Visualization framework for CAVE virtual reality systems

Kageyama, Akira

Tomiyama, Asako

Published in
International Journal of Modeling, Simulation, and Scientific Computing, vol7, 1643001 (2016)

Visualization Framework for CAVE Virtual Reality Systems

Akira Kageyama and Asako Tomiyama

Department of Computational Science,
Kobe University, Kobe 657-8501, Japan

kage@port.kobe-u.ac.jp

Abstract

We have developed a software framework for scientific visualization in immersive-type, room-sized virtual reality (VR) systems, or CAVEs. This program, called Multiverse, allows users to select and invoke visualization programs without leaving CAVE's VR space. Multiverse is a kind of immersive "desktop environment" for users, with a three-dimensional graphical user interface. For application developers, Multiverse is a software framework with useful class libraries and practical visualization programs as samples.

1 Introduction

This is a significantly extended paper of our work presented at AsiaSim2015[1].

The cave automatic virtual environment (CAVE) is a room-sized, immersive virtual reality (VR) system developed in the early 1990s[2]. To this day, it continues to provide a superior VR experience to those offered by other VR systems, such as head-mounted displays. In spite of its sizable volume and cost, the popularity of the CAVE system as a research tool for science has not declined over time. The original CAVE has a cubic geometry with an edge length of 10 feet. Besides the straightforward extension to cuboids[3], other geometries of CAVEs, such as a polyhedron with more faces[4], a cylinder[5], and a sphere[6], have been constructed. Software for CAVEs, or more generally, for large display systems, has also evolved hand-in-hand with developments in hardware[7]. A common feature of most CAVE systems is that a highly immersive experience is afforded to users by surrounding them with large screens on which stereoscopic images are projected. The perspective of the images is automatically adjusted to a viewer's eyes by a head-tracking system. A user in a CAVE room interacts with VR objects through a portable controller, which is sometimes called a wand.

Applications of CAVE systems to scientific research span a broad spectrum, from archaeology[8] to medical science[9, 10] and elementary particle physics[11, 12]. One of the earliest CAVE programs was a visualization tool for computational fluid dynamics (CFD)[13]. The visualization of CFD data continues to be an important application of CAVEs[14, 15, 16, 17, 18]. For instance, a new structure in electric current was recently discovered in a magnetohydrodynamics simulation using a CAVE VR visualization[19].

Some general-purpose visualization software packages for personal computers (PCs) and graphic workstations (GWSs) have been ported to CAVE systems. These include Visualizer[21], ML2VR (MATLAB)[20], Amira, ParaView, VisIt, AVS/Express, and others. We too have developed a general-purpose visualization program, VFIVE, for CAVEs[22, 23, 24], the main application of which is also CFD. However, there is no definitive application program for scientific visualization in CAVEs. When scientists today try to use CAVE systems to visualize their data, there are multiple visualization programs available for the CAVE system to choose from, many of which might have been developed in-house.

From the user's point of view, a rich set of multiple CAVE applications is not necessarily fortunate, for each program has to be invoked one-by-one from a terminal that is usually located outside the CAVE room. A user would hence need to send commands each time he/she enters the CAVE room and dons the stereo glasses. This inconvenience would need to be repeated every time the user switches to a subsequent

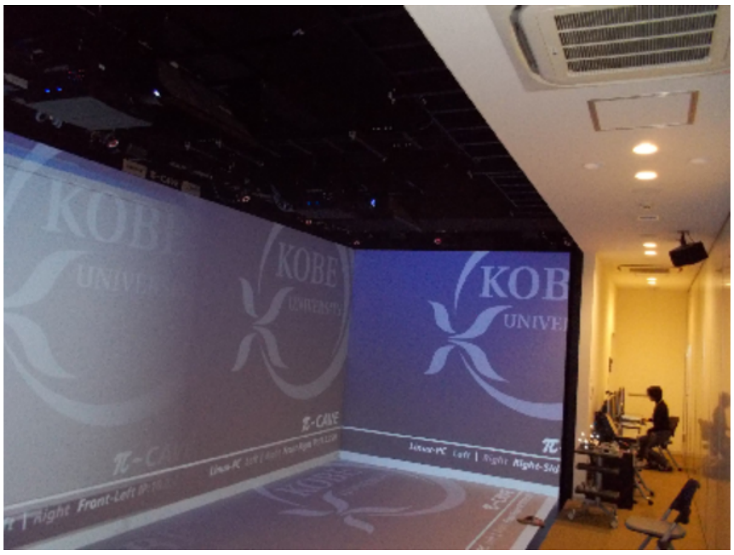


Figure 1: A four-screen, cuboid-shaped CAVE system, π-CAVE, at Kobe University, Japan. The edge lengths are 3 m $\times$ 3 m $\times$ 7.8 m. This is currently the largest CAVE system in Japan. It is at Port Island (P. I. $\rightarrow \pi$), Kobe city.

application. This scenario is akin to having a PC without an operating system. What is needed here is a kind of meta-level CAVE software that allows users to control multiple CAVE applications without leaving the CAVE room.

We have developed such an application program for CAVEs. This program, Multiverse, was originally developed for the largest CAVE system in Japan, π-CAVE, at Kobe University[25]. Soon after the development of π-CAVE, we developed five new visualization applications, and then recognized the necessity of the above-mentioned meta-level software. The first version of Multiverse was thus developed[26]. It was itself a CAVE application program based on the standard CAVElib application programming interface (API). The original Multiverse was therefore an application program dedicated to the π-CAVE system. It was designed to load the other five application programs that we had developed, and all data relevant to the applications were loaded onto the main memory at the beginning of the program's execution.

