An Underwater Vehicle Simulator with Immersive Interface using X3D and HLA

Pilwon Hur, Jeongsam Yang, Soonhung Han and Byounghyun Yoo

SIMULATION 2009; 85; 33
DOI: 10.1177/0037549708101180

An Underwater Vehicle Simulator with Immersive Interface using X3D and HLA

Pilwon Hur
Department of Mechanical Science and Engineering
University of Illinois at Urbana-Champaign, USA

Jeongsam Yang
Division of Industrial and Information Systems Engineering
Ajou University, South Korea
San 5, Wonchun-Dong
Yeongtong-Gu Suwon, 443–749, South Korea
jyang@ajou.ac.kr

Soonhung Han
Department of Mechanical Engineering
Korea Advanced Institute of Science and Technology, South Korea

Byounghyun Yoo
Modeling, Virtual Environments and Simulation Institute
Naval Postgraduate School, Monterey, California, USA

Training war fighters how to maneuver a submarine is very important, but the use of real submarines in such training is expensive and hindered by regional and temporal limitations. Modeling and simulation (M&S) can be a good alternative to costly training. However, the use of existing M&S for submarines has limitations. For instance, the commercial software and hardware need to be kept in a secure place. Depending on the location of the protected software, war fighters may travel far and take a considerable amount of time to get to these places. Long-distance travel also means they have a limited amount of time to train on the M&S machines. Furthermore, many types of M&S have only a one-channel display system, which reduces immersiveness. Another problem is that few heterogeneous simulators can be used as an integrated system. One solution to these problems involves the use of extensible 3D (X3D) graphics, a platform-independent and open standard graphic file format, which can be used with general purpose PCs. The immersiveness is increased by means of a multichannel display system and a motion chair. Finally, the individual components of a simulator can be integrated with the high level architecture and run time infrastructure (HLA/RTI). We demonstrate the proposed method through experiments with an underwater vehicle simulator.

Keywords: high level architecture, modeling and simulation, motion chair, multichannel, extensible 3D (X3D) graphics

1. Introduction

Submarines are more than a conventional weapon system of the sea. They are an independent weapon system replete with underwater-launched guided weapons with surface-target capabilities. Effective operation of a asubmarine equipped with deadly weapons can determine whether a war is won or lost; hence, the training of submariners is critical. Training on real submarines is the most effective

SIMULATION, Vol. 85, Issue 1, January 2009 33–44

DOI: 10.1177/0037549708101180
Figures 1, 2, 6–8 appear in color online: http://sim.sagepub.com

form of instruction because it enables the submariners to feel the sounds and vibrations as well as directly operate the equipment. However, this type of training is problematic because submarines are expensive, they can only be operated in the sea and training of this type also takes several months. Modeling and simulation (M&S) is an alternative training method that uses virtual reality (VR) to simulate the motion of a submarine and a submarine-like training environment.

The speed of a newly developed submarine can be measured under water by submerging the submarine and propelling it with maximum power. If this can be done, there is no need for M&S. However, when a crew is learning to evade a torpedo, real tests onboard a submarine endanger the crew. The use of M&S offers a safer alternative. The strategic training of submariners is often handicapped by the cost of commercial visualization tools and the dedicated hardware required [1]. The management of this kind of M&S system requires special secured locations and the maintenance is costly. Such requirements also entail the inconvenience of transportation as well as time limitations if the secured location is in a remote area.

This study proposes an integrated system for an underwater vehicle simulator with increased compatibility and reusability. To build the system, we constructed a low-cost platform-independent visualization system instead of a conventional high-cost visualization system for an M&S submarine; increased immersiveness with multichannel visualization and a motion chair instead of using single-channel visualization; and adopted a framework that can synchronize different simulators or simulation components. The remainder of this paper is organized as follows: Section 2 introduces the related studies for the visualization method of the VR and the M&S that was carried out for a military drill. Section 3 elaborates the integrated system for an underwater vehicle simulator based on high level architecture (HLA). Section 4 describes an underwater vehicle simulator developed in an iCAVE which is a computer aided virtual environment (CAVE) system located in Korea Advanced Institute of Science and Technology (KAIST) and shows the experimental results of the simulator. Finally, the conclusion summarizes the contents of this study and discusses future research directions.

2. Related Studies

2.1 Visualization Method for Implementing a Virtual Environment

Various graphics libraries and formats have been used for the visualization of a 3D VR environment. Low level graphics libraries, such as OpenGL and Direct3D, have a high rendering quality and directly access graphics hardware through a device drive interface (DDI). The low level of these libraries means they rely on a complicated implementation and optimization process. They also have poor portability into other programs. Java OpenGL (JOGL), which is a wrapper library that allows OpenGL to be used in the Java programming language, has the characteristics of low level graphics libraries such as OpenGL and Direct3D.

High level graphics libraries, such as Vega, Java3D and Open Scene Graph (OSG), have recently been used for the visualization of VR. Vega is a visualization development toolkit for real-time simulation; it has improved the functionality of Performer, which is a rendering toolkit based on OpenGL. Christianson used one 3D model to compare the difference between Vega and Java3D visualizing [2]. Although Vega is an expensive platform-dependent tool, it performs better than Java3D in terms of the frame rate for continuous scenes. Even though Java3D is free and platform-independent, it does not perform as well as Vega.

Java3D, a high level graphics library for supporting Web-based visualization, was developed mainly to function with Java on top of OpenGL or Direct3D. It is a suitable visualization method for platform-independent and low-cost visualization. Salisbury used Java3D for simulation visualization on the Web [3]. However, because some of the functions that are available in OpenGL may not exist in Java3D, the task of controlling and rendering scenes is at times impossible. In addition, although low level graphics libraries, such as OpenGL, are used for the rendering, all other functions, such as program logic, still use Java. Objects are created in the Java runtime, the Java3D runtime, and the building of application programs. Thus, Java3D is less efficient in terms of performance and speed.

OSG, a 3D graphics development toolkit based on OpenGL, was developed by the object-oriented method. This toolkit is easier to understand and more convenient to develop than Vega and Performer. It is also reusable in various platforms because it is being developed in standard C++ and open source as well. The only issue with OSG is that it is difficult to use for Web-based visualization.

