OpenCABIN ライブラリによるマルチノード没入ディスプレイのアプリケーション開発

Development of Applications for Multi-node Immersive Display using OpenCABIN Library

立山義祐/慶應義塾大学大学院システムデザイン・マネジメント研究科，小木哲朗/慶應義塾大学大学院システムデザイン・マネジメント研究科

Yoshisuke TATEYAMA[1]/Graduate School of System Design and Management, Keio University, Tetsuro Ogi[2]/ Graduate School of System Design and Management, Keio University

*[1]tateyama@sdm.keio.ac.jp, *[2]ogi@sdm.keio.ac.jp

Abstract: We can construct large display systems such as immersive displays CAVE or tiled display using commodity devices like personal computers and local area network. Personal computers become fast enough to render an interactive virtual reality world and local area network devices also become fast. But it is not easy to coordinate these devices as a single display. The OpenCABIN library was developed to support constructing a large display system using multiple PCs and gigabit LAN devices. OpenCABIN library is a fundamental software library for developing virtual reality applications. We show design concepts of OpenCABIN library. Using OpenCABIN library, we implemented some useful system successfully. These results showed the effectiveness of design concepts of the OpenCABIN library.

Keywords: Virtual Environment, Immersive Display, and Software Development

1. Introduction

Some decades ago, to construct an immersive display system we can use full featured graphics workstations. But in these days, these high cost machines are disappeared and personal computers (PCs) become fast enough to render large-scale virtual world. Not only PCs but also display devices and network devices become cheaper. We can easily construct large display system hardware such as a CAVE [Cruz-Neira 93] by combining them (Figure 1) [Soares 08]. To make large display systems work, we may combine multiple screens to show a world and must manage multiple rendering software processes that are designed to cooperate to draw a single world. To coordinate them, you must synchronize information and can use shared memory technique. Shared memory technique is easy method to make multiple rendering processes to cooperate to draw a single world. You can implement this easily to introduce hardware shared memory systems but they are very expensive. In these days, network devices become cheaper and faster, so we can choose an information sharing mechanism via a local area network. But it is difficult to implement it.

We are developing the OpenCABIN library. It is designed for CAVE type VR display system rendered by multiple PCs that are connected via a local area network (Figure 2).

Figure 1: A viewer was operating in an immersive display with 4 screens: KCave.

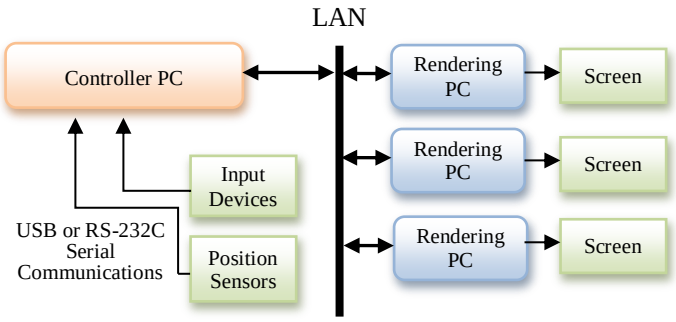

Figure 2: Large display system hardware can be constructed using multiple PCs and local area network devices.

2. OpenCABIN Library

OpenCABIN library is a fundamental software library for developing virtual reality applications. After CABIN library was developed at the university of Tokyo [Hirose 99],

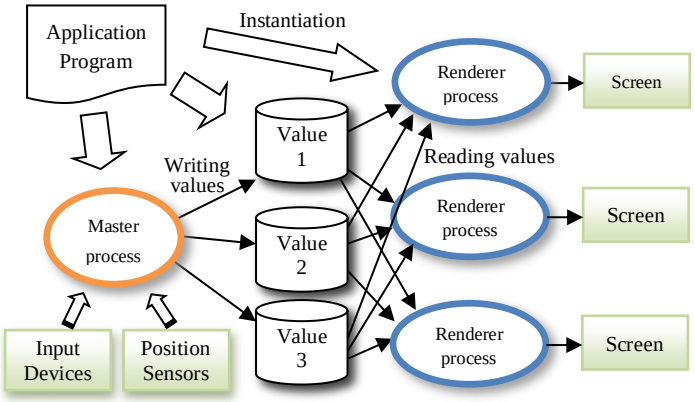

Figure 3: A master part and renderer parts are connected via shared values pointed by shared variables in a VR application.

Figure 4: A user was investigating 3 dimensional hypocentral position data in KCave.

its successor, called OpenCABIN library, was developed from scratch as open-source software. It absorbs differences kinds of display systems, so we can easily develop applications for a multi-screen stereoscopic display system. Figure 1 shows a view that its application is running at the K-Cave system at Keio University. When designing OpenCABIN library we assumed multiple PC display system. Recently multi-screen display systems like CAVEs are consisted of multiple PCs instead of a high-end graphics workstation. In addition to this basic nature as a VR library, it has two special features that enable application programmers to develop VR applications easily: plug-in mechanism and master/renderer programming paradigm.

