

User Pluggable Animation Components in Multi-user Virtual Environment

Yu-Lin Chu, Tsai-Yen Li, and Cheng-Chia Chen

Computer Science Department, National Chengchi University, Taipei, Taiwan

Abstract—One of the key functions to popularize 3D content services on virtual environments is on the extensibility of the underlying system. In our previous work, we had implemented an extensible Multi-User Virtual Environment (MUVE) called IMNet which features plug-in modules for various application purposes. This feature is enabled by adopting the eXtensible Animation Modeling Language (XAML) as the underlying language for incorporating external modules. Nevertheless, the system needs to be restarted for a new module to take effect. In this paper, we describe an attempt to use the OSGi framework to dynamically load user-designed animation modules that can be distributed and plugged in at run time. We will describe examples in IMNET to demonstrate the process of designing and distributing animation components at run time in this framework. In addition to allowing 3D modules to be contributed by the users, we believe that this function is also crucial for the realization of semantic virtual environment in the future.

Index Terms—Multi-user Virtual Environment, OSGi framework, Extensible Animation Language, Semantic Virtual Environment.

I. INTRODUCTION

Multi-User Virtual Environment (MUVE) usually refers to a 3D environment that allows multiple users to log in concurrently and interact with each other with multiple modalities, such as texts, images, audio, and animations, provided by the system. The prevalent 3D online games are one type of virtual environment with specific game logics embedded. In recent years, virtual environments of general purpose, such as Active World [10] and Second Life [13], have attracted great interest and become more and more popular. Most of these virtual environment systems allow users to choose their own geometry and appearance, and some of them allow the users to access object information in the environment and program their avatars according to some proprietary scripting languages. However, few can provide the users with the freedom of customizing their behaviors such as how their animations are generated according to the procedures designed by the users.

We think in the era of Web 2.0, for a MUVE to become successful, it is crucial to allow the users of virtual environments to contribute 3D contents of their own design to the community such that other users can also see their behaviors or animations via the procedures that can be downloaded and run

```
<AnimItem model="avatar" playMode="seq">
  <AnimHigh>Walk to Door</AnimHigh>
  <AnimImport src="PushDoor" />
  <AnimItem>
    <AnimItem model="door" DEF="DoorOpening" ...
    <AnimPlugin><Audio src="scream.wav" />
  </AnimPlugin>
  </AnimItem>
</AnimItem>
<AnimImport src="Shock" />
</AnimItem>
```

Fig. 1 Example of using XAML to specify animations by composing animation components at various levels

users to select and display. Even though some of the existing virtual environment systems allow new modules to be added to enhance its functionality, the system usually needs to be restarted for the new modules to take effect. This inconvenience could greatly hinder the possible contributions from the users for customized behaviors.

IMNET is an XML-based MUVE using the client-server architecture that has been developed in our previous work [8]. This MUVE system features the use of an extensible animation scripting language called eXtensible Animation Modeling Language (XAML)[7], an XML-based language that allows multiple levels (e.g. low, middle, and high levels) of animation components to be composed in various ways. For example, as shown in Fig. 1, <AnimItem> is the most basic tag for animation component in XAML while <AnimHigh> and <AnimImport> are tags for specifying higher level animation commands and animations stored in external files, respectively. Another design feature of this language is its extensibility for incorporating other languages as plug-ins. For example, in Fig. 1, <AnimPlugin> is used to incorporate the <Audio> tag that is only known to an external module which is designed to process it. This external module needs to be registered in a configuration file when the system starts up. A new module can be installed by adding an entry to the configuration file. However, the system needs to be restarted for the new module to take effect. As a result, it is not feasible for the users to distribute their procedures on the fly via this mechanism.

In this paper, we extend our MUVE system, IMNET, to allow animation procedures to be loaded at run time such that users can develop their own 3D animation procedures and share them with other clients in the same virtual environment.