The methods mentioned above that use graphics libraries also use graphic formats for saving VR objects. The VRML97 (Virtual Reality Modeling Language), which supports dynamic scene-graphs, is an international standard for 3D visualization. Li et al. developed a Web-based VR simulator for surgery training [4]. They used VRML for the visualization format and an External Authoring Interface (EAI) for dynamic simulation. In another study, Robert and Knight proposed multichannel visualization, which enables the same screen to be viewed simultaneously from different perspectives [5]. Unfortunately, VRML has large files because it cannot manage each scene node, which consists of a scene-graph. Thus, VRML includes all the nodes, some of which are unnecessary for some purposes. Systematic construction of semantic information is also difficult for VRML.

The Synthetic Environment Data Representation and Interchange Specification (SEDRIS) is used as a means for exchanging different data formats. SEDRIS is equip-

ped with a database for static data, enabling semantic information about environmental data and relational information to be expressed clearly. Moon and Han used SEDRIS to construct a simulation environment for digital manufacturing [6].

The extensible 3D (X3D) proposed by the Web3D consortium uses XML technology to represent VR objects. Because X3D is based on XML, it can systematically construct semantic information and provide interoperability with standards such as MPEG4. Moreover, X3D improves the visualization quality by adding nodes for high-quality rendering. With smaller files in a compressed binary format, the speed is enhanced and execution is possible on the Web [7]. Due to the benefits in X3D, many research projects on X3D visualization have recently been performed [8–11]. We used X3D as a graphic format for visualization and as a browser. In addition, we used Xj3D with Java3D so that scene-graphs could be used in the upper layer of OpenGL or JOGL, enabling the implementation of a platform-independent system.

2.2 Research on M&S for Military Training Purposes

The US Department of Defense (US DoD) carried out work on simulator networking (SIMNET) in order to make low-cost simulators over a network for the purpose of military training in small groups [12]. SIMNET, which is a distributed system for real-time simulation of battle engagements, provides soldiers with a virtual environment in which they can experience dangerous war scenarios through simulation training. The Distributed Interactive Simulation (DIS) derived from SIMNET can execute a low-cost distributed simulation, though it enables more simulators to take part in virtual environment [13]. However, in spite of its wide packet definition set, the DIS was not popular enough to be applied to all kinds of simulators.

The Defense Modeling and Simulation Office (DMSO) of the US DoD proposed HLA, which enables interoperability between many simulators. In HLA, the weak points of a distributed interactive simulation are removed, enabling the simulation components to be reused [14]. HLA has an object-based simulation software structure, the IEEE 1516 standard, for enhancing effectiveness in distributed simulation developments. Conceptual definitions are used in HLA to make individual simulation components interoperable and reusable [15]. HLA is substantially compatible with various simulators in broad distributed environments. In addition, HLA-based simulations have the potential to be easily integrated with simulators of the future, particularly because of a framework that enables reusability in existing component modules. Under the policy of the US DoD all military simulation systems developed since 2001 are based on HLA. Research is currently underway to change application programs based on SIMNET or a DIS into a HLA structure [15]. We used HLA to develop a simulator that can be applied to a distributed network environment.

3. Design of the Multichannel Display and the Motion Chair

3.1 System Overview

To develop an underwater vehicle simulator, we introduced a HLA framework for real-time integration with simulation components in distributed network environments. The simulator, as shown in Figure 1, consists of three federates (a system control module, a motion generation module and a visual simulation module). The federates are integrated with the run-time infrastructure (RTI), which consists of middleware and the IEEE 1516 standard. When the federation, namely the integrated simulation system comprising the three federates and the RTI, is operating, the RTI controls the message exchange for synchronization between individual federates. Each federate controls its own simulation time so that it can exchange information with other federates. The integrated simulation system includes a cluster of seven PCs, each with an independent IP address.

In the scenario applied to the simulation system, the submarine approaches an enemy ship. When the submarine is within firing range it fires a torpedo. If the torpedo explodes, the screen trembles and the motion chair vibrates. Figure 2 shows the data flow of the operating process for this scenario.

3.2 Federation Structure for a Distributed Simulation

3.2.1 Federate 1: System Control Module

The system control module controls all the systems, including the motion chair module and the multichannel display module. A user can use a joystick to change the location of the submarine in real time. In this way, the system control module receives commands from the user, modifies the current location and orientation of the submarine, and transfers those coordinates and motion events to the multichannel display module and to the motion chair module, respectively.

As shown in Figure 2, a user can manipulate the joystick in response to the motion and visual cues from the motion chair module and the multichannel display module. The input values from the joystick are the acceleration, the steering angle, and the event of firing torpedoes. Details of the acceleration and steering angle are fed back to a dynamics model of the submarine, where they are used to calculate the location and orientation of the submarine. The dynamics model used in this study is based on simple Newtonian mechanics of rigid body without any fluid mechanical aspects in order to make the

