NAVER : PC 클러스터 기반의 분산가상환경 커널 설계 및 구현

박창훈[ab], 고희동[a], 조창석[a], 안희갑[a], 한요섭[a], 김태윤[b]
[a]한국과학기술연구원,
[b]고려대학교

NAVER : Design and Implementation of Networked Virtual Environments Based on PC Cluster

ChangHoon Park[ab], HeeDong Ko[a], Changseok Cho[a], Hee-Kap Ahn[a],
Yo-Sub Han[a] and TaiYun Kim[b]
[a]Korea Institute of Science and Technology(KIST),
[b]Korea University

Abstract

The NAVER is based on a cluster of low-cost personal computers. The goal of NAVER is to provide flexible, extensible, scalable and re-configurable framework for the diverse virtual environments especially for Gamsung research experiments. Personal computers are divided into three servers are according to their specific functions: Render Server, Device Server and Control Server. While Device Server contains external modules requiring event-based communication for the integration, Control Server contains external modules requiring synchronous communication every frame. And, the Render Server consists of 5 managers: Scenario Manager, Event Manger, Command Manager, Interaction Manager and Sync Manager. In this paper, we discuss NAVER as effective distributed system and its application to Gamsung experiment.

Keywords: Virtual Reality, PC-cluster, Re-configurable, Extensible

1. Introduction

Virtual Reality (VR) has recently become a familiar and ubiquitous name for the area of computing which takes the user into a three dimensional virtual space. It is a

popular medium for a number of application areas such as military, education, medicine, and entertainment industry [1].

Many VR applications will migrate from workstations to personal computers (PCs). To satisfy the real time interaction, system must maintain a high visual frame rate and maximize the responsiveness to user inputs. To generate a virtual space, special-purpose visual workstations have been used. With the recent increases in computation power graphics performance of PCs, they are becoming a viable alternative to expensive workstations for VR applications [2].

In this paper, we present a new framework named NAVER based on a PC cluster. Our framework aims to facilitate the development of VR applications. In the NAVER, component nodes are classified into three categories according to its main function: Render Server, Device Server and Control Server. Render Server provides real-time 3D graphics rendering. Device Server manages devices or applications which are tightly coupled with a virtual space generated by Render Server. In Control Server, devices or applications that are loosely coupled with a virtual space are handled. NAVER is designed to provide flexible, extensible, scalable re-configurable framework for VR applications. The following are considered:

First, the structure of NAVER is extensible and reconfigurable by introducing the concept of external modules. External modules represent various devices or applications that interact with NAVER kernel.

In the NAVER, external modules are divided into two groups by the interaction characteristic with the virtual space: one interaction characteristic requires non-periodic event communication, the other requires the streaming communication like navigation or manipulation control input. For the efficient management of them, the former external module are included in the Control Server, while the latter external modules are handled by the Device Server.

Second, the NAVER enables to specify the configuration of the system and the virtual space by s script file. The script file describes the object and structure of the 3D virtual environment.

In this paper, a new framework based on low-cost PCs cluster is proposed with supporting the function of high-end visual workstation. And, this framework enables the specification and operation of virtual environment which is generated by integrating a 3D virtual space with external modules on different hosts.

2. Related Work

This section introduces a number of current virtual reality systems, and outlines their basic design: VR Juggler from Iowa State University [4], and DIVERSE from Virginia Tech University [5].

The VR Juggler project's goal is to maximize flexibility, portability, extensibility, and maintainability of both applications and the library itself. The VR Juggler kernel is micro-kernel architecture. The kernel provides a mechanism for simpler, initial development of applications that can be added, reconfigured and/or removed during run-time. In addition, VR Juggler provides a "virtual platform" for application development. This virtual platform concept extends throughout

the VR Juggler design, including areas like graphics and input device handling. VR Juggler supports several other kinds of input data. Each input type has an associated proxy interface that can be accessed by the application. Similarly, the application is protected from dealing with the complexities of displays.

DIVERSE is a modular collection of complementary software packages that were developed to .facilitate the creating of distributed operator-in-loop simulations. DIVERSE consists of two packages. The DIVERSE Toolkit (DTK) provides an access to local and networked interaction devices, both real and simulated. It also provides support for run-time swapping of I/O devices and/or emulators, enabling the creating of device independent application. The DIVERSE graphics interface for Performer (dgiPf) adds immersive and/or non-immersive graphics to simulations by augmenting OpenGL Performer, a high-level scene graph-based graphic API.

The purpose of NAVER is very similar to that of the previous system. NAVER, however, is designed as a distributed system for extensibility, and has its own scripting language for people who are not expert in programming to develop various contents.