Since Multiverse garnered a favorable reception from users of π-CAVE, we decided to convert it into a general framework that may be useful for other CAVE systems. The main improvements to the original Multiverse are: (i) the number of loadable applications to Multiverse is now arbitrarily large, and (ii) the "look and feel" of each visualization application has been unified.

Multiverse provides not only an easy-to-use visualization environment for users, but also a framework for application developers. Fundamental techniques frequently used in CAVE visualization programs, such as a stereo volume rendering and high-speed visualizations using ball-shaped objects through the point sprite method, have been built into Multiverse as class libraries. The user interface, which is generally the most bothersome part of CAVE visualization applications, is controlled by Multiverse, hence allowing application developers to focus on the visualization algorithms.

Since Multiverse uses only basic APIs—OpenGL and CAVElib—as CAVE applications, it is easy to port it to any CAVE-type system with these APIs. In fact, we installed Multiverse with no problem to another CAVE system[27] with a different configuration at Kobe University. It would also be easy to port to CAVE systems with other API such as VR Juggler[28].

Before we describe the software structure of Multiverse in the next section, we summarize π-CAVE's hardware system. The π-CAVE room is a cuboid of 3 m $\times$ 3 m $\times$ 7.8 m (Fig. 1). It has four screens: three wall screens (front, right, and left), and a floor screen. The floor image is projected from two projectors mounted on the ceiling. The wall images are rear-projected. Six sets of Digital Light Processing (DLP) projectors (Christie WU12K-M) are used with 1920 $\times$ 1200 pixels and $10,500$ lm. For head and wand tracking, an optical motion tracking system (Vicon) is used with 10 cameras (640 $\times$ 480 pixels each). The Trackd API is used to retrieve the tracking data. One of two computer systems is used for computation and rendering. One is a Linux-PC (HP Z800) with 192 GB of shared memory. Three sets of external

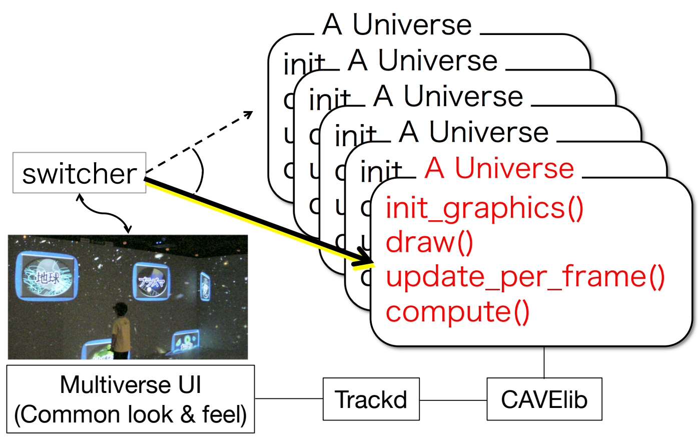


Figure 2: Conceptual diagram of Multiverse.

GPUs (NVIDIA Quadro PLEX) are used for graphics. The other computer system is a Windows-PC cluster system. The Windows-PC system is used when we use commercial software for CAVEs. We developed Multiverse mainly on the Linux-PC system.

2 Overview

2.1 Execution

Multiverse is a CAVE program that constructs a three-dimensional (3D) desktop environment in the CAVE's VR space as the interface to other CAVE applications. Each application is called a "Universe." A Universe is almost a stand-alone CAVE application with its own data and tasks. The only difference between a stand-alone application and a Universe is in the user interface: requests by a user for a Universe are sent to and controlled by Multiverse (see Fig. 2).

When a user starts Multiverse, small virtual panels, or icons, appear in the CAVE's VR space. See Fig. 3. They look like they are floating in the air in front of the viewer. Each icon represents a Universe. This mode of Multiverse, where application icons float, is called "World."

The World mode corresponds to the "desktop" of the PC's graphical user interface. The wand of the CAVE is a 3D mouse. When one of the panels is "clicked" with the wand in a manner described in the next subsection, the corresponding Universe is selected and its data is loaded from the hard disk drive to Multiverse. Data transfer may take seconds to minutes depending on size. This period is used to present information of the selected Universe to the viewer by showing a sequence of expository images. A sample shot taken during this interval is shown in Fig. 4. Descriptions of the selected Universe, such as visualization methods, simulation schemes, and the background of the simulation, are presented during this time.

When data transfer is complete, a panel titled "Entrance" appears in the middle of the CAVE room. The visualization of the Universe begins when this panel is clicked.

When the user quits a Universe, data concerning the Universe is deleted from memory, and Multiverse returns to the World mode; the user can then select the next Universe.

