Editors:
Larry Rosenblum and Simon Julier

Open-Source Tools for Immersive Environmental Visualization

William R. Sherman, Simon Su, Philip A. McDonald, and Yi Mu
Desert Research Institute

Frederick Harris Jr.
University of Nevada, Reno

Scientists of the Desert Research Institute's atmospheric, hydrological, and earth and ecosystem science divisions sift through vast amounts of data gathered from sophisticated instrumentation and computational models. To help address the data analysis needs that stem from the growing volume of data, DRI established the Center for Advanced Visualization, Computation, and Modeling (CAVCaM) to foster DRI's move into interactive, large-scale, 3D visualization of these and other environmental research issues. CAVCaM currently employs a four-sided, CAVE-like, Fakespace FLEX immersive display system, a Visbox single-screen immersive display, and a 44 CPU with 160 Gbytes of RAM SGI Altix 3700 SMP system.

Our mission is two-pronged. First, we seek to partner with our researcher collaborators at DRI and the larger scientific communities through the development and use of better visualization tools. Second, we work with research sponsors to create visualization applications for training and simulation. For example, in conjunction with research sponsored by the US Army, DRI researchers are investigating the physics behind real-world phenomena so they can create realistic, real-time environmental effects in training and simulation applications. Examples of these phenomena include the behavior of wildfires and of dust generated from various soil conditions, and the flow and effect of atmospheric contaminants on large-scale visibility.

To best achieve these goals, our underlying philosophy is to use established standards and generic tools (ideally open source) and contribute back to the open-source community. Of course, situations might arise when conditions deter us from these goals. For example, collaborators might have established procedures that require specific software systems, and they might be understandably reluctant to abandon them. Or, there might be situations when no open-source product is available that fills the need, such as realistically portraying real-world vegetation. However, these goals serve as our general guidelines and have let us quickly establish a center that is already addressing many clients' needs.

Software tools

Although CAVCaM does not focus exclusively on immersive, or VR, environments, much of our early efforts have required immersive visualization tools. The most desirable solution, of course, is to work with software packages that users can operate immersively, as well as through a desktop interface. In particular, our work with the Vis5D/Cave5D combination meets this goal. Also, some solutions work for the Kitware Visualization ToolKit (VTK), though often with some extra layers that can decrease performance and flexibility. In other cases, we are developing immersive tools that can directly interface with data from existing desktop tools, such as the ESRI Geographic Information Systems software suite.

FreeVR

To ensure that our immersive tools would be not only widely distributable but also widely usable, it is incumbent upon us to build them around a VR integration library that is both open-source and flexible enough to run on a wide spectrum of immersive display systems. FreeVR is one such library.

FreeVR was born at a time when there were few, if any, open-source VR integration libraries that would work both in projection or head-based immersive displays. The primary objective is to put an abstraction layer between the simulated world and the input and output devices. For a VR application to be widely useful, it is vital that it be transportable among immersive displays at facilities beyond the one where the application was created. The abstraction layer's purpose is to allow user-interfaces to work with many physical devices. Thus, FreeVR includes device interface software for many hardware devices commonly found at VR facilities.

FreeVR applications' programming style is not unusual for VR integration libraries. The application programmer specifies the rendering routine and separately specifies the world simulation routine. These routines are executed in independent processes or threads, providing a coarse-grain form of parallelism. This method

Published by the IEEE Computer Society

is the same as that employed by the CAVELib, making it convenient to port many existing VR applications to the FreeVR open-source alternative.