We have chosen to use the OSGi framework to enable this run-time plug-in mechanism. When the IMNET browser processes an incoming XAML script and encounters an unknown tag, the system will download the code specified as an attribute of the tag and install it in the OSGi framework inside IMNET. We will use several examples to illustrate the process of the installation and the usage of this plug-in mechanism.

The rest of this paper is organized as follows. In the next section, we will survey the research pertaining to our work. Then we will describe how we incorporate the OSGi mechanism into IMBrowser (a 3D browser in IMNET) in Section III. In Section IV, we will use two examples to illustrate the usage of this mechanism. We will then conclude the paper with discussion of how to extend the mechanism to realize semantic virtual environment.

II. RELATED WORK

The development of multi-user virtual environment has a long history. These virtual environments are sometimes called shared virtual environments or collaborative virtual environment. Several multi-user virtual environments have been proposed in the literature. For example, RING[5], DIVE[4], and MASSIVE[6] are a few typical MUVE's developed in academia. Other commercial ones include ActiveWorld[10], Blaxxun[12], and Second Life[13]. In [2], the author pointed out the challenges that collaborative virtual environments needed to face. For example, it is a challenge to allow participants to collaboratively work on a same task under the constraint of limited network bandwidth. Several attempts have been made in the literature to address this issue. Some of them focus on designing more scalable network architectures, such as peer-to-peer [10] or hierarchical connection model [3] while others focus on reducing the required amount of data transmission for a given application [8]. In IMNET, high-level XAML scripts have been designed with the goals of reducing data transmission as well as increasing the level of animation control.

In traditional MUVE's, the design of 3D contents and system functions are all centered around human users instead of machines. For example, most 3D environments use the concept of scene graph (tree) to store and manage the geometric information for the graphics objects to be displayed to the user. Human users can understand the properties of the objects according to their geometry and appearance through visualization. For example, a human user can easily interpret an object consisting of a flat surface attached with four cylinders as a table. However, it is difficult for a computer program to understand the semantics of this arrangement. The same problem also occurs on the web when we try to automate the computational tasks that need to be processed among several computers. This is the reason that the concept of "semantic web" was proposed to study how to design web services with semantic information [18].

Along the same direction, the concept of *semantic virtual environment* was proposed to address the semantic issue that is missing in current virtual environments [17]. A semantic virtual environment can be treated as a regular 3D environment with additional semantic information associated with the objects in the virtual world. The same technologies, such as RDF

and Ontology, used to model web services can also be used to model 3D semantics. In addition to allowing different virtual worlds to be interconnected, semantic information of virtual environment can be used by the users to develop their own code to interact with the environment and other avatars. However, in order to enable user-designed code to be plugged in at run time, adopting an appropriate component-based framework is much desirable.

Among the related work on semantic virtual environment, Otto [9] has developed a shared virtual environment called SEVEN that makes use of reusable software components for different virtual environment systems. In this paper, creating reusable software components is also our design goal but our focus is more on generating customizable behaviors for the avatars according to the given virtual environment. Abaci et al.[1] used the concept of "smart objects" to design a way of expressing semantics for animations in addition to geometry.

One of the key functions for realizing the aforementioned semantic virtual environment is how to dynamically download and install software components and how to allow them to interact with each other according to the interfaces that they agree upon. OSGi (Open Services Gateway Initiative) is the technology that we have chosen to realize such a mechanism. The specification of OSGi framework was designed by the OSGi Alliance for deploying and executing software components [14][15]. The objectives of this framework include increasing component reusability, enhancing platform independency, and reducing software complexity. The Equinox platform used in Eclipse is one successful implementation of the OSGi framework [13] and is also the one that we have adopted in our virtual environment.

III. SYSTEM ARCHITECTURE DESIGN

The core component of IMNET is a Java3D-based 3D browser called IMBrowser that is capable of parsing and processing XAML scripts for animations. Based on this browser, we have extended XMAL to include a mechanism for dynamic installation of a new component specified in the URL attribute of a new tag. This dynamic installation of components is enabled by an embedded OSGi framework and is connected to the 3D browser for on-the-fly generation of animations.

