NETWORKED VIRTUAL REALITY FOR REAL-TIME 4D NAVIGATION OF ASTROPHYSICAL TURBULENCE DATA

Randy Hudson
Academic Computing Services

Andrea Malagoli
Enrico Fermi Institute/
Laboratory for Astrophysics and Space Research

University of Chicago,
Chicago, IL 60637
email: rhudson@midway.uchicago.edu

ABSTRACT

This paper describes a prototype distributed virtual reality system for real-time spatial and temporal navigation of volumetric data generated by simulations of hydrodynamic turbulence on high-performance computers. Our system makes use of a virtual environment connected to a scalably parallel computer via a scalably high-speed network. The data are either computed in real-time or precomputed on the parallel computer, then are transferred to the virtual environment where fast volume visualization is accomplished by using sophisticated three-dimensional texture-mapping hardware.

Keywords: Virtual reality, parallel computing, volume rendering, high-speed networking, distributed processing

1 INTRODUCTION

1.1 Background

One of the long-range goals of the Academic Computing Services (ACS) department at the University of Chicago (UC) is to enhance UC researchers' ability to do computational science by giving them improved access from the UC campus to both remote supercomputing technology and remote colleagues, and also access to high-end virtual reality (VR) and visualization technology.

This goal will be realized via

- VR-assisted remote steering of supercomputer simulations;
- high-quality video teleconferencing;
- enhanced remote storage and retrieval of massive amounts of simulation data, video conferences and virtual experiences; and
- "collaboratories": persistent virtual spaces where researchers can meet, and use all the above tools as well as remote lab hardware (Stevens and Evard, 1994).

A first step in this direction will be enhanced access by the UC research community to the leading-edge computing technology at Argonne National Laboratory (ANL). Such technology includes ANL's IBM SP scalable parallel supercomputer, their CAVE^{TM} (Cruz-Neira et al., 1992; Cruz-Neira et al., 1993) and ImmersaDesk^{TM}(Korab, 1995; Korab and Brown, 1995) virtual environments and their Multimedia Laboratory's parallel multimedia server.

Using such new technology as scalable, high-bandwidth asynchronous transfer mode (ATM) networks and hardware-assisted, real-time volume rendering of volumetric scientific data, ACS is cooperating with ANL and various research departments at UC to develop a distributed prototype system for remote retrieval and visualization of live or archived volumetric data in a virtual environment. This system will be usable with computations that UC researchers are already doing on ANL computers. The first application of the system, the visualization of astrophysical turbulence simulation data, is such a computation.

1.2 System Components

The prototype system we describe in this paper uses the distributed processing paradigm to implement real-time navigation and animation in a virtual environment of volumetric astrophysical data produced by a computationally expensive simulation. Our distributed VR system is implemented on the IBM SP and the CAVE virtual environment at ANL. We will discuss the system in four parts:

The simulation. This is a Grand Challenge supercomputer simulation of astrophysical turbulence that is run on the SP and archived on the IBM NSL Unitree system.

The high-speed network. This transfers the data from the SP to the CAVE rapidly enough for real-time interaction with the data.

The distributed data server. This is based on the Message Passing Interface (MPI) standard (Gropp et al., 1994) and runs between the SP and the CAVE.

The VR visualization component. This runs in the CAVE and uses the specialized hardware of the Silicon Graphics, Inc. (SGI), ONYX graphics parallel computer which drives the CAVE, to display the data.

Our use of MPI makes our system portable to other distributed environments. Our use of the CAVE software library makes our system portable to a set of VR systems (§5.1) that range from high-end immersive virtual environments to inexpensive "fishtank" (standard workstation) VR stations that are already being used on the UC campus. (Since the system uses either a CAVE or an ImmersaDesk as the virtual environment, we usually will refer to these generically as the "CAVE".)

We describe each of these four parts in more detail below.