3. NAVER

In this section, we describe the main components of NAVER kernel: Scenario Manager, Command Manager, Event Manager, Interaction Manager, and Sync Manager.

The kernel aims to facilitate the integration of three servers described in the pervious section as well as dynamic management of the virtual space.

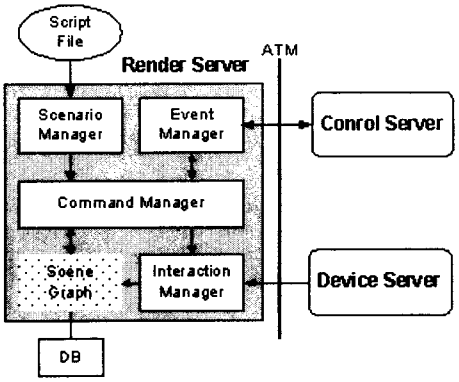

Figure 1. NAVER Kernel

3.1 Scenario Manager

The main function of the Scenario Manager is to interpret and process the script for presentation and interaction. The script enables the author to describe both static and dynamic of virtual environments. In this paper, virtual environments are defined by 3D virtual space and multi-modal interface of three servers.

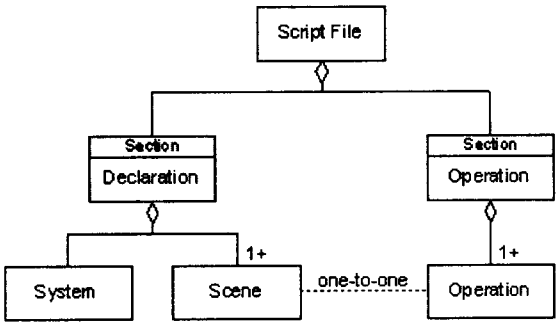

Figure 2. Structure of the script file

Figure 2 shows the structure of the script. The script consists of the following major functional components: declaration and operation. The declaration contains nodes which describe information to initialize virtual environments. There are two types of node named System and Scene. The node is the basic building block of the script. Nodes are

abstraction of various concepts and objects for physical system and 3D virtual world. In the operation, there is the Operation node to specify dynamic control of virtual environment.

A script contains one System node which specifies configuration of the Render Server, the Device Server and the Control Server. Information such as Internet address and port number is described to support communication among these servers. And, System node is a static node which cannot be modified at rum-time.

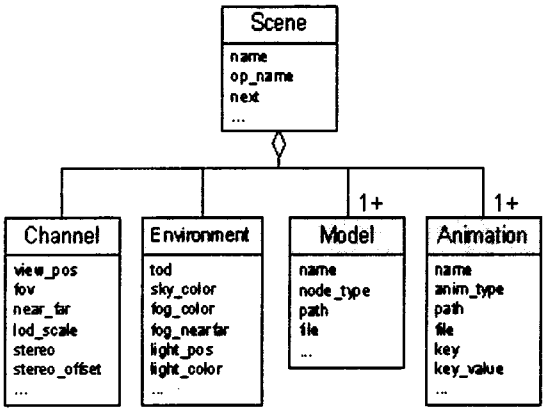

Figure 3. Scene Specification

The Scene node is used to specify a 3D virtual world. Scene is a grouping node which includes a list of nodes. Scene node includes Channel, Environment, Model and Animation which have their private function. Channel node has fields to describe a view of a scene like view position, viewing frustum, field-of-view and so on. Environment node provide a way to simulate atmospheric effects including time-of-day, the color of sky, light position and more. Model node specifies meta information about 3D model data to be included in the scene. And, animation node is designed for key framed animation. Therefore, virtual worlds can be specified by setting their values.

The Operation node can be used for controlling the Scene node as time pass. A script contains one or more scene nodes. So, a Scene node can be identified by its name filed and their sequential relation can be described by means of next field which specifies the name of next Scene node. And, each scene node needs one operation node. So, the number of scene nodes and the number of operation nodes must be same. The op_name field of Scene node specifies the name of corresponding Operation node.

Operation node has one or more Action node whose fields allow specifying the concept of time and contains one or more Command node. The Scenario Manager requests the Command Manager to process the command when the time specification of Action node is set to current time.

3.2 Command Manager

The Command Manager processes the command required by the Scenario Manager and the Event Manager. Table 1 describes the command which can be processed by the Command Manager.

Table 1. Command Classification

Set	target value data	[node].[field] String String
Send	target value	[node].[field] String
Control	device name activate target	String Boolean [node].[field]

First, the "Set" command is used to control the scene which represents a virtual world. By setting the field of node consisting of a scene, the virtual world can be updated.