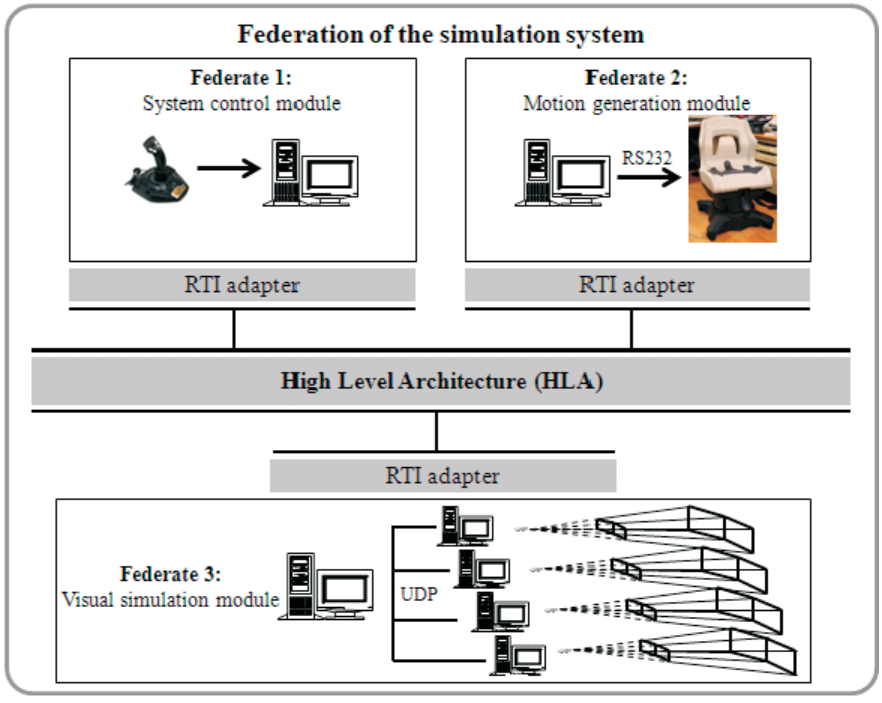


Figure 1. Federation of the integrated simulation system

model as simple as possible. This seems plausible to the purpose of this study where we assumed that the movement of the submarine is mild, all flows of water are laminar, and there are no effects of wave and abrupt changes of current. Therefore, the dynamics model in this study is simply double integrating acceleration in order to compute next position of the submarine in the water. Details of events, such as the firing of a torpedo, are used as motion commands through the event handler. Updated values on the location and orientation of the submarine are transferred to the multichannel display module and the motion chair module through the HLA/RTI.

3.2.2 Federate 2: Motion Generation Module

As shown in Figure 2(b), the motion chair (or, generation) module uses data on the acceleration, gradient and vibration events to generate a sense of motion. These values are calculated in the system control module and the moving motion chair. The motion chair module calculates the linear acceleration and centrifugal force and changes them so that they can be applied to the chair through a washout filter. Moreover, the sense of motion is increased by the motion superposition of data on the acceleration, orientation, and vibration for events [22].

We used a Joychair that provides a two degrees of freedom (DOF) motion sense with roll and pitch. The motion chair operates on an RS232C serial communication protocol with a baud rate of 4,800 bps. To tilt the motion chair to the front from its current location with maximum speed, we need to ensure that the data packet 40 4D FE 7F 0A 00 00 14 is transferred to the network. The last number, 14, is used as a check sum to check if the signal is correctly transferred. The sum of the first seven bytes is 214 in hexadecimal numbers, and the last two digits are taken:

40 4D YY XX VY VX 00 PP

YY: moves back and forth in the range between 00 and FE; the median value is 7F.
XX: moves right and left in the range between 00 and FE; the median value is 7F.
VY: represents the speed of the back-and-forth movement in the range between 00 and 0A, where 0A is the highest speed.
VX: represents the speed of the right-and-left movement in the range between 00 and 0A, where 0A is the highest speed.
PP: represents the sum of 7 bytes (check sum); a check sum is a decimal number used for checking if the signal is correctly entered.

Based on Newton's second law, $f = ma$, the sense of motion can be expressed as linear acceleration, where m is the mass of a person, and a is acceleration. When making circular motion, a person is subject to centrifugal force, $f = mv^2/r = mr\omega$, which creates the feeling as if the

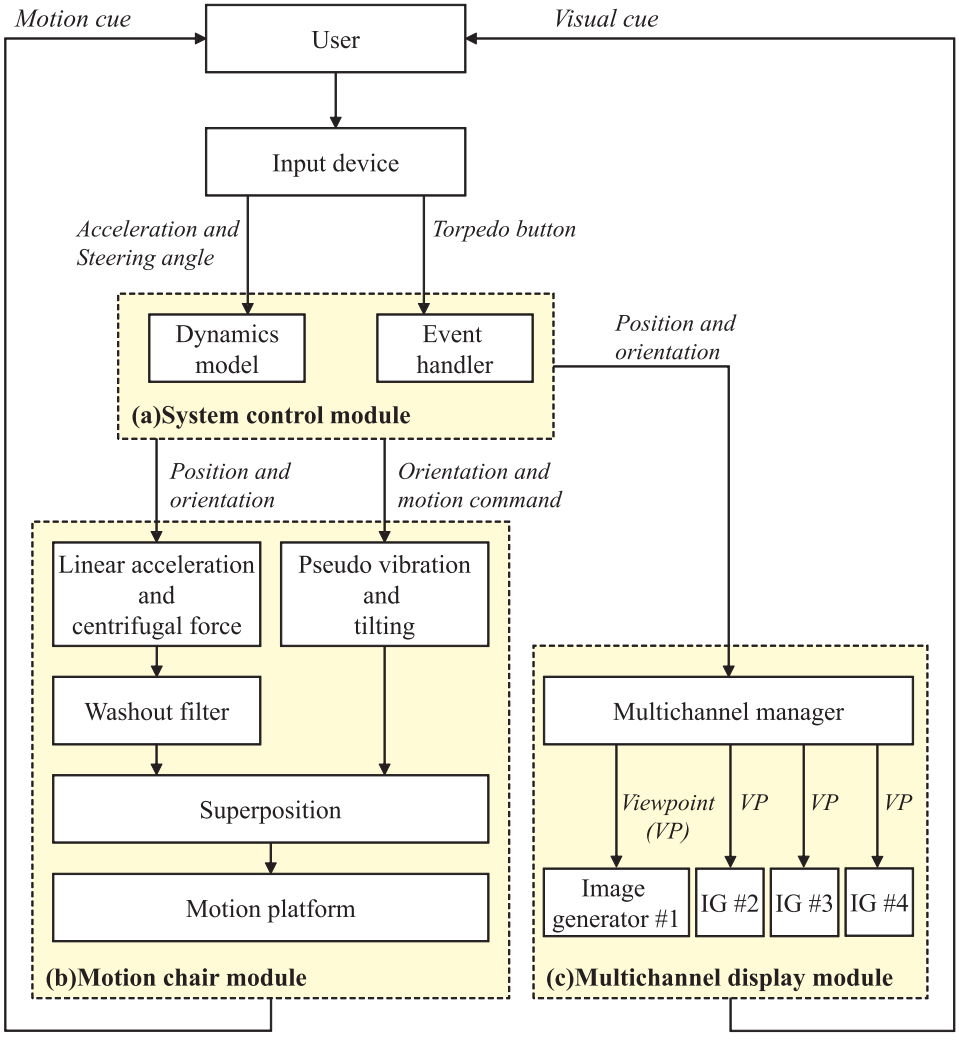


Figure 2. Work process of the integrated simulator

body tends to move outwards, where m is the mass of a passenger, v is the velocity of a submarine, r is radius of curvature, and ω angular velocity. Therefore, in most simulators, in order to make a simulation more realistic, linear accelerations and angular velocities are applied to the user by moving the motion platform on which the mock-up vehicle is located. This has to be accomplished within the workspace of the simulator, which is limited in a small room. The software that is in charge of this is called washout filter. Since a two DOF Joychair (a small-scale dynamic motion platform) cannot physically express the motion of a sway, heave, surge or yaw but only a roll and pitch, the sense of motion expressed by linear accelerations and angular velocities may be significantly reduced. In order to compensate for these decreased realism, we designed a simple washout filter which can express linear accelerations and angular velocities by adjusting roll-and-pitch angles. When linear accelerations increase, the pitch angle decreases to the negative value, and vice versa. When angular velocities increase in a counterclockwise rotation, the roll angle increases to the right, and vice versa.