Fig. 2 is the architecture of the XAML animation system implemented in IMBrowser. The system consists of a XAML parser, an animation manager, a resource controller, and the newly added OSGi framework. The syntax of an XAML script is first checked by the parser and then converted into an internal animation tree containing the information of priority, timing, and animation content for each node. In the process of converting a XAML script into an animation tree, each tag is associated with a processing procedure that can convert the tag into its corresponding node in the animation tree. In the original system, when one would like to extend XAML with a new tag, the conversion program for the tag must be registered in the parser in advance. In the new system reported in this paper, when an unknown tag is encountered, the system will first look it up in the OSGi framework for appropriate conversion program. If no such programs are found, the codebase attribute of the tag will be used to download and install the

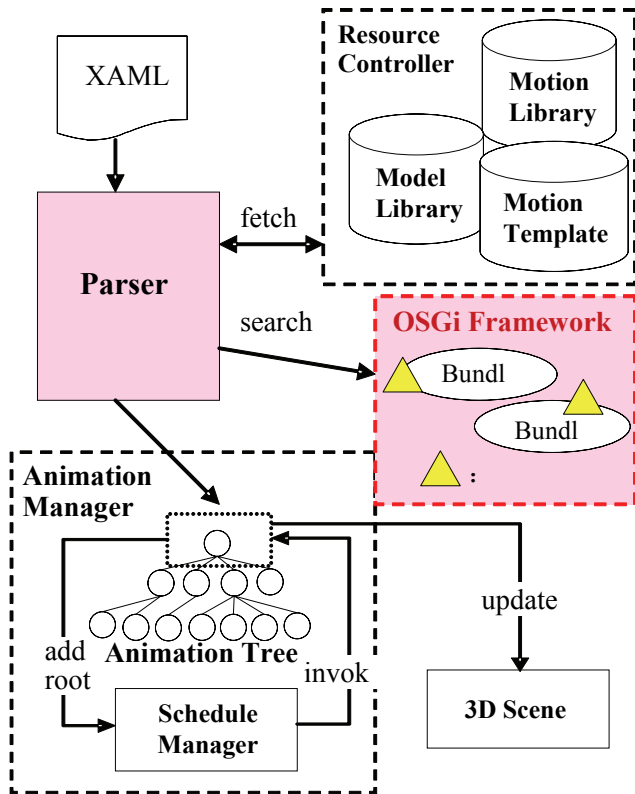


Fig. 2 Architecture of the XAML animation system

code for processing this tag.

The OSGi framework provides a management mechanism for symbolic names of components such that no bundles (referring to software components in OSGi) with the exactly the same name and version number can be installed on the same framework. In addition, in Equinox, the system persistently records the bundles that have been installed and can recover the status even after the system is restarted. Therefore, the bundles that have been installed once do not need to be downloaded even after the system is rebooted. In the following subsections, we will first describe the process of the original parser and then the process after the OSGi framework is introduced.

3.1 Original parsing process

When processing a XAML script, the AnimationFactory module in IMBrowser converts each animation element `<AnimElem>` (and its successors) into an AnimData tree structure that may recursively contains other AnimData nodes. As shown in Fig. 3, a process rule is associated with each of the animation tags including AnimHigh, AnimTransition, AnimImport, etc. For example, the module for the AnimTransition tag is used to generate the transition animation from the current key frame to the specified target key frame. The module for AnimImport takes a file name as its attribute to import an animation script stored in a separated file. If necessary, these modules may acquire information about the geometry or transformation of the object under consideration in order to generate appropriate animations for it. The proc-