The target argument means the destination. And, the data includes the list of values to be used in the requested duration, while the value contains a single value for the filed. For example, we can update the background color of a virtual space at time 10 by describing as following.

```
Action {
    at          10.0
    Set  {
        target  Channel.sky_color

        value   "0.2 0.2 1.0"
    }
}
```

Second, the "Send" command transmits an event through the Event Manger in order to control the device which is handled by the Control Server. And, the field value of any node can be delivered to the Control Server. For example, we can send an event defined to turn on the projector connected on the Control Server.

Third, the "Control" command activates or deactivates the device on the Device Server through the Interaction Manager and connects the user input to the field of any node every frame for the navigation or the manipulation. For example, following script allows the user navigate virtual worlds from time 30 using the joystick.

```
Action {
    at                  30.0
    Control  {
        device_name     "joystick"
        activate        on
        target          Channel.view_pos
    }
}
```

3.3 Event Manager

The Event Manager handles the transmission of events to and from the Control Server, which is based on asynchronous message passing. The Event Manager allows the Render Server sent an event in order to control the peripheral device managed by the Control Server. In addition, the Control Server is allowed to transmit an event including the command to the Render Server through the Event Manager. For example, the Render Server can turn on the projector of the control server. And, a user on the Control Server is allowed to control the light of a 3D virtual space through GUI component of the interface of applications.

In the Render Server, the Command Manager processes receiving events in the queue every frame. It should be noted that the amount of events is closely related with the real-time performance. The available network bandwidth and the size of message must be considered. In general, events occur less frequently than frame rate.

3.4 Interaction Manager

The Interaction Manager connects the user input from the Device Server to the scene graph which is an abstract data structure representing a virtual space in the Render Server. The Interaction Manager allows the real-time interaction such as navigation or manipulation in the virtual space. For example, a user on the Device Server can navigate a virtual space using a force-feedback joystick, which vibrates when collision occurs.

For the natural interaction, the Interaction Manager provides the synchronous communication with the Device Server by

means of polling which is the query of the device status. In the Render Server, Polling is transmitted to the Device Server every frame. At this time, information about collision or terrain following can be transmitted to the Control Server for display devices or complex dynamic calculation. Then the Device Server replies to the recent status of the device. Comparing with the Event Manager, a message exchange occurs continuously for the interaction.

3.5 Sync Manager

The Sync Manager provides synchronous multiple channels by means of a cluster of PCs. Multiple channels are needed for graphics displays which requires more than one image such as Head-Mounted-Display, CAVE and so on. By introducing software-based real-time Inter Process Communication (IPC) technique into multiple Render Servers of a cluster, the Sync Manager makes synchronous multiple channels just as if these channels are managed by a single computer. The Sync Manager is divided into Sync Master and Sync Slave.

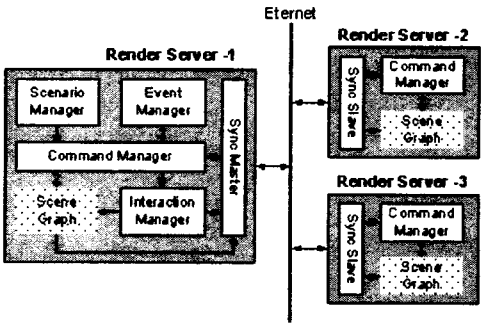

Figure 4. Sync Master and Sync Slave

In Figure 5, our protocol for synchronizing multiple channels on networked hosts is described. A cluster consists of a single Sync Master and zero or more Sync Slaves in the NAVER. A Sync Master sends Update Sync Message (USM) to Sync Slaves every frame. USM contains the change of a scene graph's state and the commands requested by Scenario Manager or Event Manager. Then, each Sync Slave swaps its frame buffer and draws a new frame by reflecting an arrived USM. After a Sync Slave completes its process, an Update Acknowledge Message (UAM) is transmitted to a Sync Master. A Sync Master waits until it receives the UAM from all Sync Slave.

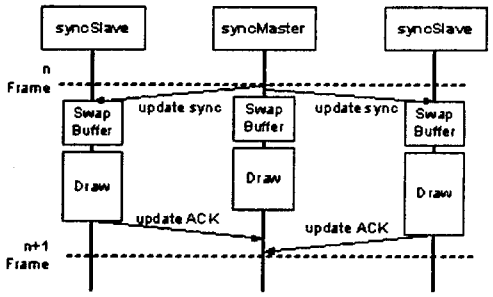

Figure 5. Protocol for synchronizing distributed channels

In the proposed protocol, the overall frame rate depends on the host who has the lowest frame rate. As the drawing time can be different each other, it should be considered for synchronization. In general, 30 Hz frame rate is required. Under this requirement, one frame rendering needs about 33ms and the average transmission time of message is about 0.1 ms on the Ethernet. Therefore, the network does not suffer the bottleneck of our protocol easily. However, as the size of message increases, the bandwidth will be exhausted.