As shown in Figure 2(b), the vibration effect and the inclination of a submarine can be directly controlled by motion commands. Vibration is usually generated using heave motion. However, when there is no heave motion available, the motion chair can produce a similar effect with a simultaneous random roll-and-pitch motion.

3.2.3 Federate 3: Visual Simulation Module

The visual simulation (or, multichannel display) module receives the value of the location and orientation of the

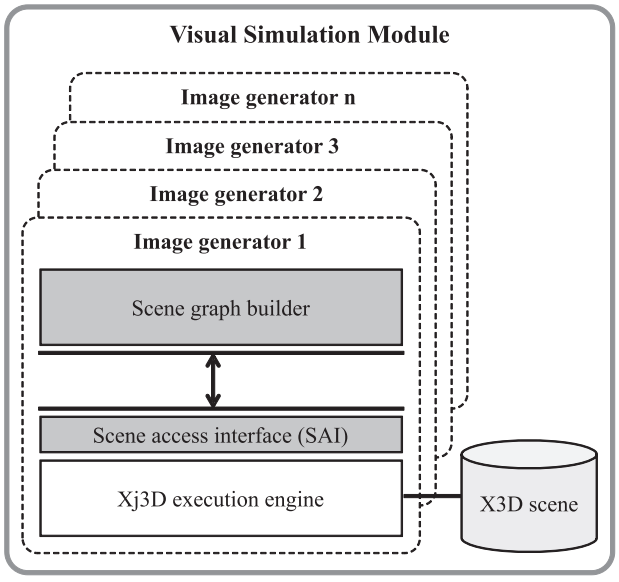


Figure 3. Process of dynamically changing an X3D scene-graph with scene access interface

submarine, which is calculated by the system control module, and then visualizes scenes on the multichannel screen. As shown in Figure 2(c), the signals of the location and orientation of the submarine are transferred to image generators composed of four independent PCs with information from each viewpoint. To implement multichannel visualization for dynamic scenes, we need to apply updated information on the position and orientation of the submarine to the X3D scene-graph for each time frame. However, the X3D format is most likely a container that stores static 3D geometric information. To dynamically change the X3D scene-graph, we used a Scene Access Interface (SAI), which is similar to the External Authoring Interface (EAI) of VRML. The SAI is a kind of application programming interface that was proposed by the Web3D consortium to dynamically modify X3D scene-graphs [16, 17]. To handle the X3D data, we used the SAI for Xj3D visualization. As shown in Figure 3, the X3D viewer that receives the value of the location and orientation of the submarine uses the scene-graph builder to transform the scene data in the X3D file into a scene-graph. Finally, the Xj3D engine is used to visualize the X3D scenes.

3.3 Synchronization of the Multichannel Display

To increase the user's immersiveness in a virtual environment, we used a multichannel display system on several ordinary PCs to support seamless mapping of large images across multiple screens. The cluster of PCs simultaneously synchronizes all the visualization programs. Figure 4 shows the viewing frustum when the visualization area in a single channel for a submarine scene is changed to two channels. To express a single scene with two screens, we need to set up the viewpoint by rotating two viewing frustums at a pre-calculated angle. Details of how to implement multichannel display are described in the appendix.

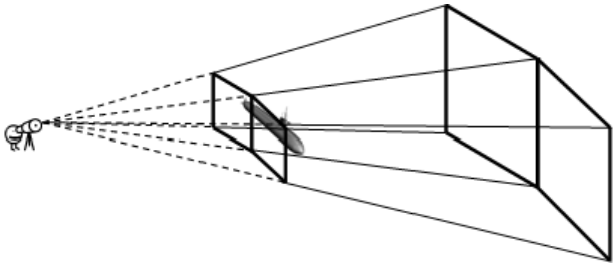


Figure 4. Concept of synchronization in a double-channel display

3.4 Message Exchange Mechanism among Federates

If the construction of a federate is completed for a distributed simulation, the federation can be executed. In this case, if the federates form a federation, the messages to be exchanged between the federates should be defined. The Federation Object Model (FOM) of the HLA used in the message exchange mechanism hierarchically defines the data structure of the information to be shared among the federates when a distributed simulation is operating. For instance, if a federation consists of a submarine federate and a torpedo federate, the federates should exchange information about the location of the objects between them. The FOM contains information about the submarine object, the torpedo object and the location of each object. Of the three federates mentioned in Section 3.2, the object to be included in the FOM is the object *SM_Info*, a system control module which contains information about the location and orientation of the submarine. The remaining two federates, the motion chair module and the multichannel display module, merely subscribe to information from an instance of *SM_Info*. Thus, no information can be shared between the motion chair module and the multichannel display module.

