COSMIC WORM IN THE CAVE: STEERING A HIGH PERFORMANCE COMPUTING APPLICATION FROM A VIRTUAL ENVIRONMENT

Trina M. Roy,
Carolina Cruz-Neira,
Thomas A. DeFanti
Electronic Visualization Laboratory
University of Illinois at Chicago

Abstract

Developing graphical interfaces to steer high performance scientific computations has been a research subject in recent years. Now, computational scientists are starting to use virtual reality environments to explore the results of their simulations. In most cases, the virtual reality environment acts on precomputed data; however, the use of virtual reality environments for the dynamic steering of distributed scientific simulations is a growing area of research. We present in this paper the initial design and implementation of a distributed system that uses our virtual reality environment, the CAVE, to control and steer scientific simulations being computed on remote supercomputers. We discuss some of the more relevant features of virtual reality interfaces, emphasizing those of the CAVE, describe the distributed system developed and present a scientific application, the Cosmic Worm, that makes extensive use of the distributed system.

1. Introduction

Visualization and virtual reality prove to be powerful tools for the exploration of scientific data. This scientific data is usually obtained from the execution of large and complex simulation programs that require a great deal of resources such as massively parallel computers, large memories and massive data stores. In most cases, the visualization is performed after the simulation has completed its execution. The dataset resulting from the simulation is stored and displayed in a virtual reality environment using a high-performance graphics workstation for rendering and manipulation. This process allows users to explore precomputed data, revealing some properties of that particular dataset, but it limits the interaction with the scientific problem as a whole. Several examples can be found in [3]. But scientists not only want to explore precomputed data, they want to steer their calculations in as close-to-real-time as possible. With high-speed networks and powerful graphics workstations, direct interaction with scientific simulations running on massively parallel computers is becoming possible. Researchers can now dynamically modify computations while they are occurring, and are able to see the effects graphically almost immediately. Scientific visualization applications like [14][16] are good examples of interactive visualization systems offering these capabilities.

An extension to the graphical interface model on a workstation is to integrate a virtual reality environment into the distributed system. This unique combination is starting to be explored and is already in demand by the high-performance computing and communications community. Pioneer work in this field is currently under development by [1][9][12].

In the following sections we present our research on the use of the CAVE virtual reality environment to steer computationally intensive simulations running on a remote supercomputer. We describe the distributed implementation followed by a discussion of the most developed of the test applications and the results obtained.

2. Virtual Reality Environments and High Performance Computing and Communications (HPCC)

Virtual reality is a relatively new approach to user interfaces [8]. It integrates a variety of input/output and display devices to give users the illusion of being immersed in a computer-generated world. By combining hardware devices and software techniques, we can make the virtual worlds react naturally as users interact with the environment. Although users are not "fooled", the illusion of "being there" is convincing enough that they get immediately involved in the exploration of their virtual surroundings. In this section we describe the features of virtual environments and the particular features of the CAVE, emphasizing the features required to achieve a successful integration within HPCC.

2.1. Virtual Environments Features and Constraints

Virtual environments provide stereoscopic head-tracked displays that allow users to unequivocally perceive three-dimensional structures of the data being analyzed. This unique feature has been shown to be extremely useful to researchers who need to control, drive or interact with their scientific data [17]. Creating a convincing sense of immersion demands very high degrees of performance at all levels of hardware and software used. From experience, the VR community accepts a minimum display rate of 10 frames per second in order to deliver an immersive experience to users [2][6]. To achieve this frame rate, we must consider not only the rendering tasks, but also tasks like data computation, reading tracker information, processing user input and generating audio cues. The accomplishment of all these tasks within the acceptable frame rate places strong constraints at the design stage of a virtual reality application. The types of design compromises needed to maintain the required frame rate are:

- Reduction of the size of the datasets used in the simulation.
- Simplification of the calculations by approximating theoretical algorithms.
- Decreasing the quality and detail of the rendered scene: fewer polygons, simpler lighting models and use of wireframe representations.

One of the main reasons for these compromises is that most virtual reality systems are self-contained [11]. That is, a single high-performance workstation is often used to control the input devices, perform the calculations, store the data and render the scene. A distributed system, where these tasks can be performed by several machines, is a more desirable environment. However, although a distributed system can increase the computational power and provide massive storage to the virtual reality environment, it creates new constraints. A distributed virtual reality system requires networks with high bandwidths which are able to move large amounts of data fast enough to work within the minimally acceptable display frame rate. Unfortunately, such networks are not yet available for general use.

