Extendable Simulation Framework for Virtual World Environment based on the DEVS Formalism

Chang Beom Choi[1], Se Jung Kwon[1], Tag Gon Kim[1],
Jae Hyun Lim[2], Dong-Hyun Baek[3] and Soohan Kim[4]

[1] KAIST, 291 Daehak-ro, Yuseong-gu, Daejeon 305-701, Republic of Korea
{cbchoi, sjkwon}@smslab.kaist.ac.kr,
tkim@ee.kaist.ac.kr
[2] Network Customizing Technology, 3-1 Doryung-dong, Yuseong-gu, Daejeon 305-733, Republic of Korea
jhlim@netcus.com
[3] Kyungwon Univ., 69, Seongnam-Daero, Bundang-gu, Seongnam-si, Gyeonggi-do, Korea
Youngtong-gu, Suwon, Kyunggi-do, Republic of Korea
dhbaek@kyungwon.ac.kr
[4] SAMSUNG Electronics HQ 416, Youngtong-gu, Suwon, Kyunggi-do, Republic of Korea
ksoohan@samsung.com

Abstract. A virtual world is an interactive virtual environment in which users interact with each other with computers, and can be used as a platform for virtual training activities. In order to enhance the trainee's immersive experience, domain-specific simulation models are required for virtual world services. For this reason, we propose an extendable simulation framework for the virtual world. The simulation framework is composed of Core Simulation Framework and Virtual Level Architecture. By utilizing the Core Simulation Framework and the Virtual Level Architecture, the content creator of the virtual world can create extendable simulations for domain-specific content.

1 Introduction

Generally, a virtual world is an interactive virtual environment in which users interact using their own graphical representations, known as avatars. In the virtual world, users interact with each other using computers, and they can collaboratively create and edit objects in the virtual world without meeting each other and interacting in the real world. Therefore, the virtual world has been effectively utilized in education, especially in virtual training.

Virtual training is a training method in a computer-simulated environment. The virtual world allows users to access educational content from their homes or work places. In order to enhance the effectiveness of education via the virtual world, the virtual world services utilize physics engines. Usually, the physical simulation modules are embedded in the game engine. However, there are several limitations of supporting simulation models in the virtual world environment. To support immersive experiences for the trainee, domain-specific simulation models are required, such as the weather simulation model, intelligent virtual agents, and war-game simulation models. In

order to provide domain-specific simulation content to the virtual world, the virtual world must embrace the simulation engine as a component. As the simulation content evolves, the simulation engine should embrace various levels of simulation models.

To give an immersive experience to users of the virtual world, we propose a simulation framework based on the discrete event system specification (DEVS) formalism [1]. The DEVS formalism specifies discrete event models in a hierarchical and modular manner. Therefore, the user may model the target system by decomposing the large system into smaller components by applying coupling schema to them. The proposed simulation framework consists of Core Simulation Framework, which manages and executes the DEVS simulation model, and Virtual Level Architecture, which support discrete event simulation and distributed simulation.

The rest of this paper is organized as follows: Section 2 introduces existing virtual worlds and simulation models of the virtual worlds. In Section 3, we propose the extendable simulation framework. Then, we introduce the case study. In the case study, we introduce the prototype of the firefighting training system. Finally, we conclude the paper.

2 Related Works

2.1 Second Life

Second Life® is an online virtual world that enables users to interact with each other via avatars [2]. Because Second Life supports in-world object creation, users can build various objects with primitive 3D polygonal shapes. In addition, Second Life supports the procedural script language so that users can add interactive behaviors to the object and the avatar.

Second Life was utilized as an educational platform at various institutions. For example, Loyalist College utilized Second Life to help Customs and Immigration students experience the daily routine of their future jobs [3]. Moreover, in the scientific community, Second Life is used for collaboration and data visualization [4]. Since the virtual world can overcome time and regional restrictions, Second Life was utilized as a suitable platform for real-time meetings with other people to share knowledge. Second Life utilizes the Linden Scripting Language [5] and Havok physics [6] to support the simulation of various machines and devices. However, Second Life has limited features for the domain-specific simulation. First, memory for a single script is restricted to 64 KiB. Therefore, the domain-specific simulation model may not be loaded in Second Life. Second, Havok physics is not designed as a simulation engine; it is sufficient to handle real-time collision and dynamics of rigid bodies, but not sufficient to handle domain-specific simulation in the virtual world.