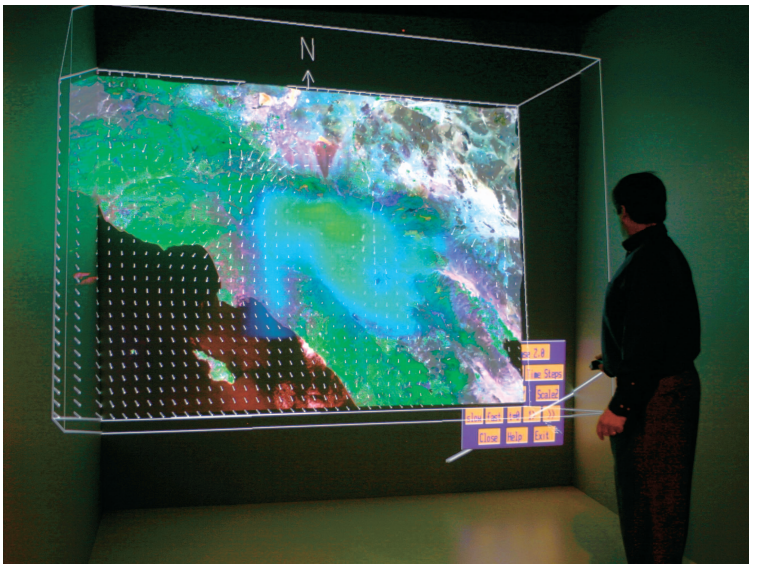
This course-grain parallelism permits some natural efficiencies. However, developers must take care to control state changes in the virtual world such that each synchronized render-frame shows the same data on all screens. This is best accomplished by segregating the world dynamics calculations from the rendering and using data locks to prevent simultaneous access. Many times, developers can integrate applications not originally written for a multiscreen display into FreeVR or other similar libraries and quickly make them work on an immersive display. Applications that do a good job of segregating the rendering from the simulation of the world facilitate such porting considerably. Unfortunately, the authors of many graphical programs do not anticipate multiple, simultaneous rendering operations and, therefore, write in a style that merges simulation and rendering. These programs are often exceedingly difficult to port to a VR environment.

FreeVR also has graphical output drivers written to work with many different systems. FreeVR primarily works with the ubiquitous OpenGL rendering library. Also, some higher-level rendering systems add many features above and beyond what the standard library provides. Specifically, FreeVR has been designed to work with SGI's Performer library, the OpenSceneGraph library, and the OpenSG system. Ports for OpenInventor and the similar Coin3D systems have also progressed, though they have multiprocessing issues.

OpenSceneGraph

In the past, many applications designed for CAVEs and CAVE-like displays have made use of the SGI Performer scene-graph library. In addition to the data structure organization the scene-graph provided, Performer was designed and built to provide synchronized multiple-view displays with efficient, parallel rendering. This design meshed well with VR systems that, in most cases, required complex scenes to display simultaneously on multiple screens. Unfortunately, Performer is a closed-source library, proprietary to SGI. Although SGI has provided ports to some non-SGI systems, there are serious doubts of the library's long-term availability.

In many ways, the open-source OpenSceneGraph library provides many of Performer's qualities. As with the Performer library, OSG divides the overall rendering process into separate application, culling, and rendering operations, allowing parallel pipelining. Unfortunately, as an open-source system, not all of the contributed modules adhere to this design. Therefore, some modules do not work well in a multiview system. For example, the particle system node uses the cull process to update particle movement and, when multiple culling processes are all writing to the same memory, problems ensue. Apart from node modules that depart from the restriction of avoiding writing during culling and rendering, the OSG library works well for many of our applications, especially those that require medium-size terrain rendering.