2.2 User Interface

In Multiverse, it is assumed that the wand has at least three buttons and one joystick. When there are other buttons and joysticks, these are ignored. By default, the joystick is assigned to control translation and rotation in the graphics world coordinates, which is called navigation in CAVE applications. Navigation is realized in Multiverse as follows: pushing the joystick along the positive/negative "y" axis causes viewpoint translation forward/backward in the direction of the wand's front vector at the given

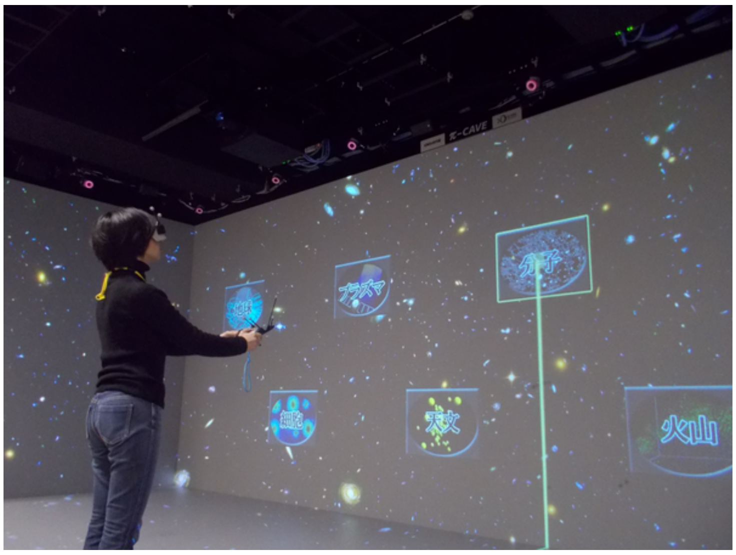
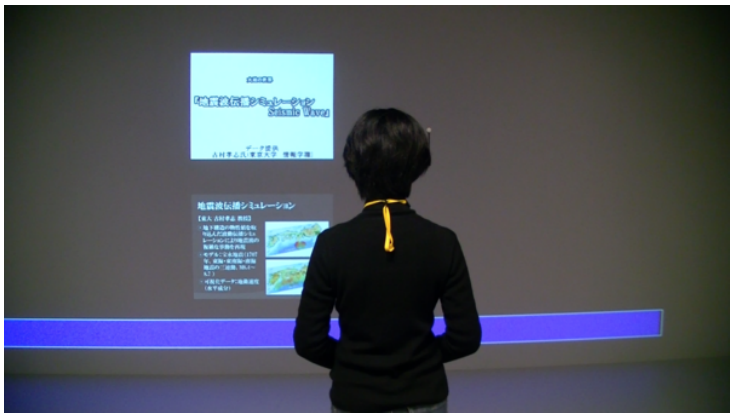


Figure 3: The "World" mode of Multiverse, where icons for registered applications ("Universes") are floating in the VR space. The user selects one of the Universes by "clicking" the corresponding icon with a virtual laser beam emitted from the controller, or wand.



「地震波伝播シミュレーション Seismic Wave」
地震波伝播シミュレーション

Figure 4: When a dataset for a selected Universe is loaded to Multiverse, expository images of the Universe are presented to the viewer, such as the adopted visualization method, simulation parameters, and the background of the simulation.

time, whereas pushing the joystick along the “x” axis causes viewpoint rotation around the vertical axis perpendicular to the floor of the CAVE room.

Button 1 on the wand is used to select a panel. When panels are shown in the World mode of Multiverse or in a Universe, a beam is emitted in the direction of the wand’s front vector. When the beam pierces a panel, the panel assumes a focused—or clickable—state, and if Button 1 of the wand is pressed in this state, the panel is selected, or “clicked.”

There is no default setting for Button 2 of the wand. Each Universe can define the function of this button. In one of our sample Universes, for example, this button is used to switch to the next data item. In visualization tasks in a Universe, in general, it is seldom the case that there is only one target data item to be analyzed. It is rather common for there to be several data to be visualized. The Button 2 can be used to change the target data.

Button 3 of the wand is assigned two functions. A short press of Button 3 opens a set of menu panels defined in each Universe. A long press of this button opens a different kind of menu panel used to exit the given Universe and return to the World mode. A menu has a special panel to close; the whole menu panels are erased by selecting the panel.

Developers of new Universes can overwrite the above (default) setting of the wand buttons in custom-made Universes.

3 Program Structure

3.1 Framework and Execution Flow

A visualization program ”Universe” is an instance of a class called `Universe`. `Universe` is derived from the base class `Vacuum` whose functions are summarized in Table 1. Although multiple instances of Universe are controlled by Multiverse, only one Universe runs at a time, and states of the running Universe, shown in Fig. 5, are controlled by a manager class called `UniverseManager`. Details of these functions and states are described in the following.

Table 1: Functions in the Vacuum class.

Executed by the main process	
`load_data`	To load data onto shared memory
`compute`	To perform computations independent of drawing
Executed by master display thread	
`update`	To update variables in shared memory
`clear_data`	To delete data in shared memory
Executed by all display threads	
`start`	To start initializing visualization
`stop`	To finalize visualization (calling `exit_graphics`, and `clear_data`)
`init_graphics`	To generate texture objects, buffer objects, etc.
`update_graphics`	To update the above objects
`exit_graphics`	To remove the above objects
`draw`	To draw graphics.