2.1. Plugin Mechanism

From a software engineering viewpoint such as implementation, testing, debugging, reusability, flexibility, and quality control, it is desirable to construct a system as several independent parts rather than as a big monolithic part. Because of limits of almost all OpenGL implementations, two or more processes cannot access to an OpenGL window. So an OpenCABIN library application is formed as plug-in software and it is loaded and executed by an OpenCABIN library execution environment at runtime. An execution environment can execute one or more plug-in applications simultaneously. As a result, even though each application shows a simple 3D object, virtual space becomes sufficient with a lot of 3D objects. An application user can freely select which object is appeared in the virtual space at runtime.

2.2. Master/Renderer Programming Paradigm with Shared Values

A multiple PCs display system costs cheaper than a high-end graphics workstation, but it is difficult to coordinate PCs to work as a single system. To clear this situation, we introduce a local shared variable mechanism. A local shared variable contains an application's value and is shared among PCs. To implement network-wide shared values, exclusive control is needed and processing speed becomes slow, but if a writer node is limited only one node and remaining nodes only read it, the situation becomes simple, no exclusive control is needed and processing speed becomes fast. Display systems are naturally separated by a master part and renderer parts. The master part of the application determines the application's behaviours. When the application determines representation to the application operator concretely, information of the display content is transferred from the master part to the renderer parts. The reverse direction of information flow cannot exist except for initialization stage of the display system. OpenCABIN library provides programmer written callbacks invoking mechanism and there are callbacks for master part and callbacks for renderer part. We call this OpenCABIN library philosophy: a master/renderer programming paradigm.

An OpenCABIN library application consists of two parts: a master part and a renderer part. A master part is executed in a master process on a master computer, and controls the application's behaviour by producing the application context. A renderer part is executed by renderer processes on renderer computers, and those processes render an application world according to reading the application context. A master part is guaranteed that it is always executed by a master process, so it is easy to develop applications which access outside servers via networks, and applications that share virtual space among CAVEs in remote places. This feature works fine not only at tele-immersive environment but also accessing some servers, for example database servers, license servers, dynamic web pages, web application services and so on.

3. OpenCABIN Library Applications

We developed immersive display applications using OpenCABIN library. These implementation studies showed that OpenCABIN library supported development of immersive display applications. We describe two examples

of applications. First one is seismic data visualization system and second one is an immersive car driving simulator.

3.1. Seismic Data Visualization System

We developed seismic data visualization system [Oonuki 08]. In this system, terrain data and plate structure data are visualized simultaneously according to the hypocentres (Figure 4). Since these data have three-dimensional locations, we organize database tables to have locational data. Therefore the user can see and understand those information intuitively, and will find characteristics of the earthquakes and relationships among earthquakes, terrain shapes and plate structures.

Our seismic data visualization system consists of plural applications. Each application displays single dataset such as hypocentral data, terrain data, basement depth data or plate data. They were developed using OpenCABIN library. Because each dataset includes locational information such as latitude and longitude, these data can be merged at the same location. A user can choose any combination of data types. In other seismic data visualization system we developed, a user was operating to visualize the hypocenter data and terrain data around Tsukuba city. In this system, images acquired from a satellite are texture mapped onto the terrain shapes. Each sphere indicated a hypocenter: the sphere's position, color and radius indicated hypocentral location, depth and magnitude. She/he could understand each earthquake intuitively.

Figure 1 shows that a user was seeing a combination among hypocenter, basement depth data and plate data. Views of combination among basement depth data, sea depth data, Pacific Ocean plate data and Philippine Sea plate data enabled a user to understand relationships of these data.

Through the operation of this system, an expert found that the depths of hypocenters in west Japan is relatively shallow and hypocenters from Tokai to Kanto are distributed on a plane. Also we can apparently see that hypocenters are distributed along the plate.

3.2. Immersive Car Driving Simulator

To decrease traffic accidents, we want to observe drivers' behaviors to find ways to drive safely. Observations at the real environment include many risks to have needless but critical accidents. Using a car driving simulator, we can observe drivers' behaviors in dangerous situations safely. We constructed an immersive car driving simulator using KCave.

Figure 5 shows an immersive car driving simulator using OpenCABIN library in KCave. Half of KCave's floor screen was removed and a car cockpit was located. Our car cockpit was composed of real car parts: a steering wheel, a brake pedal, a gas pedal and a seat. Driver's head position was traced so she/he can look around and move the head forward to watch narrow crossroad.

Course model data of the driving simulator was created from the real town using commodity 3D model authoring tool: Autodesk 3ds max (Figure 6).

Figure 5: A subject was driving in an immersive car driving simulator.