Figure 5(a) shows a Unified Modeling Language (UML) diagram of the FOM's object classes, as well as their attributes and inheritance. As all object classes of the HLA are inherited from the *ObjectRoot* class, they have an attribute of *privilegeToDeleteObject*. Federates with all of these inherited attributes can get rid of object instances. The motion chair object and the multichannel display module object have no individual attributes. Figure 5(b) shows the structure of the interaction class in the FOM. The *SimulationEnds* class is used to order the sim-

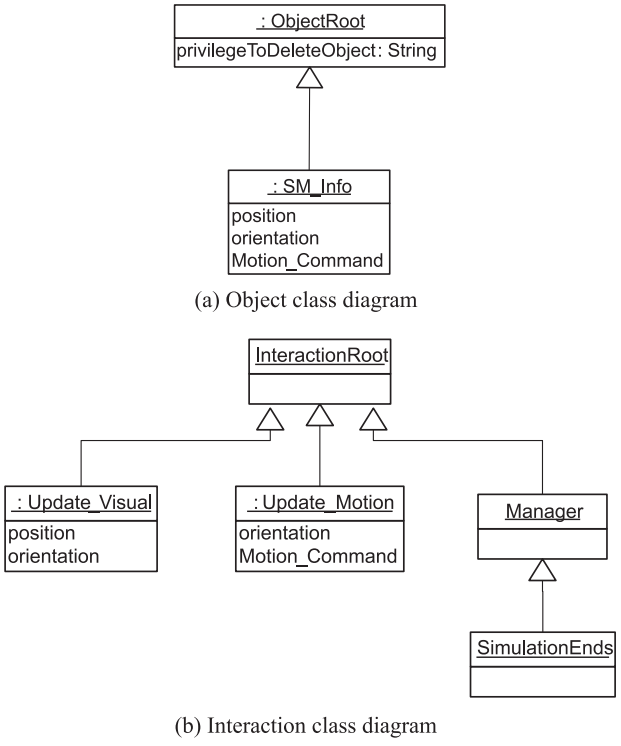


Figure 5. A unified modeling language diagram of objects and interactions [21]

ulation to be finished. The manager class, which the *SimulationEnds* class inherited, is a class that the HLA basically provides for managing the simulation. If the system control module publishes an interaction named *SimulationEnds*, the motion chair module and the multichannel display module finish their operation upon receipt of the interaction.

If the federation is formed, all the federates that belong to the federation set up several object attributes. The federate of system control publishes attributes such as the location, orientation, and motion events of the submarine object. The federates of the motion chair module and the multichannel display module then subscribe attributes to these attributes. In this case, the relation of publication and subscription is stored in the RTI. The message exchange procedure between the three federates that make up the federation is as follows. After the system control module uses user input to update the attributes of the submarine object, namely the location, orientation and event motion, these attributes are sent to the RTI by the RTI command *Update_Attribute_Values*. The RTI searches for the relation of publication and subscription and uses the RTI command *Reflect_Attribute_Values*+ to send the attribute values to whichever federate that wants to receive them. In this case, the + sign refers to a callback function directly called by the RTI.

Table 1. Environment for the implementation

Software	Operating system	Windows XP SP2
	HLA/RTI	MAK RTI (C++)
	X3D SDK	Xj3-D Toolkit (Java)
	Network protocol	Viewer side: UDP, Federates: RTI
Hardware	VGA card	Nvidia GeForce 6600
	CPU	Intel P4, 3.0 GHz
	RAM	1 GB
	Input device	Microsoft SideWinder Precision 2 Joystick
	Virtual environment	8 channel iCAVE (Fig. 9)
	Motion chair	2 DOF Joychair (Fig. 3)

4. Implementation and Experiment

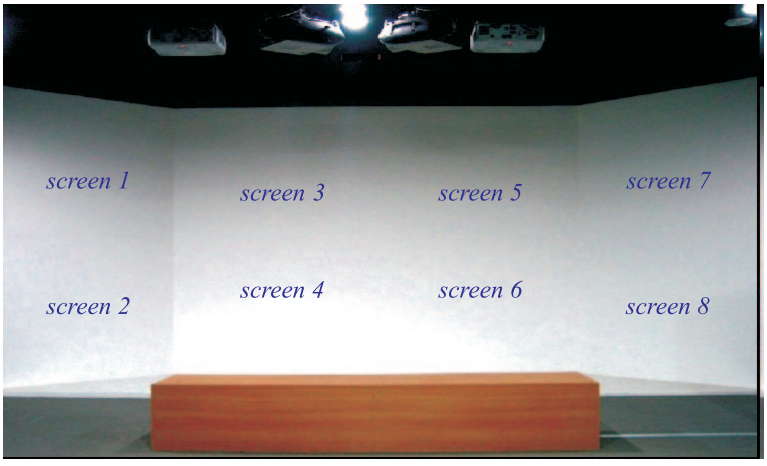
4.1 The iCAVE System

The underwater vehicle simulator that uses X3D and the motion chair in the multichannel display room, as shown in Figure 6, were implemented in iCAVE, which is a multichannel VR system developed at the KAIST [17, 18]. The iCAVE system consists of eight screens, each 1,900 mm by 1,400 mm, and eight front projectors. For underwater vehicle simulation, we installed seven PCs: two sets for the system control module and the motion chair module and five sets for the multichannel display module. The five PCs for the multichannel display module consist of four slave PCs, which display X3D models in their corresponding screens, and one master PC, which synchronizes each X3D scene from the slave PCs to generate a single large scene. The graphic card in the slave PCs is connected to two sets of the front projector. The master PC sends synchronization packets to each slave PC through a LAN with a 100 Mbps ethernet. Each slave PC has a resolution of 1,024 by 1,536 and the entire screen has a resolution of 4,096 by 1,536. The User Datagram Protocol (UDP) was used as a network protocol to connect the four slave PCs and the master PC.

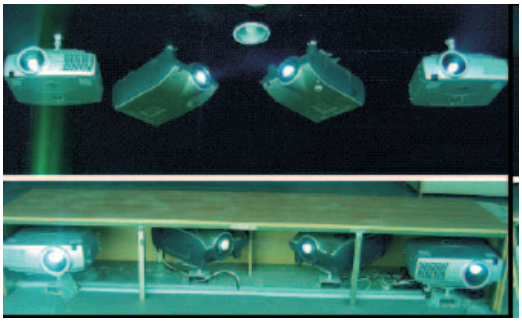
4.2 Implementation Environment

X3D-based multichannel visualization was applied to an underwater vehicle simulator and then integrated with the two DOF Joychair to improve the sense of motion. An RTI from MÄK, a company of VT Systems Inc., was used to integrate the HLA framework. Table 1 shows the implementation environment.

Figure 7 shows an underwater vehicle simulator with the multichannel visualization. Since the multichannel display module enables several viewpoints to be displayed at the same time, the submarine can be seen from various



(a) Eight screens with 1900 mm x 2800 mm for each



(b) Eight front projection type projectors

(c) Joychair

Figure 6. The virtual environment of iCAVE

angles. In addition, the motion chair can be used to indicate the linear acceleration, the centrifugal force, the inclination of the submarine and the vibration after firing a torpedo.