Therefore, the distributed virtual reality system has to be designed keeping in mind that less-than-real-time situations are bound to happen, and that the system should be able to perform under such circumstances.

2.2. Virtual Reality Interface: The CAVE

At the Electronic Visualization Laboratory, we have developed our own virtual reality environment called the CAVE [4]. The goals that motivated the design of the CAVE were to produce a virtual reality environment that was suitable for scientific research and to provide a user interface to steer high performance computing applications running on remote supercomputers. The CAVE produces the sense of immersion by surrounding a user with wrap-around screens on which images are rear-projected in stereo on the walls and down-projected in stereo onto the floor. To perceive stereo users wear three-dimensional shutter glasses. Multiple users can step into the CAVE and share an experience, although only one of them is tracked, and therefore controls the viewpoint for everybody. A wand with a six-degree-of- freedom tracker and several buttons is used to interact with the virtual scene. Integrated in a high performance computing environment, the CAVE is used to steer applications running on a remote supercomputer. In general, steering a remote application involves actions such as:

- Navigating through the data and finding interesting areas to explore in the CAVE.
- Request an enhanced rendering from the remote machine (i.e., successive refinement).
- Calculate a new simulation step on the remote machine to be sent to the CAVE.
- Modify the data in the CAVE and send the data back to the remote machine.
- Run the simulation with a change in the parameters.
- Continuously inform the supercomputer of the activities performed in the CAVE.

Due to restricted network bandwidth, the processing of user actions that require data exchange between the CAVE and the remote computer will result in the loss of real- time performance in the CAVE. In these less-than-real-time situations, the CAVE works better than any head-mounted display (HMD) because the projection planes are the actual walls of the CAVE and obviously do not move as the user moves, whereas an HMD's projection plane is always at a fixed distance and position with respect to a userÕs eyes. This presents a fundamental difference between the two systems because simple head movements, like rotations and nodding, require rendering the same scene with only an adjustment to the stereo perspective in the CAVE, but they require an HMD to regenerate the scene according to the new line of sight. Therefore, when real-time display is lost in the CAVE, users can still see the scene projected on the surrounding walls of the CAVE although they may not perceive the correct stereo until the data exchange is finished In an HMD the image will freeze in front of the userÕs eyes and then ÒpopÓ to its correct location when the data exchange is complete. A more detailed discussion can be found in [5].

3. Distributed System

The CAVE environment is controlled by a Silicon GraphicsÕ (SGI) Onyx machine with 4 processors running at 150 MHz, 512 Mbytes of physical memory, and three Reality Engines2 (RE2). Each one of the Reality Engines is connected to one of the CAVE walls. The communication with the tracking device, an Ascension Flock of Birds with two sensors, is done through an RS232 serial line at a rate of approximately 50 Hz. For the audio system, MIDI commands are send via an RS422 port. The CAVE is connected via a FDDI-to-HIPPI network to a Thinking MachinesÕ Connection Machine-5 (CM-5) with

512 nodes and 32 Mbytes of physical memory per node.

In order to maintain a display rate of 10 frames/second, the tasks needed to operate the CAVE have been separated into several parallel processes communicating via shared memory. According to Figure 1, these processes are:

- Rendering of the scene: generation of the stereo pairs corresponding to each wall.
- Tracker reading.
- Audio process.
- Application or computation process.
- Network communication process.
- Control and synchronization process.

Figure 1. CAVE Processes diagram.

These processes have varying performance rates and therefore have been designed to run asynchronously. This allows a user to explore the data at a frame rate determined by the performance of the tracking and rendering processes without being constrained by the network performance and/or the rate at which the data is being updated by the remote simulation. We split up the processes in this manner in order to maintain the minimum display frame rate of 10 frames/second required to deliver an immersive experience. This technique is particularly valuable while communicating with the remote supercomputer since network delays will not have a detrimental effect on the CAVEÕs rendering performance.