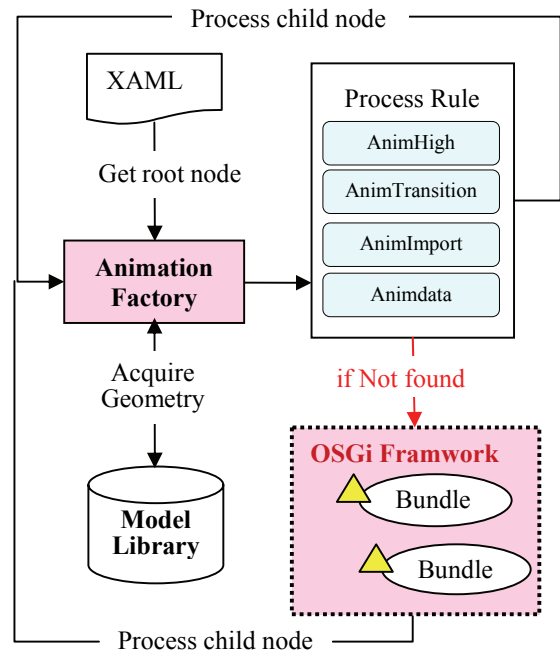


Fig. 3 The parsing process of the XAML system with the new OSGi framework

essing of these high-level commands result in animations specified in a lower level script sent back to the animation manager for updating the 3D scene. Every time when a new conversion procedure is designed, the tag and its associated code need to be installed and registered manually in IMNET. This new module will not take effect until the system is restarted.

3.2 New parsing process with OSGi

On a MUVE, software updates cannot happen too often if we would like to keep the MUVE system available most of the time. Therefore, we have designed a new mechanism with the OSGi framework to enable run-time installation and activation. When IMNet is started, we also start an OSGi framework and allow the AnimationFactory module to interact with the OSGi framework as shown in Fig. 3. When the AnimationFactory module finds an unknown tag, it first searches the OSGi framework for a registered OSGi service associated with the given tag name. If the service does not exist at the time of lookup, the system will automatically download and install the component according to the given URL in the attributes.

IV. IMPLEMENTATION AND ILLUSTRATIVE EXAMPLE

In this section, we will use an example to illustrate how a user-designed “bow” action can be executed at all clients at run time. A snapshot of the bow action created by the example is shown in Fig. 4. In this illustration, we will distinguish the roles of developer and user who are located at different clients. We assume that the developer will design an appropriately formatted bundle and upload it onto the server. Other clients will then be able to parse the script containing this tag and generate appropriate animations on their client programs.

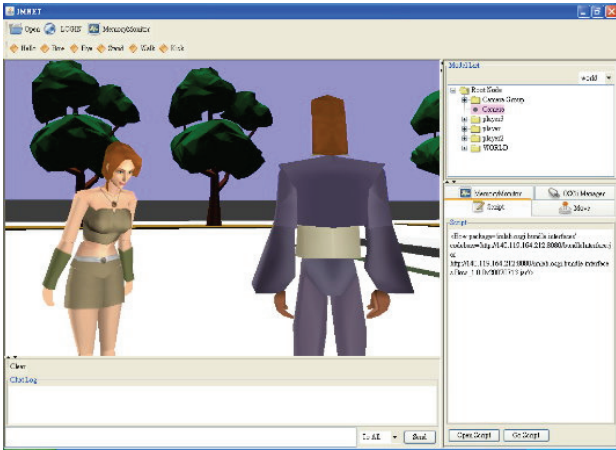


Fig. 4 Snapshot of the client interface issuing and displaying the bow action.

```

Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: bow_bundle Plug-in
Bundle-SymbolicName: bow_bundle
Bundle-Version: 1.0.0
Bundle-Activator:
imlab.osgi.bundles.MyBowActivator
Bundle-Localization: plugin
Import-Package:
org.osgi.framework;version="1.3.0"
Export-Package:
imlab.osgi.bundle.interfaces
    
```

Fig. 6 An example of OSGi manifest file

```

<Bow package='imlab.osgi.bundle.interfaces'
codebase='http://imlab.cs.nccu.edu.tw/li/bow.jar' />
    
```