3.2 Loading/Clearing Data

Consider a CFD simulation. The target data to be analyzed are 3D vector fields (e.g., flow velocity and vorticity) and scalar fields (e.g. pressure and enstrophy). When a Universe to visualize them is invoked from the World mode of Multiverse, the function `start` is called to initialize the parameters of the Universe before the data are loaded; the state is then changed to LOAD_DATA: See the rightmost column of Fig. 5. In this state, the function `load_data` in Table 1 is called to transfer the data to the shared memory. The expository images mentioned in § 2.1 and a progress bar (the horizontal blue bar in the lower part of Fig. 4) are presented by the display threads while `load_data` is running in the main process. In the next state INIT_GRAPHICS, each display thread calls function `init_graphics` to

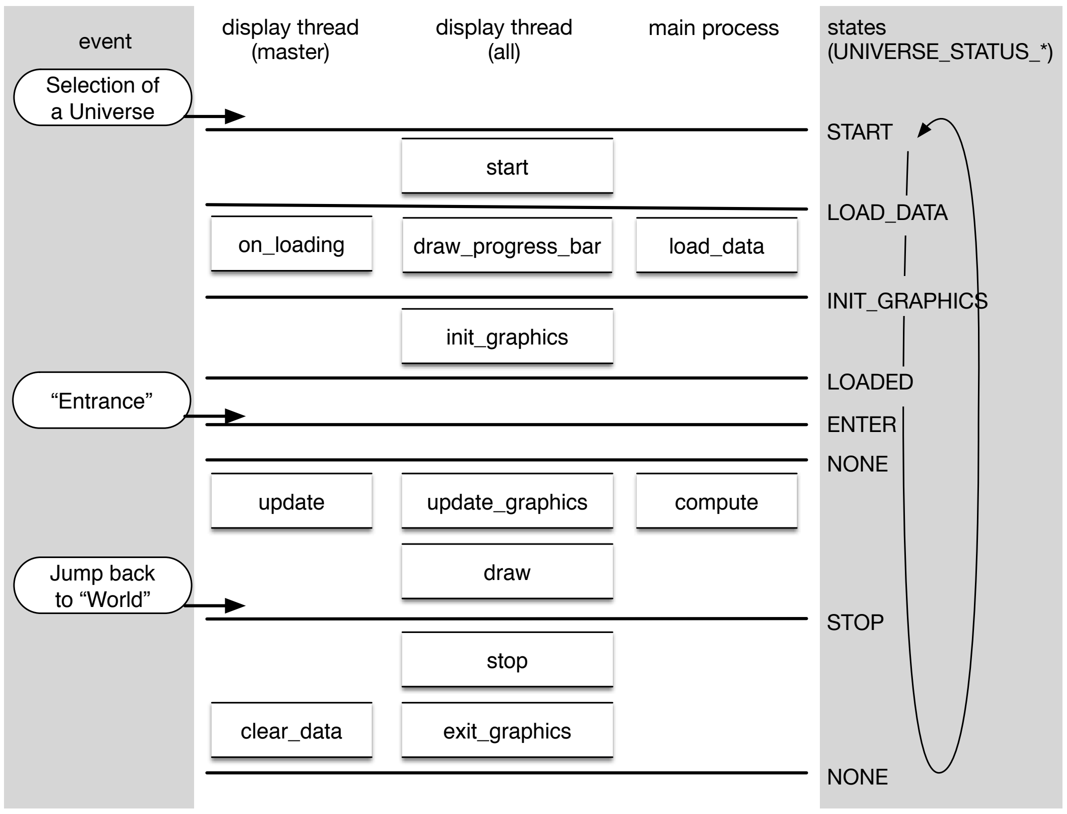


Figure 5: Execution flow of Multiverse.

perform initialization for rendering with OpenGL. In this function, OpenGL's objects, such as the vertex buffer objects and texture objects, are created and initialized with the loaded data.

When the user exits the Universe, the functions `exit_graphics` and `clear_data` are called to free the memory allocated to `init_graphics` and `load_data` before returning to the World mode.

3.3 Updating Data

Each Universe has two types of processes to update data: "frame update" and "calculation."

The functions `update` and `update_graphics` in Table 1 are for "frame update." In each frame, `update` is first called by the master display thread, which updates variables in shared memory, such as the frame index or the positions of the displayed objects. Other display threads wait for `update` to finish, following which all the display threads execute `update_graphics` to update OpenGL objects. In our sample Universes, included in Multiverse, the data necessary for new objects are, in most cases, sent to the GPUs at the beginning of each Universe.

The function `compute`, which is for the "calculation" process, is called by the main process and is thus executed asynchronously from the display threads. Computationally heavy tasks that may reduce the frame rate are usually assigned to `compute`.

3.4 Drawing

The main part of each Universe as a visualization program is the function `draw`. One may implement any OpenGL call in this function. It is, however, critically important to keep a high frame rate to realize the real-time response to the user's head (eyes) motion in a CAVE. In many cases, it is effective to use OpenGL buffer objects. In a sample Universe called SeismicWave included in Multiverse (see Fig. 6), where the time development of the ground velocity of a seismic wave is visualized for N steps of time in a simulation[29], N vertex buffer objects are created for the ground velocity data.