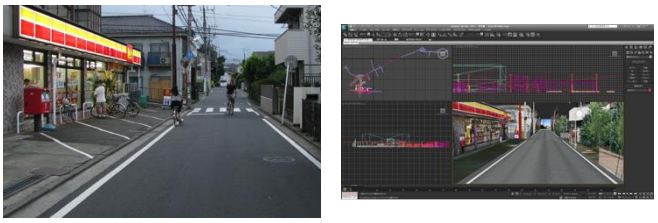

Figure 6: Course model data of the driving simulator was created from the real town using commodity 3D model authoring tool: Autodesk 3ds max.

We conducted some observation experiment where the drivers watch when she/he drive to turn right in a narrow crossroad. An elder driver can drive and accomplish this task in this immersive virtual environment.

4. OpenCABIN library for tele-immersive environment

OpenCABIN library is designed to abstract variations of display devices configurations. If you want to implement virtual conference system using multiple large display systems like CAVEs or tiled displays at remote places, you must think about variations of different display system configurations. But OpenCABIN library's display device abstraction mechanism reduces application development cost. Because of master/renderer programming paradigm, it is easy to develop some applications that communicate other display systems.

5. Discussion

Design choice of Application Programming Interface (API) is important for programmers. There are some useful scenegraph APIs such as OpenGL Performer [Rohlf 94], OpenSG [Reiner 02], VTK [Schroeder 96] and Syzygy [Shaeffer 03]. Scenegraph is a powerful technique for distributed environment where system can know high level knowledge of the contents and decrease amount of communication. Though OpenGL is low level API, there are

many programmers familiar with it and graphics hardware vendors provide novel features via OpenGL. So it is important for virtual environment to support OpenGL API.

Some useful systems were proposed to develop applications for multiple PC display systems using OpenGL API. WireGL [Humphreys 01] and its successor Chromium [Humphreys 02] capture OpenGL commands of application program that is executed at master node and distributes them to rendering nodes. This approach is powerful because system does not require programmers to modify the application program. But it always requires systems to communicate some amount of information between the master node and rendering nodes at runtime.

CaveLib [Pape 97] is de-facto standard fundamental software for CAVE system. All nodes execute a single application program and its execution is synchronized at some code points (typically end of the rendering iteration). States of the input devices such as Wand and head tracking data are distributed to rendering nodes by CaveLib system. This system works well if the application data set is limited.

Our OpenCABIN library approach is partially same as CaveLib at the point that the application program is executed by all nodes. But OpenCABIN library has shared variable mechanism and the application can acquire external data even if it is from the Internet.

6. Conclusion

We implemented a version of the OpenCABIN library and released it as open source software. It can help to develop VR applications in large display systems that consist of multiple screens and PCs. Using OpenCABIN library, a tele-immersive application was implemented successfully. This result showed the effectiveness of design concepts of the OpenCABIN library.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number 23500154.

References

Cruz-Neira, C., Sandin, D., DeFanti, T. Surround-Screen Projection-based Virtual Reality: The Design and Implementation of the CAVE, SIGGRAPH'93: Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques, pp. 135–142, 1993.

Hirose, M., Ogi, T., Ishiwata, S., Yamada, T., 1999. Development and Evaluation of Immersive Multiscreen Display "CABIN". Systems and Computers in Japan, Scripta Technica, Vol.30, No.1, pp.13-22, 1999.

Humphreys, G., Eldridge, E., Buck, I., Stoll, G., Everett, M., Hanrahan, P., WireGL: a scalable graphics system for clusters, Proceedings of the 28th annual conference on Computer graphics and interactive techniques, p.129-140, 2001.

Humphreys, G., Houston, M., Ng, R., Frank, R., Ahern, S., Kirchner, P., Klosowski, J. Chromium: a stream-processing framework for interactive rendering on clusters, ACM Transactions on Graphics (TOG), v.21 n.3, July 2002.

Oonuki, S., Tateyama, Y., Ogi, T. 2008. Seismic Data Visualization in Tele-immersion Environment. Asiagraph 2008 in Tokyo Proceedings, Vol. 2, No. 2, pp. 186-189, 2008.

Pape, D., Carolina Cruz-Neira, C., Czernuszenko, M., CAVE User's Guide, http://www.evl.uic.edu/pape/CAVE/prog/CAVEGuide.html, 1997.

Reiners, D. Opensg: A Scene Graph System for Flexible and Efficient Realtime Rendering for Virtual and Augmented Reality Applications, Darmstadt dissertation, 2002.

Rohlf, J. and Helman, J. IRIS performer: A high performance multiprocessing toolkit for real–Time 3D graphics. Proceedings of SIGGRAPH ’94, pages 381–395, 1994.

Schaeffer, B. and Goudeseune, C. Syzygy: Native PC Cluster VR, in IEEE VR Conference, 2003.

Schroeder, William J. and Martin, Kenneth M. and Lorensen, William E. The design and implementation of an object-oriented toolkit for 3D graphics and visualization. Proceedings of the 7th conference on Visualization '96, 1996.

Soares, L., Raffin, B., Jorge, J. PC Clusters for Virtual Reality. The International Journal of Virtual Reality, 7(1):67-80 67, 2008.