2.2 OpenSim

The objective of OpenSim is to develop the full open-source server to host virtual world services using existing Second Life clients [7]. One characteristic of OpenSim is that the server can easily expand through plug-in modules, and it has been utilized in various researches.

OpenSim supports several physics simulation engines such as Open Dynamics Engine [8] and BulletX [9]. By supporting several physics simulation engines, OpenSim can handle physics simulation in the virtual world, such as detecting the collision of objects and solving the dynamics of rigid bodies. However, OpenSim does not consider domain-specific simulation in the virtual world. Therefore, OpenSim does not contain the simulation engine necessary to handle the domain-specific simulation model.

3 Extendable Simulation Framework for Virtual World

When the content creator of the virtual world wants to utilize realistic simulation models in the virtual world, the virtual world service framework should find the simulation models and send them to the virtual world servers. In order to support the virtual world service framework to handle various level of the simulation model, the virtual world service framework must provide simulation model management service. The simulation model management service is composed of storing the simulation model, searching the simulation model, transferring the simulation model, and controlling the simulation model. The store and search services assist the authors with registering the simulation model and searching the proper simulation model for the authors' intent. When the simulation models are decided, the virtual world service framework initializes the virtual world server with simulation models by transferring the simulation models from the simulation model storage to the server. After the initialization of the simulation models, the authors of the virtual world control the simulation using the simulation model control service.

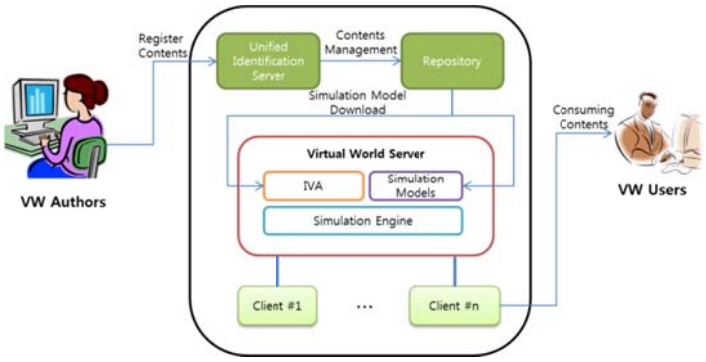

Fig. 1. Conceptual figure of the virtual world service framework

Figure 1 conceptualizes the virtual world service framework. The role of the Unified Identification Server (UIS) is to handle account information, manage content for the virtual environment, and search the account information and content. The Repository server is used to manage assets of the virtual world, such as item information for avatars, objects in the virtual world environment, and simulation models. If the content creator wants to create virtual worlds with domain-specific simulation models, the virtual world server must contain a simulation engine that supports simulations. Moreover, the server should manage the virtual world's users, the simulation models, and the simulation engine. In this virtual world service framework, when the authors of the virtual world request the simulation models, the UIS sends the simulation model request to the Repository server. Then the Repository server transfers the corresponding simulation models to the virtual world server and the virtual world server initializes the domain-specific simulation models. However, if the virtual world server contains the simulation engine as a module, supporting various simulation engines may be limited, and the managing simulation engine and the executing simulation model may influence the overhead of the virtual world server.

Therefore, we propose an extendable simulation framework for the virtual world. Figure 2 shows the conceptual figure of the extendable simulation framework used in the virtual world service framework. As shown in the figure, the simulation engines are separated from the virtual world server. The benefits of separating the simulation engine from the server are twofold. First, various simulation engines can be applied in the virtual world without modifying the virtual world server. Second, when the content creator wants to utilize more than two domain-specific simulations in the virtual world, the extendable simulation framework can support more than two simulation engines to generate simulation results.

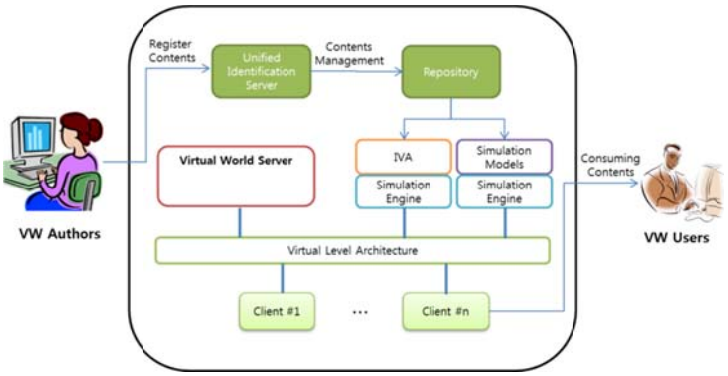

Fig. 2. Conceptual figure of the extendable simulation framework

In order to support domain-specific simulation models, we propose the Core Simulation Framework (CSF) and Virtual Level Architecture (VLA). The CSF supports the domain-specific simulation, and the VLA supports the interoperation between the CSF and the virtual world server.