4. Implementation

As we mention before, since NAVER is

based on the PC cluster it can be easily configured for multi-channel displays. In CAVE like display environment, each channel displays to each wall surrounding the user to give the actual image according to the viewpoint of the user.

In this section, we implement CAVE like system that has 4 screens: 3 walls(left, front, right) and 1 floor. Each screen is displayed corresponding a single render server. It is a regular hexahedron (2.2M x 2.2M x 2.2M) whose frame is made of aluminum that has the lower permittivity. BARCO projector as back projection projects three sides of CAVE system. The other projector as downward projection projects the bottom of CAVE system [Fig. 6].

Figure 6. CAVE system

This system is required to consider the followings in order to be a platform that can measure various human factors using the wide field of view for and immersion. First, it should be able to trace the movement of the user and redisplay the proper images for each movement. When redisplaying images, there might be a time delay and image transition occurred by the interaction between the user and the objects. And these make the user feel the difference between the actual world and the virtual world and cause sickness or increase eye strain [7]. To overcome these problems, we make that the tracking device, Flock of Birds, delivers the data faster than the refresh rate of the rendering server and decide the viewing frustum according the user position in CAVE system. In addition, we let the calibration patterns, which is basically a regular grid image, of 4 screens meet to display all images as one image.

In the virtual space, it is required to give not only the 3D information that interacts with examinee but also questionnaire or additional information. To satisfy these requirements, we implement the browsing board that can deliver the 2D data such as message or image in the 3D virtual space. The browsing board is displayed at the prefixed position in the window without being interference with the viewpoint of examinee. Since the browsing board is transparent, it displays messages and images on the 3D virtual space without blocking the scenery to maintain the user immersion context [Fig. 7].

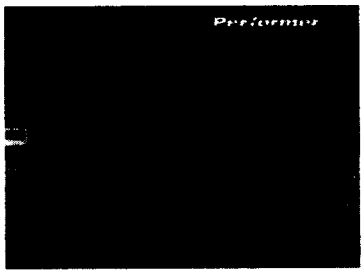

Figure 7. Implementation of the browsing board

Figure 8 shows the procedure to process both the browsing board and the external module in the control server at the same time. And the external module in the control server operates with the database that stores questions for examinee and the answers from them. The browsing board on the render server shows examinee the question coming from the external modules.

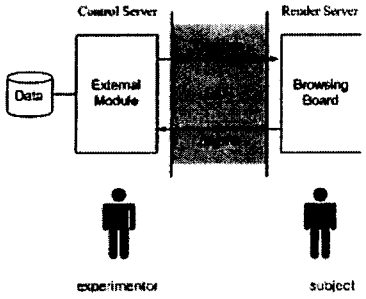

Figure 8. Ask using browsing board

The experimenter can control the start of questionnaire using the external module. When an examinee has done a questionnaire, the answer is delivered through the external module and stored in the database. Therefore, the browsing board can be used in both ways: to give a questionnaire for experiment as 2D information in 3D virtual space, to deliver an additional information.

5. Conclusion

We have described NAVER, a flexible, extensible, scalable re-configurable framework based on a cluster of low-cost PCs. In the NAVER, Scenario Manager provides the script file in order to allow the author specify the definition and operation of the virtual environment which consists of a virtual space and external modules. External modules can run on the remote host and includes various input/output devices and applications. To integrate external modules with a virtual space, we separate them into Device Server and Control Server according to its communication characteristics as Interaction Manager and Event Manger respectively. To support multiple image display channels in HMD, CAVE and so on, NAVER provides an unique technique to synchronize multiple channels that are effective on distributed PC clusters.

References

1. B. MacIntyre and S. Feiner, "A Distributed 3D Graphics Library", ACM *SIGGRAPH '98*, pp. 361-370, 1998.
2. K. Watsen and Mike Zyda. "Bamboo- A Portable System for Dynamically Extensible, Real-time, Networked, Virtual Environments", *IEEE Virtual Reality*, pp. 252-259, 1998.
3. G. Humphreys and P. Hanrahan. "A Distributed Graphics System for Large Tiled Displays". *IEEE Visualization '99*, pp.215-223, 1999.
4. John Kelso, Lance E. Arsenault, "DIVERSE: A Framework for building Extensible and Reconfigurable Device Independent Virtual Environments", Technical Paper.
5. Allen Bierbaum, "VR Juggler: A Virtual Platform For Virtual Reality Application Development". *IEEE Virtual Reality*, pp. 89-96, 2001.
6. DeFanti T. A., "Surround-screen projection-based virtual reality : the design and implementation of the CAVE", ACM *SIGGRAPH 93*, pp. 135-142, 1993.