2 SIMULATION

The data visualized by our system is a large, 4D scalar field—a time series of volumes—generated by a Grand Challenge simulation of self-gravitating astrophysical hydrodynamics developed by one of the authors. It consists of 300 time snapshots (figure 1), each one a volume of 128^3 scalar volume elements (voxels). The computation simulates an interstellar gas collapsing under the influence of its own gravitational field to form a compact spherical object. The mechanism for this process, know as "Jeans instability", is the fundamental mechanism by which intergalactic gases condense to form stars. The instability is triggered when the gas's internal pressure gradients cannot balance the gravitational force generated by the gas mass. The data actually represent the evolution of the mass density of the gas. Since this particular simulation cannot produce adequate quantities of data fast enough to be visualized live in the CAVE, we visualize a large, archived time series. (Many other simulations will be steerable from the CAVE as they run.) The computations have been carried out on 32 nodes of ANL's SP.

The numerical code used for the simulation combines an explicit high-resolution method for compressible hydrodynamics with a linear multigrid scheme to compute the gravitational field. The data are stored on the Unitree system connected to the SP for later visualization and analysis, but currently, before archiving, they are subsampled to a resolution of 64^3 voxels and converted from floating-point values to eight-bit integers to make them small enough for processing in the CAVE (§5 and §6) and for transferring over the network quickly enough (§5.3). The problem domain is periodic, so what goes out one side of the computational box comes in from the other side. Due to the nonlinearity of the governing Navier-Stokes equations, the gas motions show complex behavior. Both chaotic and coherent structures are formed, which coexist in time and space.

3 MREN

The network component of our system is the Metropolitan Research and Education Network (MREN)[†], an advanced, high-bandwidth ATM network being developed by UC in partnership with ANL,

[†]URL: http://www.uchicago.edu/ns/mren.html

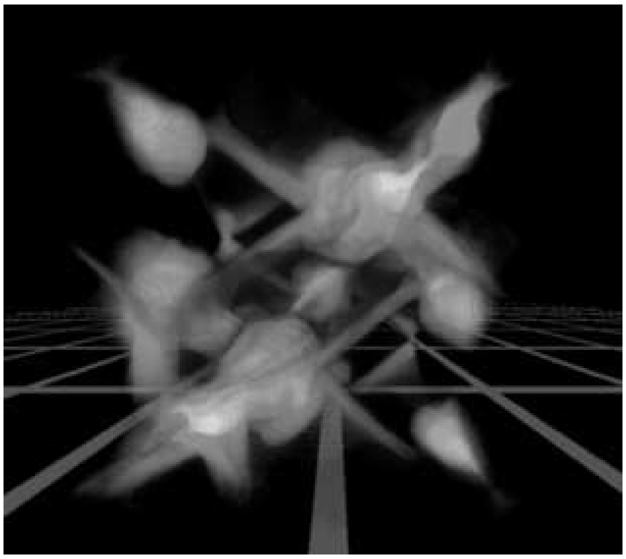

Figure 1: CAVE volume rendering of a gas collapsing under the influence of its own gravitational field to form a star

Ameritech and several other universities and national research centers, and serving several research institutions in the Chicago area. The communications backbone infrastructure for MREN is a SONET-based OC-48 (2.4 gigabits per second) fiber ring.

The university's high-speed campus network, which is dedicated to high-end research applications, is currently connected to MREN via a DS-3 (34 megabits per second (Mb/s)) link. This link will be upgraded to OC-3 (155 Mb/s) in the near future. ANL, including the IBM SP, CAVE and ImmersaDesk, is connected to MREN at OC-3. Thus, UC researchers will soon have OC-3 access to ANL technology. Other Chicago-area research institutions connected to MREN are Fermi National Accelerator Laboratory, the University of Illinois at Chicago and Northwestern University.

ATM is the best technology for such a network because it can support all the different types of network traffic that will be produced by the variety of applications described in §1.1 and because this support is scalable to gigabit speeds and beyond.

4 DISTRIBUTED DATA SERVER

The distributed data server software runs between the SP and the CAVE. It uses a master-slave paradigm and a Nexus-based (Foster et al., 1994) MPI implementation. The master process is also the controlling CAVE process (§5.1), which requests a sequence of data frames (representing a sequence of time snapshots) when the user decides to play the time series forward or backward. The data server fetches one data frame at a time from the SP to satisfy the request. It can run in three modes. The primary mode, for remote visualization, reads frames from the remote archive each time one is requested by the CAVE. The other two are secondary modes that allow local visualization: one reads frames from local disk as they are requested; the other reads them from local memory as they are requested (after initially being loaded during start up).

