. Our research goal is to create the autonomous avatar in 3D virtual space.

IronPython seamlessly integrates expressive scripting ability of Python and .Net framework. The combination of IronPython and OpenMetaverse library is flexible and especially effective to make 3D avatar interface in the OpenSim server. Script for avatar interaction is also written in IronPython, and it is part of our system. It can make use of valuable resources that have been developed so far for Python application. It will reduce the load of development for interactive 3D avatar interface.

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] H. Dohi and M. Ishizuka, “An Interactive Presentation System in a 3D Virtual World using an OpenSimulator Server”, *Proc. IIEEJ Image Electronics and Visual Computing Workshop (IEVC-2010)*, Mar. 2010

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

[5] “IronPython”, http://ironpython.codeplex.com

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

[7] A. Marriott, “VHML—Virtual Human Markup Language,” *Proc. Talking Head Technology Workshop, at OzCHI Conf.*, 2001

[8] 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)

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

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

[11] H. Prendinger, S. Ullrich, A. Nakasone, and M. Ishizuka, “MPML3D: Scrripting Agents for the 3D Internet”, *IEEE Trans. Visualization and Computer Graphics*, Vol. 17, No. 5, pp 655-668, May, 2011

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

[13] M. Stephen, “devobot – Base framework for libsl bots written in Python”, http://code.google.com/p/devobot

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

[15] “PyOGP”, http://wiki.secondlife.com/wiki/PyOGP and http://bitbucket.org/enus_linden/

[16] “The Second Life Economy in Q3 2011”, http://community.secondlife.com/t5/Featured-News/The-Second-Life-Economy-in-Q3-2011/ba-p/1166705 , 2011

[17] “Voice Extensible Markup Language (VoiceXML) 2.1”, *W3C Recommendation 19 June 2007*, http://www.w3.org/TR/voicexml21, 2007