3.1 Core Simulation Framework

The CSF consists of the Model API layer, the simulation engine, and the system layer, and supports the extendable simulation environment based on the plug-in feature. Figure 3 shows the conceptual architecture of the CSF.

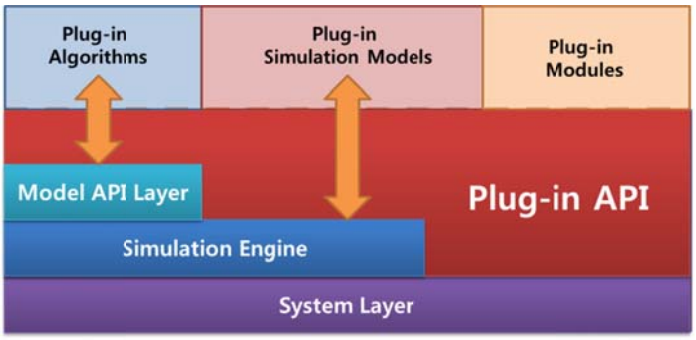

Fig. 3. Core Simulation Framework

The system layer of the CSF controls the management modules, such as network communication, simulation model initialization, and simulation engine control modules. The simulation engine of the framework may depend on the simulation domain. As a result, the system layer of the CSF supports the plug-in API to change the simulation engine based on the simulation domains [10].

The CSF's simulation engine should support the plug-in feature for the simulation models. In this research, we adopted the simulation engine for DEVS formalism. By adopting the formalism, the authors of the simulation model can reuse the domain-specific simulation models, or change the domain-specific algorithms for the simulation model by applying a layered approach [11].

3.2 Interoperation of Simulation

Interoperation between the CSF and the virtual world server could be achieved by using the Virtual Level Architecture (VLA). The VLA supports interoperation among simulation servers and the virtual world server by providing core services, such as data synchronization and time synchronization.

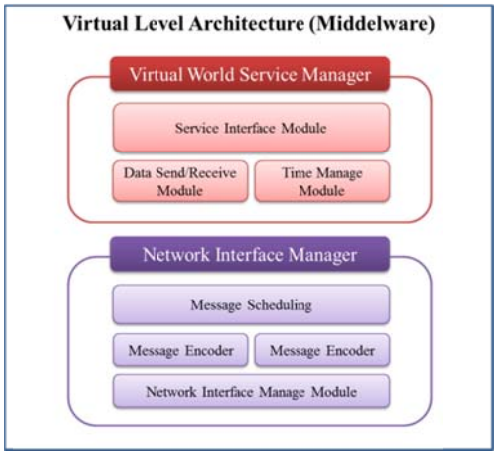

Fig. 4. Conceptual figure of the Virtual Level Architecture

Figure 4 denotes the managers of the VLA. The VLA manages various levels of services in the virtual world framework, such as system configuration control, system execution control, content management, and content streaming services. In order to handle these services, the VLA provides the API with time synchronization and data synchronization. Therefore, the developer of the virtual world server or the simulation model can develop each component without considering time and data synchronization.

In order to support interoperation between the simulation engine and the virtual world server, the service interface module in the virtual world service manager provides the interface to handle data synchronization and time synchronization. In order to support the time synchronization between the virtual world server and the simulation engine, the time synchronization algorithms are implemented in time management modules. For data synchronization, the data send/receive module manages the data based on the time. In order to support virtual world service manager, the network manager handles message scheduling and encoding/decoding the network packets.

4 Case Study

The purpose of this case study is to build a prototype of the firefighting training simulator. Modeling and simulation experts and the domain experts collaborated with each other to develop the firefighting training simulator [11]. The modeling and simulation experts built the simulation models and implemented the simulation framework using VLA, and the domain experts consulted on the development of the fire simulation model. Figure 5 shows the conceptual design of the simulator. Each simulation server has fire simulation models and engines, and every simulation server is interconnected with its neighboring simulator through VLA.