5 VR VOLUME VISUALIZATION

The astrophysical simulation dataset is displayed in the CAVE with a general-purpose volume visualization tool, which is being developed by one of the authors.

5.1 Virtual Environment

The CAVE, developed at the Electronic Visualization Laboratory (EVL) of the University of Illinois at Chicago, is a fully immersive, projector-based VR system driven by an SGI ONYX graphics multiprocessor. Rear-projection display screens form a ten-foot-cubed room that users walk into. Stereographic pairs of computer-generated images projected to the walls and floor of the CAVE are integrated with one another to produce a seamless, 3D virtual space in which to do research. A typical CAVE application is a parallel program consisting of a controlling process and a separate drawing process for each CAVE wall. Any CAVE application is immediately portable in untracked, monoscopic simulator mode to an SGI workstation, and in tracked, stereographic mode to the ImmersaDesk and the *Infinity Wall*. The ImmersaDesk is a one-walled, portable version of the CAVE also developed at EVL. The Infinity Wall is a large, one-walled version of the CAVE developed by EVL, in collaboration with the University of Minnesota and the National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign, with support from SGI.

5.2 Volume Rendering

Volume visualization encompasses a set of techniques for displaying all or part of a 3D volume of scalar data (a "3D scalar field" or "3D image"). Of the two main methods of volume visualization—isosurface modeling and (direct) volume rendering (Drebin et al., 1988)—our visualization software currently does the latter. Volume rendering usually displays a 3D scalar field as a translucent substance of varying opacity, making it possible to view the entire 3D field at once. In general, volume rendering is a time-consuming technique.

For VR users to believe in the reality, in some sense, of the virtual scene being displayed, the scene and their interaction with it must be updated at a rate of at least 10 frames per second (fps). Little volume rendering has been done in virtual environments until recently, partly because the time to render a single time step (animation frame) has been too great.

5.3 Alternatives

There are two main ways to get from volumetric data that are generated or stored on the SP to a volume-rendered display of the data in the CAVE:

1. The data can be sent over the network to the CAVE, where two stereo images can be volume-rendered for each CAVE wall;

2. The data can be volume-rendered into a stereo pair of images on the SP, and the images sent over the network for display in the CAVE (§6).

In the current version of our system, we perform volume rendering on the ONYX as in option 1.

To develop our prototype, we are using the OC-3 ATM link between the SP and CAVE at ANL, where we have attained a better than 100 Mb/s effective transfer rate. If we wish to transfer and display 10 fps, we need to fit one frame of data into about 10 Mb. The largest *cubic* volume that meets this requirement is 109 elements on a side and one byte per voxel. Because of ONYX hardware constraints (§5.5), our data need to be converted to eight bits per voxel and subsampled to 64^3 voxels. As mentioned above (§2), these tasks are done on the SP before archiving the data.

5.4 Texture Mapping

Texture mapping (Catmull, 1974; Blinn and Newell, 1976), with which we implement volume rendering, is a way to map the pixels of a (2D or 3D) image onto some computer-graphical primitive, such as a polygon. For example, in 2D texture mapping, the pixels of a 2D image—the texture—are drawn onto the corresponding pixels of a polygon. In 3D texture mapping, the texture is a 3D image, such as a single time step of our simulation. Here, if a polygon intersects the 3D image, its pixels assume the values of the image that lie in the plane of intersection.