To communicate between the CAVE and the CM-5, we use the Data Transfer Mechanism (DTM) Library developed at the National Center for Supercomputing Applications (NCSA) [7]. DTM is a message-passing library designed to simplify the task of interprocess communication and to facilitate the creation of sophisticated distributed applications in a heterogeneous computing environment. To accomplish this, DTM provides a method of interconnecting applications at run-time and reliable message-passing complete with synchronization and transparent data conversion. DTM has been optimized for large messages (100 Kbytes and up), but is also efficient for smaller messages. DTM was particularly useful in the development stages of this project because our network hardware passed from Ethernet to FDDI to HIPPI. All these changes were handled internally by DTM; therefore, we did not have to make any changes in our code. A set of C++ style classes has been developed on top of DTM to facilitate the organization of the data being transferred. We use a client-server model, with the CAVE acting as the client of the simulation running on the CM-5. In the following section we describe the Cosmic Worm, one of the applications that, since the early stages of the project, has been making extensive use of the distributed system described here. The Cosmic Worm, a collaboration between the authors and scientists at NCSA, visualizes scientific data computed by simulations running on a remote CM-5.

4. The Cosmic Worm

Scientists use the Cosmic Worm to study certain types of cosmic behavior seen in astrophysical objects such as supernovae and black holes. In particular they are observing the effects of a heavy fluid lying on top of a lighter fluid, a process known as the Rayleigh-Taylor Instability, and are studying gravitational wave components predicted by Einstein’s Theory of General Relativity.

Simulations of cosmic phenomena, such as Rayleigh-Taylor instabilities and the nature of gravitational waves, are highly complex, computationally intensive and generate massive amounts of data. In an effort to manage this complexity and interactively visualize this data, we are developing this visualization package taking advantage of the CAVE's high performance computing environment.

4.1. Visualization

The Cosmic Worm [10] (or simply Worm, named after the theoretical stellar object called a wormhole) uses the CAVE to visualize Rayleigh-Taylor and gravitational wave data computed by simulations running on a remote CM-5. These simulations produce a series of time steps, each representing one stage in the progression of the simulation over time. From each time step, Worm extracts isosurfaces based on certain physical parameters (i.e. density value) to be displayed in the CAVE. To study different parameters, several isosurfaces are superimposed as opaque and semi- transparent surfaces. Different colors are used to distinguish the parameter values. As Worm receives new datasets from the simulation, new surfaces are generated to reflect the changing state of the system being modeled.

Figure 2 shows two stages of the evolution of a Rayleigh-Taylor simulation. As the simulation evolves over time, the gravitational force causes the heavier liquid to form fluid fingers that flow down into the lighter liquid, causing mixing and turbulence. This behavior is seen in many astrophysical objects, such as the atmospheres of some stars and the remnants of supernovae.

Figure 3 shows a time step in a gravitational wave simulation. The existence of gravitational waves is predicted by Einstein's Theory of General Relativity; however, they have yet to be measured directly. The theory explains that these types of waves could be generated by violent cosmic events, such as a supernova or the collision of two black holes. Solving Einstein's equations for a three-dimensional gravitational field yields gravitational wave components that can be visualized as they propagate through space-time.

4.1.1 Implementation Details

Figure 4 shows the configuration and flow of data through the system. During Worm's initialization, a separate process, the "receive process," is spawned to monitor an input port for incoming datasets from the CM-5. These are processed one at a time in a pipeline manner. The simulation calculates time step x+1 while Worm computes an isosurface for time step x and the user is seeing a visualization of time step x-1 in the CAVE. The speed at which a time step gets processed is highly dependent on the problem size and the complexity of the isosurface.

The isosurfaces are calculated on the SGI Onyx using a parallelized version of the Marching Cubes algorithm [13]. Attempts were made at writing a massively parallel Marching Cubes for the CM-5 but the initial results showed a significant slowdown compared to even a single processor SGI due to the large amount of inter-CPU communication and synchronization required by the algorithm [15].

Figure 4. Cosmic Worm Configuration

Interaction with the remote simulation is accomplished via a menu system invoked with the wand. The available control options are described in Table 1. The simulation can be stopped or paused at any time during a run; the user sees a visual representation of the most recent time step received from the

simulation, and can also examine accumulated time steps. Additionally, if the simulation has been stopped, the user can modify the simulation input parameters from the CAVE using wand-controlled sliders. The changed parameters are sent to the CM-5 when the user selects the restart option from the menu. The simulation will restart computation at the initial time step using the new parameters. Each control option selected from the menu generates a control message that is sent to the CM-5 directly from the Worm application process.