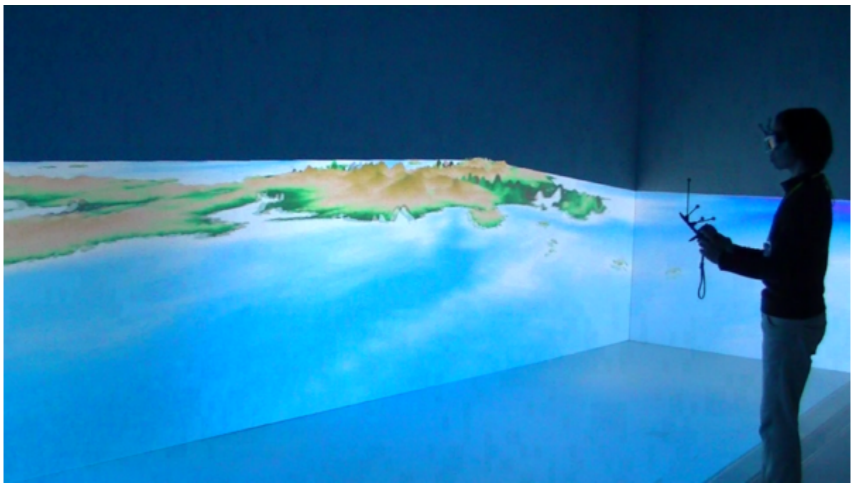


Figure 6: A Universe called SeismicWave. In this Universe, a height plot visualization of the ground velocity of a seismic wave propagation simulation is shown in an animation.

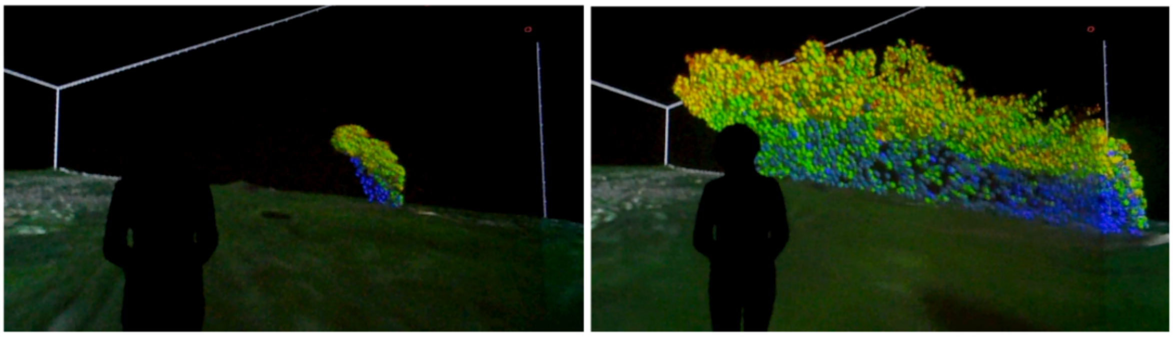


Figure 7: Sample Universe called VolcanicAsh included in Multiverse. Tens of thousands of balls are used to visualize the motion of ash particles in a volcanic eruption.

3.5 Application Universes

Figure 7 shows another sample Universe called VolcanicAsh. This is a CFD simulation of a volcanic eruption by Suzuki and Koyaguchi[30]. Volcanic ash particles traced in the simulation are visualized in this Universe. The position of the ashes is sent to GPUs as vertex buffer objects.

The ball-like objects in Fig. 7 depict the volcanic ash particles. They are drawn by textures using point sprite. Here, the particles are grouped by diameter and drawn by applying the point sprite technique to each group with different point sizes and colors.

As in the usual visualization applications, a popular visualization method for scalar fields is volume rendering. We implemented volume rendering for CAVEs for Multiverse on the basis of VSVR (Very Simple Volume Rendering) program developed by Lewiner[31]. Figure 8 shows a Universe, CellDivision, where VSVR is used. A time sequence of microscope images is visualized as an animated stereo volume rendering in the CAVE.

Another sample Universe in our Multiverse is GeomagField, a snapshot of which is shown in Figs. 9 to 11. It is a visualization application of a magnetohydrodynamics simulation of the Earth's interior[19, 32]. A variety of visualization methods for vector and scalar fields are part of GeomagField. In fact, most visualization methods implemented in a general visualization program VFIVE[22, 23, 24] for CAVEs are included in GeomagField.

As described in this subsection, our sample Universes includes basic rendering techniques for CAVE programs (buffer objects, point sprite, and stereo volume rendering), so they would be bases for development of new programs.

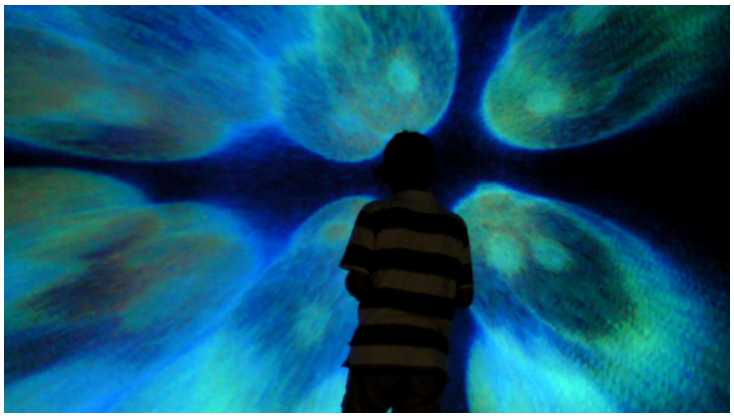


Figure 8: Sample Universe, CellDivision, for stereo volume rendering in CAVEs. Time development of a three-dimensional scalar field is visualized as an animated stereo volume rendering.