1 **Visualizing atmospheric aerosols over Los Angeles in Cave5D.**

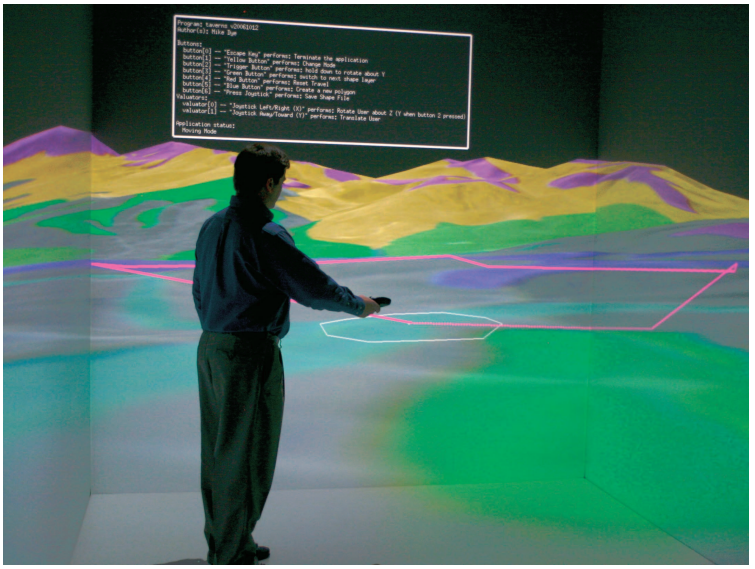
Vis5D/Cave5D

In 1988, the Space Science and Engineering Center at the University of Wisconsin, Madison, released the Vis5D atmospheric visualization tool.[1] In 1994, a VR port of this open-source application was created for the CAVE immersive display system.[2] This port, named "Cave5D," used the latest version of Vis5D available at that time (version 4.0) and the now commercially available CAVELib VR integration library.

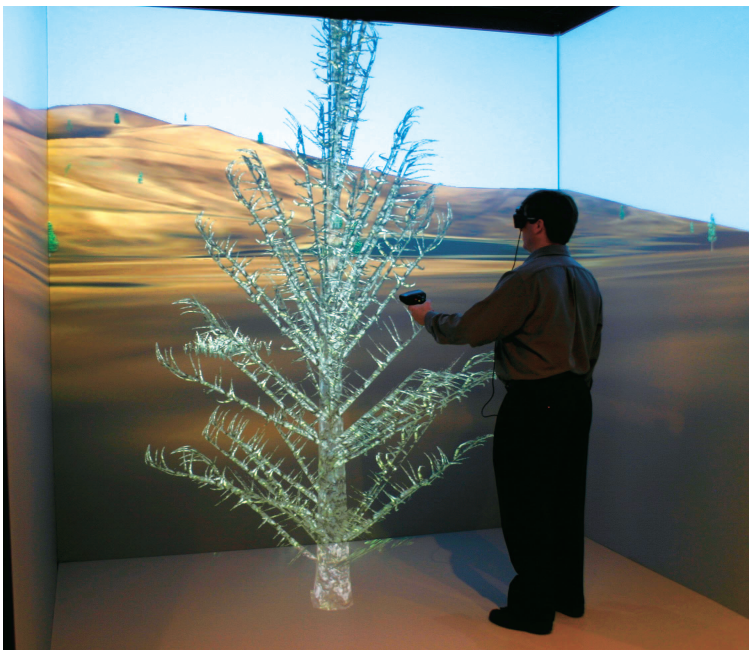
Our effort began in 2005 by taking the latest version of Vis5D (version 1.3.0 beta of what is now called Vis5D+, available from http://sourceforge.net) and integrating it with the latest immersive version (Cave5D 2.0, dating from the late 1990s). At the same time, we integrated this version of Cave5D with the FreeVR open-source VR integration library.

We added an API function to the Vis5D+ core that provides a means of specifying which data management functions are to be used in lieu of the standard `calloc()`, `malloc()`, and `free()` functions. This has eliminated the need to maintain separate versions of the Vis5D+ code—one for Vis5D+ itself, which uses the standard data management functions, and another for Cave5D, which uses shared memory.

The volume visualization technique that Vis5D previously used was that of layered, axis-aligned, texture maps rendered from back to front. Those axis-aligned cross-sections that are most nearly orthogonal to the line of view are the ones the application presented. However, for visualizations in a CAVE-like display, the chosen planes' optimal orientation and ordering differs among the screens. This, in turn, leads to a loss of visual coherence and a high degree of distraction and confusion. We addressed this difficulty by dividing the volume into cubes and, progressing from back to front, rendering the cubes' front faces. Although the chosen faces and cube order might differ among the displays, the overall effect is a consistent family of cubes and maintained visual coherence (see Figure 1).