Fig. 7. An example of how to use a tag containing bundle information

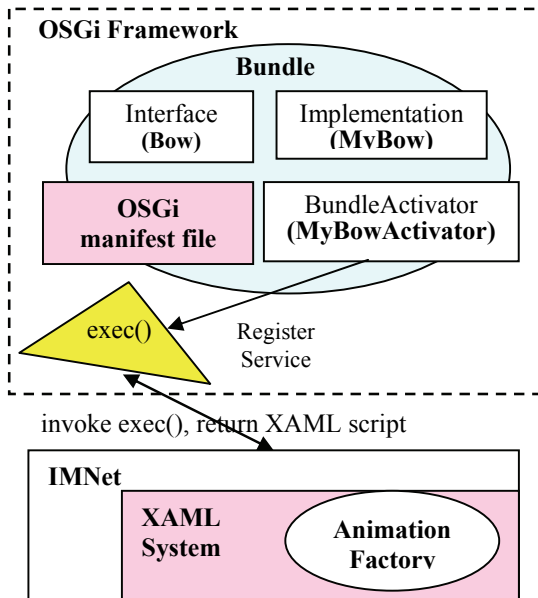


Fig. 5 Construct of a bundle and its connection to IMNet

4.1 Designing an OSGi Bundle

The design of an OSGi bundle consists of four components: *interface*, *implementation*, *activator*, and *manifest*, as shown in Fig. 5. In this example, the name of the service as well as the interface is called *Bow*. This interface is implemented by the *MyBow* class. When the `exec()` function of this interface is invoked, an XAML script is returned to the animation engine for execution. In addition, we have to implement the *BundleActivator* interface, a standard OSGi interface, for activating a bundle by calling the `start()` function in the interface. In our example, the *MyBowActivator* class implements this interface and declares an object of *MyBow*, called *bower*, in the member function of `start()`. This *bower* object is registered with the OSGi framework by calling the standard `registerService()` function.

We also need to set up the manifest file for the bundle as shown in Fig. 6. From this manifest file, we know the name and version of this bundle, `bow_bundle_1.0.0`, which needs to be unique in an OSGi framework. In this file, we set the *BundleActivator* to the `imlab.osgi.bundles.MyBowActivator` class and export the `imlab.osgi.bundle.interfaces` package for other components in the same framework to use this bundle. Finally, all these related files will be packed into a JAR file for download.

4.2 Bundle Upload, Download and Installation

The created bundle can be uploaded to the web server provided by IMNET via the client interface. When a bundle is uploaded, it is stored in a directory named after the developer, and this name serves as the namespace for distinguishing the bundles created by different developers. Nevertheless, the developer can choose to put the bundle in any legal URL. A sample usage of the bundle as an extension of XAML is shown in Fig. 7. It will also output the usage as a string for the developer to copy or distribute.

As an IMNet user, the developer can issue this XAML script to its avatar through a window (lower left corner in Fig. 4) provided by the client interface. This script will be broadcast to other clients at run time through the IMNet protocol. When a script such as the one shown in Fig. 7 is received by an IMNet user, the *AnimationFactory* will resort to the OSGi framework because the tag is not recognized as a standard one in XAML. The OSGi framework is first searched for a service called `imlab.osgi.bundle.interfaces.Bow` to see if it is already installed. If not, the code specified in the `codebase` attribute will be downloaded. The *AnimationFactory* module will then call the `install()` command in *Equinox* to install `bow.jar` into the OSGi framework. After the installation, the `start()` function in *bundleActivator* is called to register the bundle service. Once the ser-

```

<MoPlan package='imlab.osgi.bundle.interfaces'
codebase='http://imlab.cs.nccu.edu.tw/plan.jar'>
  <param name="-s" value="1.1 2.3"/>
  <param name="-g" value="5.2 3.8"/>
</MoPlan>

```

Fig. 8. An example of specifying a motion planning problem in a user-defined module.