4.3 Experiment

The scenario we applied to the simulator is as follows:

- a submarine is far away from an enemy submarine, which is at Busan port located in South Korea
- after the submarine identifies the location of the enemy ship, it approaches the enemy ship until it is within the firing range of a torpedo
- a torpedo is fired at the target
- when the torpedo collides with the enemy ship, the motion chair vibrates and the screen trembles.

The experiment involving this scenario was posted on YouTube (www.youtube.com/watch?v=PUbVsbVcuSE). Our results were compared with the results of two other simulators: namely Vega of Presagis, a high-end commercial visualization tool; and the Deep Exploration CAD Edition of Right Hemisphere, a middle and low-end commercial visualization tool. The comparison focused on four items: the frame rate, the file size and loading time, the openness to platform, and the image quality.

4.3.1 Frame Rate

Table 2 compares the frame rate at the four arbitrarily assigned viewpoints of the camera location against five 3D geometries (the inland terrain near Busan, the underwater terrain near Busan, the submarine, the torpedo and the sea near Busan). The comparative results of the 3D geometries, as represented by Xj3D and Deep Exploration, reveal that all the viewpoints except for the first have similar performances. The reason Xj3D has a higher frame rate than Deep Exploration at viewpoint 1, where a smaller number of polygons are visualized by the occlusion culling method, than at the other viewpoints (viewpoints 2, 3 and 4) is that Xj3D is designed to focus on the

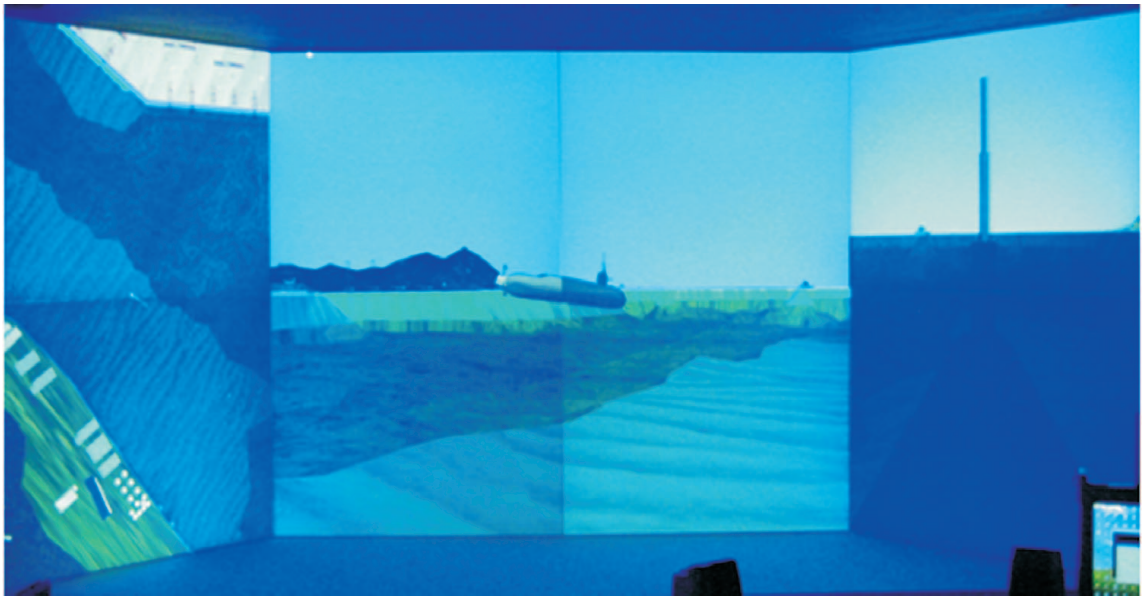


Figure 7. Results of integration of the X3D multichannel display, the motion platform and the high level architecture and run time infrastructure

Table 2. Comparison of experimental results

		Xj3D (X3D)	Vega (Open Flight)	Deep Exploration (Open Flight)
Frames per second	Viewpoint 1	30 fps	18 fps	11 fps
	Viewpoint 2	10 fps	12 fps	10 fps
	Viewpoint 3	8 fps	24 fps	8 fps
	Viewpoint 4	10 fps	24 fps	9 fps
Loading time		30 s	80 s	20 s
Openness	Cost	Free	More than $20,000	$1,495
	Platform	Platform independent	Win, Linux, SGI Irix (All versions for each operating system should be purchased)	Windows
Image quality		Objects at a distance are frequently flickering due to low precision (16 bit) of the z-buffer. However, the problem can be solved by separating boundary surfaces to some extent.		Objects at a distance flicker less frequently due to the relatively high precision (24 bits or more) of the z-buffer. Due to anti-aliasing, scenes are rendered with high quality at the cost of performance.

basic functionalities of shading or rendering. Moreover, Deep Exploration has a large number of calculations due to the anti-aliasing involved in making the image quality smooth.

The frame rate of Xj3D in viewpoints 2, 3 and 4, all of which have many polygons, is similar to that of Deep Exploration. Vega has a higher frame rate than Xj3D because the Open Flight model used by Vega uses a certain level of detail. Because the X3D model in our scenario did not use that level of detail, it rendered all geometries, some of which are at a long distance and are not necessary to be rendered as they are almost invisible. As a result, there is an increase in the number of polygons to be visualized per frame.

4.3.2 File Size and Loading Time

As shown in Table 2, the loading times are 30 seconds for Xj3D, 80 seconds for Vega and 20 seconds for Deep Exploration. In general, the loading time is correlated with the file size. As shown in Table 3, when the same 3D geometry is compared with the three file formats, X3D has smaller files than those of VRML or Open Flight of

Table 3. Comparison of file size (unit: kilobyte)

3D models	Xj3D(X3D)		VRML	Open Flight
	ASCII	Binary	ASCII	Binary
Busan terrain	14,887	1,347	27,395	12,598
Busan underwater terrain	939	144	2,602	1,222
Submarine	386	64	1,116	407
Torpedo	25	6	73	N/A
Busan sea surface	22	5	35	83

Deep Exploration. Because X3D uses a different and better encoding method than VRML, the file structure is optimized more in X3D than in VRML. However, it is hard to say whether X3D has a more optimized file structure than Open Flight because Open Flight has the information that is more internal and non-geometric than that of X3D, though the binary-type file size of X3D is smaller than that of Open Flight. Because Deep Exploration uses an internal optimized loading algorithm only for the static rendering without simulation, it can load the same 3D geometry of Open Flight more quickly than Vega.