5.5 VR Display Software

Our volume visualization software uses the ONYX's 3D texturing hardware to do volume rendering in much the same way as does SGI's *volren*[‡] application, and as described in (Cabral et al., 1994; Teschner, 1994). The texture mapping hardware of the ONYX at ANL currently limits the size of a single 3D texture to 2 MB and requires the image resolution in each dimension to be 2^n. At a resolution of 128^3 bytes, the simulation data would require just 2 MB. But, this amount of data prevents us from attaining the frame rate we want (§5.3). (In reality, the hardware "texture definition" process that must occur on the ONYX with each new data frame currently allows about 3 fps with the 64^3-voxel size, but newly available hardware is expected to improve this.) Also, if we wish to use mipmapping (Williams, 1983), some texture memory is required for overhead, leaving less than 2 MB for a single 3D texture. For the prototype, half-resolution in each dimension is acceptable. So, to allow room for mipmapping and in order to get 10 fps while retaining a cubic data volume, the data are subsampled and converted as described above (§2 and 5.3).

The volume of simulation data defines a discrete scalar function over a rectangular interval of 3D space. Treating this function as a 3D texture, the texturing hardware trilinearly interpolates between the discrete values to reconstruct it as a continuous function. We then sample this new function with a stack of from 100 to 200 regularly spaced slicing polygons perpendicular to the user's line of sight. Parts of the polygons, in general, extend beyond the faces of the data volume

[‡] "Interactive Volume Rendering Using Advanced Graphics Architectures," *Online Technical Center white paper—URL: http://www.sgi.com/Technology/volume/VolumeRendering.html,* Silicon Graphics, Inc.

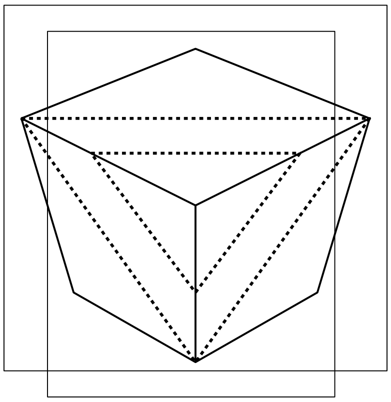

Figure 2: Slicing polygons at an arbitrary orientation with respect to a data volume

into empty space and are clipped away. Figure 2 shows a data volume, two slicing rectangles, and the resultant clipped slicing triangles. For all polygons in the stack, the texturing process is performed on the parts that remain: a given polygon's pixels take on the values of the 3D field that lie in the slice of the field that is coincident with the polygon.

The polygons can be texture mapped with pseudocolors or gray values (brightness) to segment the data into different ranges of mass density condensation. The translucence required for volume rendering is attained by the additional mapping of the data to alpha channel values. Via use of the alpha blending hardware, the slicing polygons are composited into the frame buffer during rendering (Porter and Duff, 1984). The blending allows the entire 3D image to contribute to the final picture. The effect is of a translucent block of substance of varying opacity (figure 1). Greater opacity is used to represent greater mass density.

Users have the choice of a number of interactive capabilities, including full immersion in the data and manipulation of the data's size, orientation and position. Via separate control of transparency and color tables, the user can segment the data by making certain ranges of values visible or invisible. The opacity can be adjusted to vary the display from translucent to opaque, the latter being equivalent to an isosurface display. There are controls for playing the time series backward and forward, single-stepping, directly selecting which frame to display, and loading a subsequence of frames into local RAM for rapid animation.

Results are best if the slicing polygons are kept perpendicular to the user's line of sight. In a virtual environment, where the user can move in world space, the line of sight can have any orientation, and the polygons must be rotated to remain perpendicular to it. Since the polygons can slice the texture volume at any orientation, the vertices of the clipped polygons must in general be recomputed for each animation frame.

6 DISCUSSION

We have described a scalable, distributed prototype system for networked VR display and control of remote archived or live supercomputer simulations. This system, when completed, will allow UC researchers to control and view simulations from the UC campus. It will also be usable with 3D medical data, such as CT scans. Our system also sets the stage for remote access at the higher speeds to which OC-3 ATM is scalable.

An important extension planned for our system is the support of shared virtual environments, using either software being developed at ANL for this purpose or EVL's CAVE library. This will allow several researchers at different locations to collaborate without leaving their places of work. Such shared virtual environments will later be included in collaboratories (§1.1).