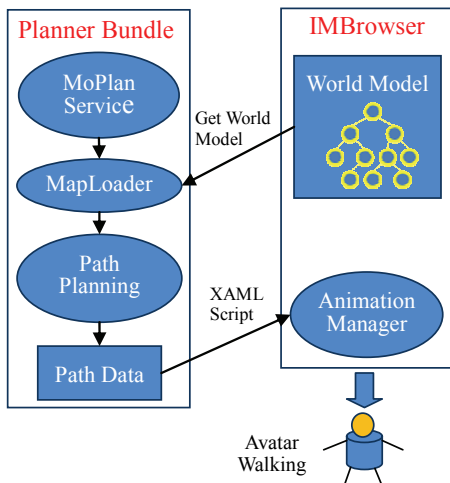


Fig. 9 Relation between IMBrowser and an external plug-in module (a motion planning bundle in this case)

vice is started, the AnimationFactory module will call the `exec()` function in the Bow service to generate main contents of the animation as a XAML script which is a low-level bow animation in this case.

V. ANOTHER EXAMPLE

In this section, we will use the example of a customized path planner to demonstrate the mechanism of allowing dynamic pluggable components in IMNet. We assume that a user has designed a motion-planning service that can generate collision-free paths by specifying the initial and goal configurations of its avatar for a given geometric description of the objects in the virtual environment. The tag for invoking the service is defined as `<MoPlan>` which can be downloaded and installed from the given URL as in the previous example. After the motion planning module is uploaded to the server, any user can take advantage of this module by issuing an XAML script similar to the one in Fig. 8. After downloading and installing the plug-in module, other users will run the path planner by first loading the geometric information from the virtual world and generate a collision-free path for the avatar to follow as shown in Fig. 9. This path is specified in a low-level XAML script that is then sent to the animation manager for execution. In Fig. 10, we show an example of the collision-free path that has been generated by the planner to avoid the obstacles (trees) in the environment.

VI. CONCLUSION AND FUTURE EXTENSION

Multi-user virtual environments have attracted many atten-

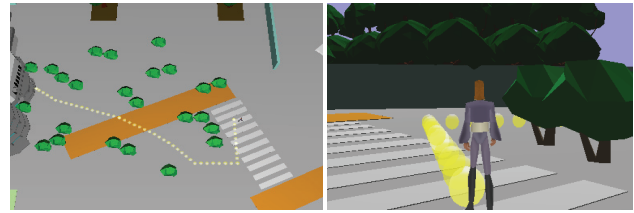


Fig. 10 Top view and side view of the 3D scene with the path generated for the avatar by the planner

tions in recently years because of the rich interactivities that can be enabled in the virtual world. However, a user usually is not allowed to customize his/her behaviors due to the lack of a flexible plug-in mechanism. In this paper, we have presented an implementation of virtual environment that is designed to be extensible to external plug-in modules. This dynamic pluggable ability is facilitated by incorporating the OSGi framework into IMNet. As the era of semantic web and web 2.0 comes, we believe that this extensibility will become crucial to allow the sharing of 3D contents and animations in MUVE. We have used two examples to illustrate the usage of this mechanism, and we believe that more customized animation modules will be created by users and shared by all users in the future.

The mechanism of dynamic installation and invocation is a necessary function for realizing semantic virtual environment where virtual agents should be able to retrieve semantic information from the environment or other agents. Currently we are working on making IMNET a semantic virtual environment. For example, we are designing the ontology of the objects in the virtual environment to contain semantic information such as geometry, attributes, and functions. We will design standard interfaces for a user's module to inquire semantic information and take appropriate actions after reasoning on the acquired information. We are also planning to allow the virtual agents to communicate with each other through some known ontology of virtual agent. With these new functions, we hope that realistic behaviors of a virtual agent can be generated automatically in IMNET in the near future.

ACKNOWLEDGEMENT

The authors would like to acknowledge the support of this work from National Science Council of Taiwan under a contract NSC 96-2221-E-004-008.