4.3.3 Openness to Platform

X3D is the ISO standard for real-time 3D computer graphics. Visualization application programs, such as the Xj3D viewer, can be used without any additional fee (as shown in Table 2). While X3D can be used regardless of the platform, Vega is dependent on different platforms, such as Windows, Linux and SGI Irix. Vega is therefore difficult to exchange with different platforms. Furthermore, each platform involves additional costs. Deep Exploration is available only for a Windows environment.

4.3.4 Image Quality

The results of image quality, as shown in Table 2, indicate that the visualization image in Xj3D and Vega flickers in certain scenes. The flickering mainly occurred in scenes where the sea and land crossed. Additionally, this phenomenon occurs when the scene is viewed from a long distance rather than from close range. Because the depth difference between the water and land is small, the overall visualization scale becomes larger when the scene is viewed from a long distance; consequently, the height difference of the land and water becomes smaller. When this height difference becomes so small that the precision level for the depth difference is in the z-buffer, the graphics hardware misperceives the fact that the surfaces of the land and sea are located at the same height; as a result, the two surfaces appear separately. This phenomenon, known as z-fighting, creates an artifact.

Less flickering occurred in Deep Exploration than in Xj3D or Vega because Deep Exploration uses a higher level of precision (24 bits or more) in the z-buffer than either Xj3D or Vega (16 bits). Solving the flickering problem is difficult because the Xj3D developers would need to modify the source code of Xj3D. An alternative way of decreasing the flickering is to make either the land level higher or the sea level lower so that difference between the sea level and the land level becomes larger. In fact, when the sea level is lowered, the flickering disappears. Vega's Marine module has a module named Vega Marine. By using this module, we can express a rising and falling wave more realistically. Furthermore, by adjusting the transparency of the sea water and applying a fog effect to the land terrain, underwater scenes and the sea, we can create a similar environment to the real world.

5. Conclusion

A conventional simulator uses costly commercial visualization tools and dedicated hardware, but it has problems such as specific space for managing the system, as well as problems with maintenance and accessibility due to location and time restraints. The use of a single-channel visualization method also diminishes the sense of immersiveness. Furthermore, conventional simulators are difficult to integrate with others made for different environments. To solve these problems, we developed a form of submarine simulation based on the platform-independent ISO standard X3D and a cluster of ordinary PCs. To increase immersiveness, we applied multichannel visualization and a motion platform to an underwater vehicle simulator. The HLA/RTI is used to effectively integrate federates, such as a motion chair, a joystick, and a multichannel display. To experiment with the underwater vehicle simulation system, we constructed a virtual environment similar to the Busan port, South Korea. The results confirm that the performance of our underwater vehicle simulation system is comparable to that of a high-end commercial visualization tool.

We succeeded in implementing a simple distributed simulation environment. In the future, we intend to address the performance of the HLA and more realistic problems of large-scale systems in a virtual war scenario involving several submarines, airplanes and battleships. Lastly, we used the commercial MÄK RTI to construct a distributed simulation environment. Recently, the extensible modeling and simulation framework has proposed the use of Web-enabled RTI for Web technology [19, 20]. Moreover, a General Public License (GPL) based on the RTI is currently being negotiated. Hence, in the future, it will be possible to implement a system with a low-cost framework for a distributed simulation environment by using the Web-enabled RTI or an RTI based on a GPL instead of the commercial RTI implementation.

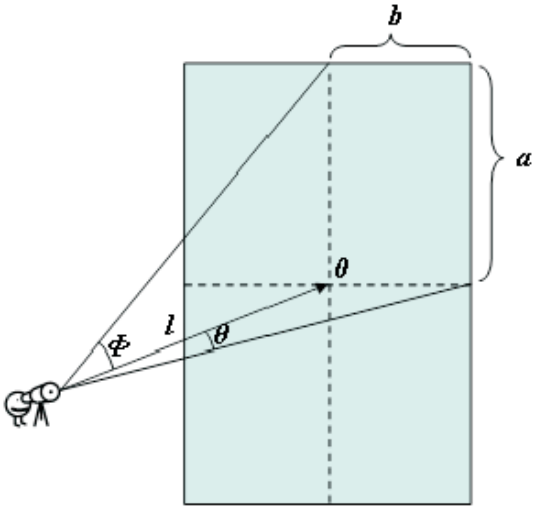


Figure 8. Variables for synchronization in a double-channel

6. Appendix

Synchronization of multi-channel display

To explain how to synchronize multi-channel display, it would be better to take a simpler example of double-channel display.

The aspect ratio for a single screen is the ratio of the horizontal to the vertical length of the screen. Using the variables defined in Figure 8, we can calculate the rotation angle of the viewing frustum as follows:

$$\begin{aligned} \textit{Aspect ratio} &= \frac{\textit{resolution width}}{\textit{resolution height}} \\ &= \frac{a}{b} = \frac{2l \times \tan\theta}{2l \times \tan\phi} = \frac{\tan\theta}{\tan\phi}. \end{aligned} \quad (1)$$

If the horizontal and vertical resolution of the PC is already set up, and the horizontal and vertical length of the screen is known, the aspect ratio can be calculated. In other words, the relation between and can be computed. If the distance between the user and the screen is , we can use Equation (1) to calculate both and as follows:

$$\begin{aligned} a &= 2l \times \tan\theta \\ \therefore\ \theta &= \tan^{-1}\frac{a}{2l} \\ \therefore\ \phi &= \tan^{-1}\frac{\tan\theta}{\textit{Aspect ratio}}. \end{aligned} \quad (2)$$

Thus, as expressed in the Equation (3), the field of view is double the smaller value of and as calculated by Equations (1) and (2):

$$\textit{Field of view} = 2 \times \min(\theta, \phi). \quad (3)$$

When the field of view has been calculated and applied to the multichannel visualization system, we can rotate each viewpoint by angle clockwise and counterclockwise, respectively, to display a single scene on two screens.

To synchronize the channels in a cluster of PCs, we used a method that transfers synchronized packets to each PC on the network. Because each channel of the multi-channel display module is physically close to each other, Transmission Control Protocol (TCP) or UDP can be used instead of the RTI. We used the UDP in order to avoid the system latency of TCP when there are many channels to be synchronized or many packets to be transferred.