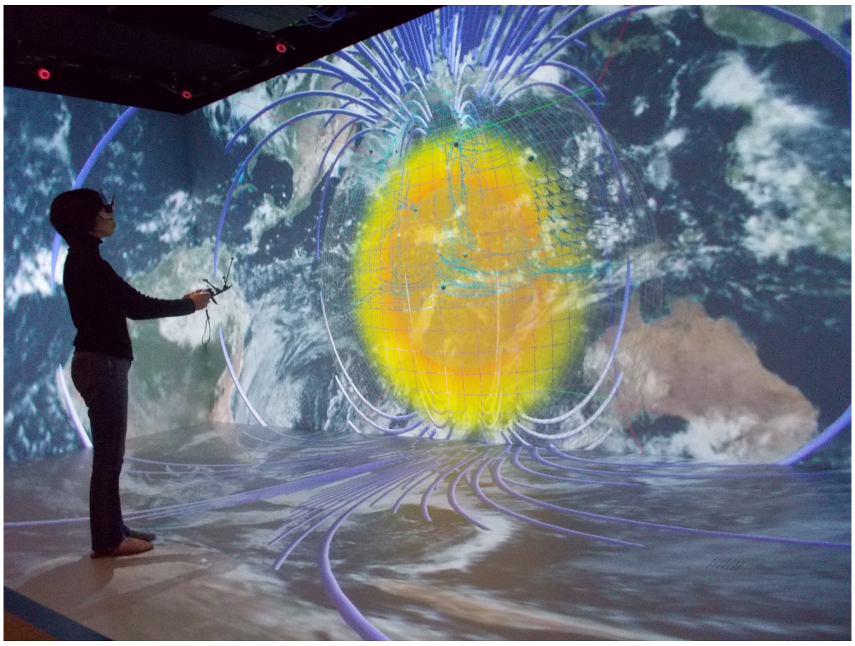


Figure 9: Another sample Universe, GeomagField. The flow and magnetic fields in a magnetohydrodynamics simulation are analyzed by various visualization methods for scalar and vector fields. In this snapshot, the temperature distribution is shown by a volume rendering.

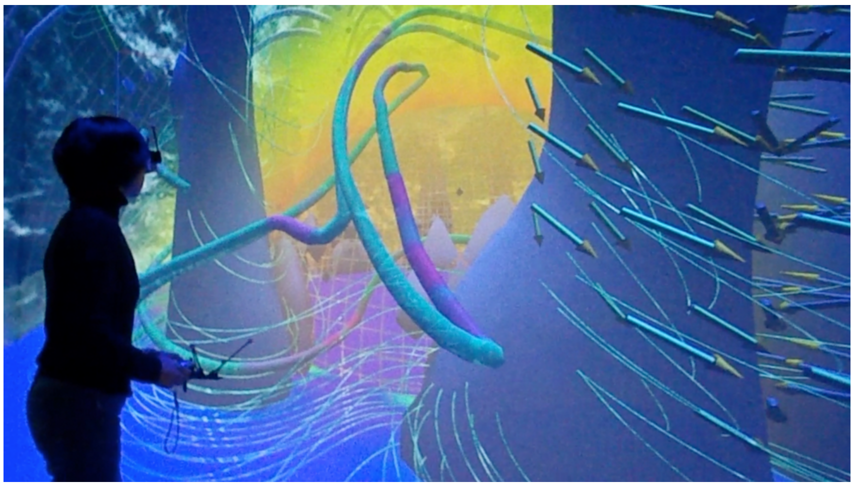


Figure 10: Visualization in GeomagField. Flow velocity is shown by arrow glyphs and stream lines (thin blue curves). The positions of the arrow glyphs and the seed points of the stream lines are interactively controlled by hand (or wand). The vertical bluish pillars represent isosurfaces of vorticity. A magnetic field line that is frozen in, or advected by, the flow is shown by the colored tube.

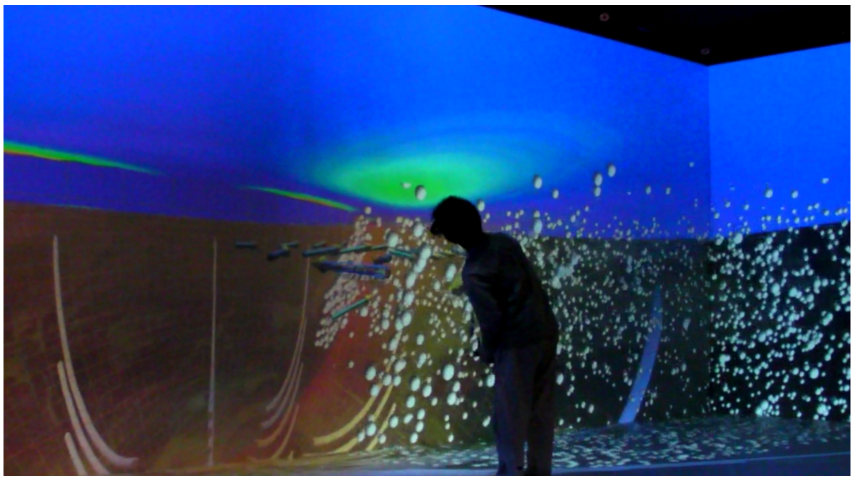


Figure 11: Another snapshot of visualization in GeomagField. The convection flows in the Earth's core are visualized by tracer particles (white balls). Vorticity distribution in the equatorial plane is shown by the color contour. The tracer particles appear in a conic region lit by a handheld virtual flushlight.