Option	Description
Start	Initial start command.
Stop	Terminate a running simulation.
Pause	Pause a running simulation for review of data.
Continue	Continue paused simulation.
Restart	Restart stopped simulation with new input parameters.

Table 1. Simulation Control Options

4.2. Advantages in using the CAVE

Worm has shown to be an excellent application for virtual reality given the complex three-dimensional structures of the data generated by the simulation. These structures are not well known to the scientists who, in order to gain a better understanding of the phenomena being studied, need a visualization environment that allows them to truly perceive and interact with the three-dimensionality of the objects. The CAVE offers a unique perspective by immersing the scientists in the visualization and giving intuitive control for navigation through the system and exploration of internal complexities. Furthermore, because the CAVE allows multiple users to experience a visualization at the same time, scientists are able to discuss discoveries in groups, an important aspect of scientific research.

Using the CAVE in a distributed environment allows for a higher level of detail in the visualizations because of the ability to interactively steer the remote simulations. Without interactive steering, the data must be saved to disk for later display in the CAVE. The amount of data generated by these simulations runs into gigabytes (see Table 1), which prevents the storage of every time step computed. Therefore, the simulations only save periodic time steps, resulting in a significant loss of relevant data. With interactive steering from the CAVE, Worm allows a scientist to study every stage of the evolution by visualizing each time step as it is calculated on the remote machine. With close-to-real-time interaction the user can pause the simulation to study the complexities of a particularly interesting time step or restart the simulation with different parameters. This is the first time that our collaborators have had the ability to interact with their three-dimensional simulations.

4.3. Computational Requirements

Table 2 gives the computational requirements for both simulations. In all cases the memory and computation requirements preclude the use of workstations. Computation times are given for both the CM-5 and a uni-processor workstation (essentially a one-node CM-5) for comparison purposes.

Table 2. Computational Requirements

The problem sizes for each simulation vary depending on the requirements for interactivity and depth of

information. A large run (greater than 2003) is the most useful for acquiring in-depth information about the system, but cannot be controlled interactively due to the amount of data generated and the lengthy computation time. Reducing the problem size produces smaller volumes of data which can be handled interactively, but will not contain as much quantitative information. Studying smaller volumes of data, scientists can only get a general idea about how the data is evolving but they cannot gain any concrete knowledge.

One technique that we use to visualize larger datasets is subsampling: Instead of using the entire volume of data for calculating the isosurface, periodic values (i.e. every second or third value) are extracted resulting in a coarser yet more manageable dataset. This also speeds up rendering since fewer polygons are generated for the isosurface. Unfortunately, the drawback to subsampling is similar to that of reducing the problem size: it limits the amount of quantitative information that can be gleaned from the visualization. However, the information lost is of a different nature. Specifically, subsampling distributes the loss of data giving scientists a general idea about the entire volume, where problem size reduction eliminates a large part of the data showing only a portion of the volume at full resolution. Worm has a subvoluming capability that allows the user to select a portion of a large subsampled dataset for visualization at full resolution; however, interaction is still restricted by the computation time required to generate the data.

5. Conclusions and Future Work

We describe in this paper the initial implementation of a distributed system for steering remote simulations using a virtual reality environment. We have been successful in demonstrating that a distributed virtual reality system can be integrated in a high performance computing environment with the current technology. Scientific simulations can now be interactively controlled from the CAVE, giving scientists a new and unique environment for exploring their research data.

In the case of the Worm application, the size and computational complexity of the simulations require the use of high-performance computing resources. Until now this has constrained the visualization to small datasets previously computed on the CM-5 and transferred to the graphics workstation for display. Now, scientists have close-to-real-time control over the simulation on the remote machine; As they interact with the data being displayed in the CAVE, they are capable of steering the remote simulation. As a result, scientists are released from the burden of having to pre-compute and store large amounts of data for later playback in a graphics environment. Scientists are also able to see dynamical three-dimensional structures in their data, which they never observed before.

Further research should be done into many of the areas involved in combining virtual reality and supercomputer simulations. In particular, we are interested in applying the techniques presented here to other types of applications. We think applications with any of the following characteristics would be well suited for this environment:

- Computationally intensive simulations.
- Applications requiring massive data (input and/or output)
- Computationally intensive graphics.
- Continuous communication between the graphics and the remote computing machine.