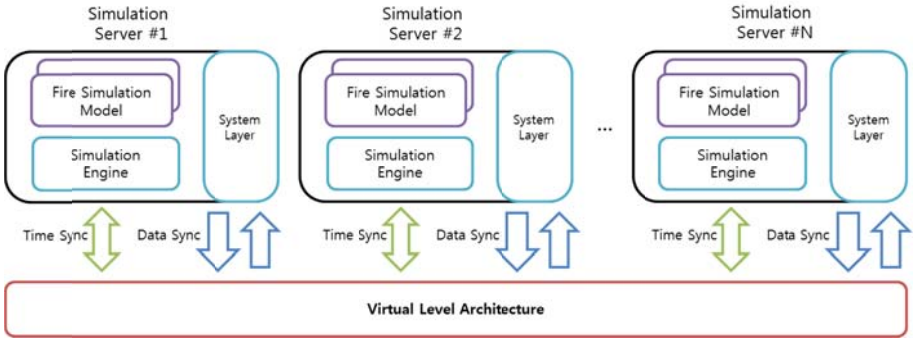

Fig. 5. Conceptual architecture of the Fire-Fighting Training Simulator Prototype

The prototype simulates spaces inside a certain building virtually. A trainer can make scenarios with flammable materials, non-flammable materials, different starting points for the fire, and so on. As the fire starts at some point and spreads, several trainees should move and extinguish the spreading fire. From the simulator, the fire-fighter can experience these diverse scenarios virtually. The design of the overall simulation models including the fire models is described in Figure 6. The coupled model consists of the *UserController* model, the Transducer model, and the grid of Fire models.

The *UserController* model is controlled by user interface, and has the positions of the users (trainee). The positions (i,j) that users locate means that the user tries to extinguish the flames in the *Fire model* (i,j). Hence, the *UserController* model sends extinguishing messages to proper *Fire models* through the *toFireModel* ports. The *Fire models* are interconnected to adjacent Fire models through *in_east*, *in_west*, *out_east*, *out_west*, and so on. The flames on the models spread to other models by sending messages of the fire model to adjacent to the other fire models.