4 Development of New Visualization Applications

As mentioned in § 3.1, `Vacuum` is the base class for every visualization application (Universe). A new Universe can be developed by creating a subclass of `Vacuum` and implementing the functions listed in Table 1, and by adding custom functions to the Universe.

Every Universe has two kinds of image data. One is an icon image shown in the floating menu in the World mode. The other is a group of images used in the exposition presented while data relating to a Universe data is being loaded. The icon images registered to Multiverse are stored in "universe-Menu," which is a member of the `World` class. The expository images are stored in a member of the `UniverseManager` called "menu," which is an instance of a class called `UniversePanel`.

The class `UniverseManager` manages all Universes registered in Multiverse. More precisely, each Universe is registered as an instance of class `UniverseEntry`, which contains the object of a Universe together with its application name as a character string. The set of instances of the `UniverseEntry` class is stored as an array called "uniList," which is a member of `UniverseManager`.

5 Summary

In this study, we described our design and implementation of a 3D desktop environment, Multiverse, for CAVE-type VR systems. Using this environment, a user in the CAVE's VR space can select visualization programs (Universes) by "clicking" 3D icons that float in the CAVE's VR space. From the viewpoint of application development, Multiverse provides a kind of software framework which helps create new CAVE visualization applications. One can produce a new visualization program for CAVE systems by inheriting a basic class called `Vacuum`, whose member functions are assigned to threads or processes characteristic of CAVE programs. Because the user interface is controlled by Multiverse, the developer can focus on the visualization algorithms.

Acknowledgment

The data for SeismicWave, VolcanicAsh, and CellDivision were provided by Drs. T. Furumura, Y. Suzuki, and K. Yamagata, respectively. This work was supported by a Grant-in-aid for Scientific Research (KAKENHI) 23340128.

References

[1] A. Kageyama, A. Tomiyama: Multiverse: A Software Framework for Visualization in CAVE Virtual Reality Systems, Proceedings of AsiaSim2015 (2015) 86–93.

[2] C. Cruz-Neira, D. J. Sandin, T. A. Defanti, Surround-screen Projection-based Virtual Reality: The Design and Implementation of the CAVE, Proceedings of SIGGRAPH '93 (1993) 135–142.

[3] T. W. Kuhlen, B. Hentschel, Quo Vadis CAVE: Does Immersive Visualization Still Matter?, IEEE Computer Graphics and Applications (5) (2014) 14–21, doi:10.1109/MCG.2014.97.

[4] T. A. DeFanti, G. Dawe, D. J. Sandin, J. P. Schulze, P. Otto, J. Girado, F. Kuester, L. Smarr, R. Rao, The StarCAVE, a Third-generation CAVE and Virtual Reality OptiPortal, Future Generation Computer Systems 25 (2) (2009) 169–178, doi:10.1016/j.future.2008.07.015.

[5] A. Febretti, A. Nishimoto, T. Thigpen, J. Talandis, L. Long, J. D. Pirtle, T. Peterka, A. Verlo, M. Brown, D. Plepys, CAVE2: A Hybrid Reality Environment for Immersive Simulation and Information Analysis, Proceedings of IS & T/SPIE Electronic Imaging, The Engineering Reality of Virtual Reality 2013, San Francisco, CA (2013).

[6] J. Kuchera-Morin, M. Wright, G. Wakefield, C. Roberts, D. Adderton, B. Sajadi, T. H. Höllerer, A. Majumder, Immersive Full-surround Multi-user System Design, Computers & Graphics 40 (2014) 10–21, doi:10.1016/j.cag.2013.12.004.

[7] H. Chung, C. Andrews, C. North, A Survey of Software Frameworks for Cluster-based Large High-resolution Displays, Visualization and Computer Graphics, IEEE Transactions on 20 (8) (2014) 1158–1177, doi:10.1109/TVCG.2013.272.

[8] D. Acevedo, E. Vote, D. H. Laidlaw, M. S. Joukowsky, Archaeological Data Visualization in VR: Analysis of Lamp Finds at the Great Temple of Petra, A Case Study, Proceedings of the Conference on Visualization'01, IEEE Computer Society (2001) 493–496.

[9] S. Zhang, C. Demiralp, D. F. Keefe, M. DaSilva, D. H. Laidlaw, B. D. Greenberg, P. J. Basser, C. Pierpaoli, E. A. Chiocca, T. S. Deisboeck, An Immersive Virtual Environment for DT–MRI Volume Visualization Applications: A Case Study, Proceedings of Visualization 2001 (VIS'01) (2001) 437–584.

[10] B. Chen, J. Moreland, J. Zhang, Human Brain Functional MRI and DTI Visualization with Virtual Reality, ASME 2011 World Conference on Innovative Virtual Reality, American Society of Mechanical Engineers (2011) 343–349.

[11] B. Izatt, K. Scholberq, R. P. McMahan, Super-kave: An Immersive Visualization Tool for Neutrino Physics, Proceedings of IEEE Virtual Reality 2013 (2013) 75–76, doi:10.1109/VR.2013.6549370.

[12] R. Tredinnick, J. Vanderheiden, C. Suplinski, J. Madsen, CAVE Visualization of the IceCube Neutrino Detector, Proceedings of IEEE Virtual Reality 2014 (2014) 117–118, doi:10.1109/VR.2014.6802079.