We also plan to explore rendering option 2 above (§5.3). A single pair of stereo images rendered on the SP could be displayed in the CAVE by texture mapping them onto two polygons displayed appropriately in front of the user's eyes—a "virtual head-mounted display" (Sutherland, 1965). Over an OC-3 network, two 931×698, eight-bit-per-pixel stereo images could be sent at the desired 10 fps rate, assuming the SP could render them from the dataset rapidly enough. A 3D dataset with a cross section of that resolution—say, $931 \times 698 \times 698$, at eight bits per voxel—would take about 349 times as long to transfer, and quite a bit of time to render on the ONYX as well. Many scientific and medical datasets have resolutions comparable to this.

Many scientific computations produce not only scalar, but vector fields, which also can be volume rendered. Volume rendering of 3D vector fields will be implemented in a future version of our system.

Building on research being done at ANL, another feature that will be added to the current prototype will be support for archiving and retrieving the virtual experiences themselves in ANL's Multimedia Lab.

ACKNOWLEDGMENTS

This project was supported in part by a NASA/ESS HPCC Grand Challenge Award at the University of Chicago, and by Academic Computing Services of the University of Chicago. It was also supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Computational and Technology Research, U.S. Department of Energy, under Contract W-31-109-Eng-38.

The IBM SP system used for this research is part of the Argonne High-Performance Computing Research Facility. The HPCRF is funded principally by the U.S. Department of Energy. The CAVE and ImmersaDesk virtual environments used for this research are part of Argonne's Computing and Communications Infrastructure Futures Laboratory.

References

Blinn, J. and Newell, M. (1976). "Texture and Reflection in Computer Generated Images". *Communications of the Association for Computing Machinery*, 19(10):542–547.

Cabral, B., Cam, N., and Foran, J. (1994). "Accelerated Volume Rendering and Tomographic Reconstruction Using Texture Mapping Hardware". In *Proceedings of the IEEE 1994 Symposium on Volume Visualization*, pages 91–94, Washington, DC.

Catmull, E. (1974). *A Subdivision Algorithm for Computer Display of Curved Surfaces.* PhD thesis, University of Utah.

Cruz-Neira, C., Sandin, D., and DeFanti, T. (1993). "Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE". In *Proceedings of SIGGRAPH '93*, pages 135–142, Anaheim, CA.

Cruz-Neira, C., Sandin, D., DeFanti, T., Kenyon, R., and Hart, J. (1992). "The CAVE: Audio Visual Experience Automatic Virtual Environment". *Communications of the Association for Computing Machinery*, 35(6):65–72.

Drebin, R., Carpenter, L., and Hanrahan, P. (1988). "Volume Rendering". In *Proceedings of SIGGRAPH '88*, pages 65–74, Atlanta, GA.

Foster, I., Kesselman, C., and Tuecke, S. (1994). "The Nexus Task-Parallel Runtime System". In *Proceedings of the 1st International Workshop on Parallel Processing*, pages 457–462.

Gropp, W., Lusk, E., and Skjellum, A. (1994). *Using MPI.* MIT Press, Cambridge, Mass.

Korab, H. (1995). "Next Generation of Internet to Debut". *Access*, 9(2):23.

Korab, H. and Brown, M., editors (1995). *"Virtual Environments and Distributed Computing at SC'95"*. ACM/IEEE Supercomputing '95.

Porter, T. and Duff, T. (1984). "Compositing Digital Images". In *Proceedings of SIGGRAPH '84*, pages 253–259.

Stevens, R. and Evard, R. (1994). "Distributed Collaboratory Experimental Environments Initiative: LabSpace, A National Electronic Laboratory Infrastructure". URL: http://www.mcs.anl.gov/FUTURES_LAB/MULTIMEDIA/RESEARCH/labspace.html.

Sutherland, I. (1965). "The Ultimate Display". In *Proceedings of the 1965 IFIP Congress*, volume 2, pages 506–508.

Teschner, M. (1994). "Texture Mapping". *IRIS Universe*, 29:8–11.

Williams, L. (1983). "Pyramidal Parametrics". In *Computer Graphics (SIGGRAPH '83 Proceedings)*, volume 17, pages 1–11.