7. Acknowledgement

This work was partially supported by Ajou University, Korea, through project no. 20083830.

8. References

[1] Brutzman, D., M. Zyda, M. Pullen, and K. Morse. 2003. XMSF overview, progress examples and exercise planning. In *Proceedings of the XMSF JFCOM Workshop*, Suffolk, Virginia, USA.

[2] Christianson, B. 2000. *Comparison of Vega and Java3D in a virtual environment enclosure*. Master Thesis, Naval Postgraduate School.

[3] Salisbury, C., and S. Farr. 1999. Web-based simulation visualization using Java3D. In *Proceedings of the 31st Conference on Winter Simulation*, Phoenix, Arizona, USA, pp. 1425–1429.

[4] Li, Y., K. Brodlie, and N. Phillips. 2000. Web-based VR training simulator for percutaneous rhizotomy. *Studies in Health Technology and Informatics* 70, 75–181.

[5] Robert, J., and R. Knight. 2000. Multiple window visualization on the Web using VRML and the EAI. In *Proceedings of the 7th UK VR-SIG Conference*, pp. 149–157.

[6] Moon, H.I., and S. Han. 2005. Mapping digital manufacturing simulation to synthetic environment using SEDRIS (in Korean). *Journal of the Korean Society for Simulation* 14: 15–24.

[7] Blais, C., D. Brutzman, D. Horner, and S. Nicklaus. 2001. Web-based 3D technology for scenario authoring and visualization: the SAVAGE project. In *Proceedings of the Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC)*, Orlando, Florida, USA.

[8] Blais, C., D. Brutzman, J. Weekley, and L.T.J. Harney. 2002. Emerging Web-based 3D graphics for education and experimentation. In *Proceedings of the Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC)*, Orlando, Florida, USA.

[9] Sims, E. 2007. Reusable, lifelike virtual humans for mentoring and role-playing. *Computers and Education* 49: 75–92.

[10] Lee, D. 2003. Extensible Interactions in X3D (in Korean). *Journal of the Korea Information Science Society* 30: 349–351.

[11] Lee, S. 2002. A study on the 3D humanoid modeling animation technology using X3D (in Korean). In *Proceedings of the Korean Institute of Maritime Information and Communication Sciences Conference*, pp. 812–816.

[12] Singhal, S., and M. Zyda. 1999. *Networked virtual environments: design and implementation*. Addison-Wesley Press, New York.

[13] Judith, D., and K. Frederick. 1999. *Creating computer simulation systems – an introduction to the high level architecture*. Prentice Hall, New York.

[14] Ping, I. 2000. *High level architecture performance measurement*, Master Thesis, Naval Postgraduate School.

[15] Defense Modeling and Simulation Office: High Level Architecture, www.dmso.mil/public/transition/hla/
[16] The Xj3D Project, www.xj3D.org
[17] Kim, Y., J. Yang, and S. Han. 2006. A multichannel visualization module for virtual manufacturing. *Computers in Industry* 57: 653–662.
[18] Hur, P. 2006. *HLA-based integration of underwater vehicle simulations using X3D multi-channel visualization and a motion platform (in Korean)*, Master Thesis, KAIST.
[19] The Web3D Consortium: X3D Specification, www.Web3D.org/specifications/X3D-3.0.dtd
[20] Extensible Modeling and Simulation Framework (XMSF), www.movesinstitute.org/xmsf/xmsf.html
[21] Hur, P., J. Yang, and S. Han. 2008. A underwater vehicle simulator using X3D and a motion chair in a multi-channel display room (in Korean). *Transactions of the Society of CAD/CAM Engineers* 13(1): 45–57.
[22] Yoo, B., M. Cha, and S. Han. 2005. A framework for a multi-sensory VR effect system with motional display. In *Proceedings of 2005 International Conference on Cyberworlds*, Singapore, pp. 237–244.

Pilwon Hur *is a PhD candidate in the Mechanical Science and Engineering department at University of Illinois, Urbana-Champaign. He joined human dynamics and control lab (HDCL) in August, 2006. Currently, he is working on the stability and robustness of human postural control system with both deterministic and stochastic approaches. His special interest is in the effect of age and vision on the stability of human balance. Before he joined HDCL, he was in the virtual reality (VR) group in the pursuit of master's degree at KAIST. His research interests are stochastic modeling of human postural control system, and application of VR and mixed reality (MR) to human postural control, especially, for rehabilitation.*

Jeongsam Yang *is an assistant professor in the Department of Industrial & Information Systems Engineering and is leading the CAD laboratory (http://cadlab.kaist.ac.kr) at Ajou University. He worked at Clausthal University of Technology (Germany) as a visiting researcher and the University of Wisconsin-Madison (USA) as a postdoctoral associate. He obtained his PhD in mechanical engineering in 2004 at KAIST. His current research interests are product data quality (PDQ), VR application in product design, product data management (PDM), knowledge-based design system, and STEP.*

Soonhung Han *is a professor at the Department of Mechanical Engineering (http://me.kaist.ac.kr) of KAIST (Korea Advanced Institute of Science and Technology, www.kaist.ac.kr). He is leading the Intelligent CAD laboratory (http://icad.kaist.ac.kr) at KAIST, and the STEP community of Korea (www.kstep.or.kr). His research interests include STEP (ISO standard for the exchange of product model data), VR (virtual reality) for engineering design, and knowledge-based design system. His domain of interests include automotive and shipbuilding. He has a BS and a MS from the Seoul National University of Korea, another MS from the University of Newcastle-upon-Tyne in the UK, and a PhD from the University of Michigan in the USA. He is involved in the professional societies of CAD/CAM (www.cadcam.or.kr) and e-Business (www.calsec.or.kr). He is an editorial member of the web-based journal, International Journal of CAD/CAM (www.ijcc.org).*

Byounghyun Yoo *is a post-doctoral research fellow of Modeling, Virtual Environments and Simulation (MOVES) Institute in Naval Postgraduate School, Monterey California. Before he joined the research group of MOVES, he had been a research fellow at the Department of Mechanical Engineering in KAIST. His research interests include modeling and simulation based on geospatial information, modeling and rendering of large scale virtual environment, fusion of virtual reality and geographic information system, multi-sensory VR system, and VR system integration.*