[13] V. Jaswa, CAVEvis: Distributed Real-time Visualization of Time-varying Scalar and Vector Fields Using the CAVE Virtual Reality Theater, Proceedings of Visualization '97 (1997) 301–308.

[14] H. M. Tufo, P. F. Fischer, M. E. Papka, K. Blom, Numerical Simulation and Immersive Visualization of Hairpin Vortices, Proceedings of the 1999 ACM/IEEE conference on Supercomputing (1999) 62.

[15] D. Fu, B. Wu, G. Chen, J. Moreland, F. Tian, Y. Hu, C. Q. Zhou, Virtual Reality Visualization of CFD Simulation for Iron/Steelmaking Processes, Proceedings of the 14th International Heat Transfer Conference (2010) 1–8.

[16] N. Yan, T. Okosun, S. K. Basak, D. Fu, J. Moreland, C. Q. Zhou, Numerical Simulation and Virtual Reality Visualization of Horizontal and Vertical Axis Wind Turbines, Proceedings of the ASME 2011 International Design Engineering Technical Conference & Computers and Information in Engineering Conference IDETC/CIE 2011 (2011) 1–8.

[17] A. S. Forsberg, D. H. Laidlaw, A. Van Dam, R. M. Kirby, G. E. Karniadakis, J. L. Elion, Immersive Virtual Reality for Visualizing Flow through an Artery, Proceedings of the conference on Visualization'00, IEEE Computer Society Press (2000) 457–460.

[18] D. J. Quam, T. J. Gundert, L. Ellwein, C. E. Larkee, P. Hayden, R. Q. Migrino, H. H. Otake, J. F. LaDisa, Immersive Visualization for Enhanced Computational Fluid Dynamics Analysis, Journal of Biomechanical Engineering 137 (2015), 031004, doi:10.1115/1.4029017.

[19] A. Kageyama, T. Miyagoshi, T. Sato, Formation of Current Coils in Geodynamo Simulations, Nature 454 (7208) (2008) 1106–9, doi:10.1038/nature07227.

[20] D. J. Zielinski, R. P. McMahan, W. Liu, and S. Ferrari, ML2VR: Providing MATLAB Users an Easy Transition to Virtual Reality and Immersive Interactivity, Proc. IEEE Virtual Reality Conference, (2013) 83-84

[21] M. I. Billen, O. Kreylos, B. Hamann, M. A. Jadamec, L. H. Kellogg, O. Staadt, and D. Y. Sumner, A Geoscience Perspective on Immersive 3D Gridded Data Visualization, Comput. & Geosci. 34 (2008) 1056–1072

[22] A. Kageyama, Y. Tamura, T. Sato, Visualization of Vector Field by Virtual Reality, Progress of Theoretical Physics Supplement 138 (2000) 665–673.

[23] A. Kageyama, N. Ohno, Immersive VR visualizations by VFIVE. Part 1: Development, International Journal of Modeling, Simulation, and Scientific Computing 4 (2013) 1340003, doi:10.1142/S1793962313400035.

[24] A. Kageyama, N. Ohno, S. Kawahara, K. Kashiyama, H. Ohtani, Immersive VR Visualizations by VFIVE. Part 2: Applications, International Journal of Modeling, Simulation, and Scientific Computing 4 (2013) 1340004, doi:10.1142/S1793962313400047.

[25] A. Kageyama, Y. Yamaura, D. Meno, Y. Masada, K.-I. Yoshizaki, K. Yamada, Application Launcher for CAVE, Proc. International Conference on Modeling and Simulation Technology (JSST 2011), Takanawa, Tokyo, 2011, 286–290.

[26] A. Kageyama, Y. Masada, Applications and a Three-dimensional Desktop Environment for an Immersive Virtual Reality System, J. Phys.: Con. Ser. 454 (2013) 012077, doi:10.1088/1742-6596/454/1/012077.

[27] K.-I. Yoshizaki, A. Kageyama, Dynamical Visualization of Vector Field via Multiple Streamlines in Virtual Reality Environment, Memoirs of the Graduate Schools of Engineering and System Informatics Kobe University 5 (2013) 7–9, doi:10.5047/gseku.e.2013.002.

[28] A. Bierbaum, C. Just, P. Hartling, K. Meinert, A. Baker, C. Cruz-Neira, VR Juggler: A Virtual Platform for Virtual Reality Application Development, Proceedings of Virtual Reality (2001) 89–96.

[29] S. Takemura, T. Furumura, T. Saito, Distortion of the Apparent s-wave Radiation Pattern in the High-frequency Wavefield: Tottori-ken Seibu, Japan, Earthquake of 2000, Geophysical Journal International 178 (2) (2009) 950–961, doi:10.1111/j.1365-246x.2009.04210.x.

[30] Y. J. Suzuki, T. Koyaguchi, 3D Numerical Simulation of Volcanic Eruption Clouds During the 2011 Shinmoe-dake Eruptions, Earth Planets Space 65 (6) (2013) 581–589.

[31] T. Lewiner, VSVR: A Very-simple Volume-rendering implementation with 3D Textures, Preprint No. MAT. 16/06, Department of Mathematics, PUC - Rio de Janeiro, Brazil.

[32] T. Miyagoshi, A. Kageyama, T. Sato, Zonal Flow Formation in the Earth's Core, Nature 463 (7282) (2010) 793–6, doi:10.1038/nature08754.