REFERENCES

- [1] T. Abaci, J. C'iger. Action semantics in Smart Objects, in *Proceedings of Workshop towards Se-mantic Virtual Environments (SVE 2005)*, 2005.
- [2] S. Benford, C. Greenhalgh, T. Rodden, and J. Pycock. Collaborative virtual environments, *Communications of the ACM*, vol. 44, no. 7, pp. 79–85, 2001.
- [3] A.M. Burlamaqui, M.A.M.S. Oliveira, L.M.G. Goncalves, G. Lemos, J.C. de Oliveira. A Scalable Hierarchical Architecture for Large Scale Multi-User Virtual Environments, in *Proc. of 2006 IEEE Intl. Conf. on Virtual Environments, Human-Computer Interfaces and Measurement Systems*, pp. 114-119, 2006.
- [4] E. Frecon and M. Stenius. DIVE: A Scalable network architecture for distributed virtual environments, *Distributed Systems Engineering*

- Journal (Special issue on Distributed Virtual Environments)*, Vol. 5, No. 3, pp.91-100, September 1998.
- [5] T. Funkhouser. Network Topologies for Scalable Multi-User Virtual Environments, in *Proceedings of IEEE VRAIS'96*, p.222-228, April 1996
- [6] C. Greenhalgh and S. Benford. MASSIVE: a collaborative virtual environment for teleconferencing, *ACM Trans. CHI*, Vol.2, No.3, pp.239-261, Sep 1995.
- [7] T.Y. Li, M.Y. Liao, J.F. Liao. An Extensible Scripting Language for Interactive Animation in a Speech-Enabled Virtual Environment, in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME2004)*, Taipei, Tai-wan, 2004.
- [8] T.Y. Li, M.Y. Liao, P.C. Tao. IMNET: An Experimental Testbed for Extensible Multi-user Virtual Environment Systems, *ICCSA 2005, LNCS 3480*, pp. 957-966, 2005.
- [9] K. A. Otto. The Semantics of Multi-user Virtual Environments, in *Proceedings of Workshop towards Semantic Virtual Environments (SVE 2005)*, 2005.
- [10] A. Steed, C. Angus. Enabling scalability by partitioning virtual environments using frontier sets, *Presence: Teleoperators and Virtual Environments*, Vol. 15, No.1, pp.77-92, February 2006.
- [11] ActiveWorlds, <http://www.activeworlds.com>
- [12] Blaxxun, <http://www.blaxxun.com>
- [13] Equinox, <http://www.eclipse.org/equinox/>
- [14] OSGi Alliance, <http://www.osgi.org/>
- [15] OSGi Technical Whitepaper, <http://www.osgi.org/documents/collateral/OSGITechnicalWhitePaper.pdf>
- [16] Second Life, <http://secondlife.com/>
- [17] Semantic Virtual Environments, <http://page.mi.fu-berlin.de/otto/sve/index.html>
- [18] Semantic Web, <http://www.w3.org/2001/sw/>



Yu-Lin Chu received his BS in 2006 from National Chengchi University, Taiwan. He is currently a graduate student in the Computer Science Department of National Chengchi University, and is advised by Prof. Li and Prof. Chen. He is a member of Intelligent Media Laboratory led by Prof. Li. His main research interest is the enabling technologies for rich interactions among virtual agents in semantic virtual environment.



Tsai-Yen Li received his BS in 1986 from National Taiwan University, Taiwan, and MS and Ph.D in 1992 and 1995, respectively, from Stanford University. He is currently a Professor in the Computer Science Department of National Chengchi University in Taiwan. His main research interests include Computer Animation, Intelligent User Interface, Motion Planning, Virtual Environment, and Artificial Life. He is a member of IEEE, ACM, and TAAI of Taiwan.



Cheng-Chia Chen received his BS in 1983 from Department of Electrical Engineering, National Taiwan University, Taiwan, and MS and Ph.D in 1985 and 1993, respectively, from Department of Computer Science and Information Engineering, National Taiwan University. He is currently an Associate Professor in the Computer Science Department of National Chengchi University in Taiwan. His current research interests focus on Model-Driven Engineering, Semantic Web Service, XML technology, and rapid software tool development.