The initial work has been done on the CM-5 due to availability, but our design methodology is not restricted to that particular architecture. At the time of writing this paper, we are working on extending

this distributed model to work with other massively parallel architectures; we are experimenting with shared memory machines, such as the Silicon Graphics Challenge Array and with other distributed memory computers such as the IBM SP1 and SP2. Our goal is to be able to create a methodology to communicate between the CAVE and remote machines that is independent of the hardware used.

6. Acknowledgments

We would like to thank the people that contributed to this project: Gary Lindahl of the Electronic Visualization Laboratory (EVL) for his technical support; Jeff Terstriep and Daniel Weber of NCSA for providing us with the DTM library and kindly agreeing to modify it to include our requirements; and Jon Goldman of EVL for developing part of the application code. Also thanks to Ed Seidel, Joan Masso, Greg Bryan and Mike Norman of NCSA for their ongoing help with putting their astrophysical codes into the CAVE environment. Our special thanks to Maxine Brown for her constant encouragement.

7. Bibliography

[1] Bryson, S. and Gerald-Yamasaki, M. The Distributed Virtual Windtunnel. Proceedings of Supercomputing Ô92. Minneapolis, MN. November 1992.

[2] Bryson, S. Survey of Virtual Environment Technologies and Techniques. SIGGRAPH ’92 Course Notes #9. July 1992. pp. 1.1-1.35.

[3] Cruz-Neira, C. et al. Scientists in Wonderland: A Report on Visualization Applications in the CAVE Virtual Reality Environment. IEEE 1993 Symposium on Research Frontiers in Virtual Reality. Visualization Ô93. San Jose, CA, October 1993. pp. 59-66.

[4] Cruz-Neira, C. et al. The CAVE Audio Visual Experience Auto Virtual Environment. Communications of the ACM. June 1992. pp. 65-72.

[5] Cruz-Neira, C., Sandin, D.J. and DeFanti, T.A. Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE. ACM SIGGRAPH Ô93 Proceedings. Anaheim, CA, August 1993. pp. 135-142.

[6] Cruz-Neira, C. Virtual Reality Overview. SIGGRAPH Ô93 Course Notes #23. July 1993. pp. 1-18.

[7] Data Transfer Mechanism UserÕs Guide. National Center for Supercomputing Applications. University of Illinois at Urbana-Champaign. 1994.

[8] Ellis, S.R. Nature and Origins of Virtual Environments: A Bibliographical Essay. Computing Systems in Engineering, Vol. 2 No. 4., 1991. pp. 321-347.

[9] Faigle, C. et al. Integrating Virtual Environments with High Performance Computing. IEEE Virtual Reality Annual International Symposium (VRAIS Ô93). Seattle, WA, September 1993. pp. 62-68.

[10] Goldman, J. and Roy, T. Cosmic Worm. IEEE Computer Graphics & Applications. To be published. July 1994.

[11] Hitchner, L. Virtual Planetary Exploration: A Very Large Virtual Environment. SIGGRAPH Ô92 Course Notes #9. July 1992. pp. 6.1-6.16.

[12] Leigh, J. et. al. Realistic Modeling of Brain Structures with Remote Interaction Between Simulations of an Inferior Olivary Neuron and a Cerebellar Purkinje Cell. 1993 Simulation Multiconference on the High Performance Computing Symposium. Arlington, VA. March 1993. pp. 248-253.

[13] Lorensen, C. and Cline, H. A High Resolution 3D Surface Construction Algorithm, Computer Graphics, Vol. 21. No. 4. July 1987. pp. 163-169..

[14] Moran, P. Tele-Nicer-Slicer-Dicer: A New Tool for the Visualization of Large Volumetric Data. Technical Report #014. National Center of Supercomputing Applications. University of Illinois at Urbana-Champaign. August 1993.

[15] Paul, B. Personal communication. Space Science and Engineering Center, University of Wisconsin, Madison. February 1994.

[16] Pepke, E. and Langsley, R. SciAn Visualization Package. Personal communication. Supercomputer Computations Research Institute, Florida State University. January 1994.

[17] Song, D. and Norman, M.L. Cosmic Explorer: A Virtual Reality Environment for Exploring Cosmic Data. IEEE 1993 Symposium on Research Frontiers in Virtual Reality. Visualization Ô93. San Jose, CA, October 1993. pp 75-79.

Carolina Cruz-Neira
carolina@evl.eecs.uic.edu