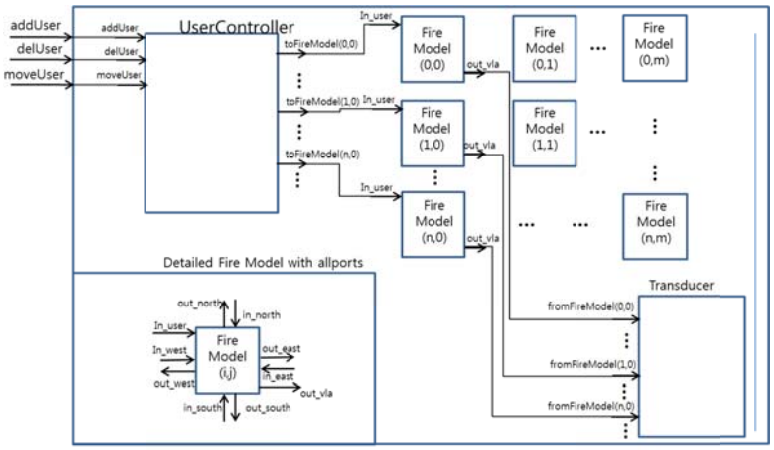

Fig. 6. Overall Coupled Model for Firefighting Simulation

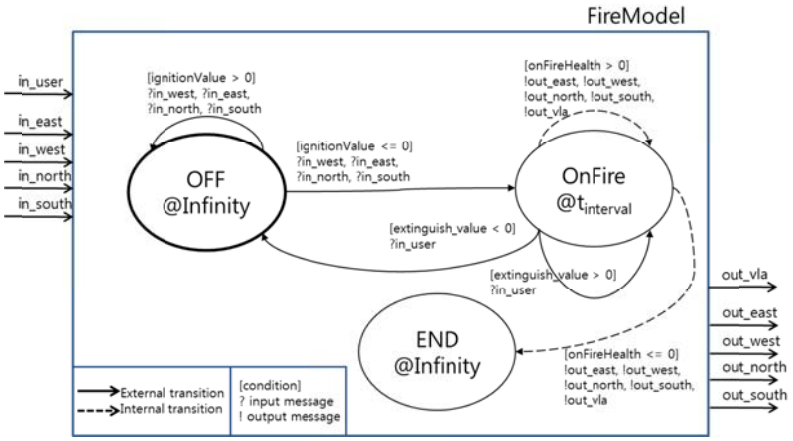

Fig. 7. Design of a Fire Model

The design of the fire model as an atomic model is described in Figure 7. The atomic model has three states: *OFF*, *OnFire*, and *END*. *OFF* is an initial state or a state with an extinct fire. *OnFire* is that the space is burning. *END* state is when the flammables in the space are burned down. There are three variables of state transitions: *ignitionValue*, *extinguish*_value, and *onFireHealth*.

- *ignitionValue*: A value representing the level for ignition of the cell. Reduced by the message from the other fire models.
- *extinguish*_value: A value representing the level for extinguishing the flames completely in the cell. Reduced by the extinguishing message from the *in_user* port.
- *onFireHealth*: Duration of the flames. Reduced as the flames are held on the space.

When the fire model on the edges of the grid send messages, some messages of output ports are sent to another simulation server. Through the *out_vla* port, the message is sent to the *Transducer* model. The *Transducer* model collects the messages and sends them to other servers through VLA.

Figure 8 is a captured picture of the four-simulation server. On the left side of the figure, there are four GUI windows for each server. Black squares indicate non-flammable materials such as walls, and green squares represent flammable materials. In this figure, red flames are spreading in several places. If the red flames arrive at the edge of the server, we can see the red flames spreading to other server.

On the right side of the figure, there are GUI windows for the user, the trainee. Users' avatars are depicted by simple yellow circles. As trainees move their avatars on the GUI windows, they can increase their experience in extinguishing the building's flames.

Fig. 8. The captured picture of execution in the four-simulation server

5 Conclusion

This paper introduces an extendable simulation framework for the virtual world. The virtual world is a computer-based interactive virtual environment where users interact with each other through computers. In order to give a trainee an immersive experience, the virtual world's content may be base on the domain-specific simulation models. In order to support the extendibility of the simulation framework, we propose a simulation framework based on the DEVS formalism. Therefore, the creator can utilize the existing simulation models by utilizing the DEVS-based simulation models. The simulation framework, which consists of the Core Simulation Framework and the Virtual Level Architecture, allows the user to reuse the simulation models.

The Core Simulation Framework embraces plug-in features so that the simulation framework can plug-in various modules during run-time. The Virtual Level Architecture handles time synchronization and data synchronization among components of the virtual world services. Regarding future research, we will extend the simulation framework to support extendable virtual world environments.

Acknowledgement

This research is supported by Ministry of Culture, Sports and Tourism (MCST) and Korea Creative Content Agency (KOCCA) in the Culture Technology (CT) Research & Development Program 2009.

References

1. Zeigler, B.P., Kim, T.G., Praehofer, H.: Theory of Modeling and Simulation. Academic Press Orlando, FL (2000)
2. Second Life: http://www.secondlife.com
3. Rippin, P., 2009. Virtual World Simulation Training Prepares Real Guards on the US-Canadian Border: Loyalist College in Second Life. Linden Lab. , Source: <http://secondlifegrid.net.s3.amazonaws.com/docs/Second_Life_Case_Loyalist_EN.pdf>. Accessed [date accessed here].
4. Lang, A.S., Bradley, J.C.: Chemistry in Second Life, Chemistry Central Journal 3 (2009) 14, doi:10.1186/1752-153X-3-14.
5. Linden Scripts Language. http://wiki.secondlife.com/wiki/List.
6. Havok Phsyics: http://www.havok.com/
7. OpenSim: http://www.opensimulator.org
8. Open Dynamics Engine: http://www.ode.org/
9. BulletX: http://xnadevru.codeplex.com/
10. Bae, J. W., Kim, T. G.: DEVS Based Plug-in Framework, Spring Simulation Multiconference 2010, Orlando, FL (2010) 147–153
11. Sung, C.H., Hong S.Y. and Kim, T.G.: Layered Structure to Development of OO War Game Models Using DEVS Framework, Proceedings of Summer Computer Simulation Conference, Philadelphia (2005) 65–70.