2 Immersively interacting with Geographic Information Systems layer data.



3 A SpeedTree tree with foliage removed and burned away.

Visualization ToolKit

Another exemplary open-source visualization package that we use is the VTK visualization toolkit from Kitware.[3] VTK uses the dataflow processing paradigm of passing information between modular, conceptual components in information filtering and visualization. VTK can be useful for prototyping visualizations through a desktop interface. The best current solution for rendering VTK on immersive displays is through Paul Rajlich's VTK-to-Performer translator. However, transferring the geometric representation from the internal VTK data into Performer's data structures has a delay. Plus, there is a longer term issue of Performer's future availability.

Ideally, we would like to rely on VTK to do only the rendering and allow FreeVR to handle windowing, and so on. Unfortunately, VTK does not currently provide this option. The rendering in VTK is bound into modules that open windows and create graphic library contexts and also provide the interaction controls. To integrate well with most VR integration libraries, which calculate perspective matrices for each window on the basis of the user's head position, the most desirous situation is for VTK to provide the routine that does all the rendering without modifying the perspective matrix already on the stack—a `vtkRender()` routine. Presently, this does not seem to be a trivial modification to the VTK API.

GIS reader/writer

Much research at DRI involves the relationship between physical models and actual real estate—problems that frequently require Geographic Information Systems (GIS) tools. Most DRI GIS researchers use the ESRI suite of tools. CAVCaM must work with data compatible with these tools to facilitate collaboration with these scientists.

Using publicly available information, we wrote an API to read and write simple layer datafiles, which contain point, line, and polygonal information (see Figure 2). We can render these as their native shapes or create a texture map for draping over Digital Elevation Mapping data.

We wrote our initial interactive GIS reader/writer application using the basic OpenGL library. This version works well so long as the terrain covers only a medium-size region—one in which the texture map for the entire terrain fits within the hardware rendering card's memory. To allow larger spatial coverage, we implemented a second version of the interactive GIS tool for use with the OSG library to take advantage of its terrain paging capabilities.

SpeedTree

For some of our applications, realism is the key ingredient. Because our applications deal primarily with outdoor environments, we sought tools that would render realistic vegetation. What we found and pursued is the SpeedTree vegetation rendering library. SpeedTree uses textures and multiple levels-of-detail methods to rapidly render realistic trees and other vegetation.

SpeedTree is primarily designed as a source of realistic outdoor scenes for large game worlds. SpeedTree is a commercial product and, therefore, is an exception to our precept of building on open-source tools. However, our license does include source code, so although most of the SpeedTree customers use Microsoft Windows or console gaming systems, we were able to implement the rendering system on the Linux systems that drive our immersive displays. For copyright purposes, we must have an alternate, lower-quality vegetation graphical representation to include in general distributions of our work.

Applications

Ultimately, our work aims to produce tools to help our collaborators evaluate their science or train their personnel. In early 2005, we began working on projects with different DRI research groups. In all cases, our software development has focused on the previously men-

tioned tools to help us build a suite of tools that will be of use not only on these applications but also for our future projects.

Wildfire simulation

Wildfires are a frequent summer-time concern for land managers and communities neighboring wild lands throughout the world. We have been working with the DRI Program for Climate, Ecosystem, and Fire Applications to produce a wildfire visualization system to help better understand wildfires and to benefit training, prediction, and policy decisions related to wildfire management.

An important feature for the wildfire visualization is the vegetation found in the terrain that will be the wildfire's fuel source. Our initial experiments with vegetation rendering was with simple X-style billboard trees. However, we wanted to move to a solution that would look realistic even when located very near to the trees or brush. For this, we turned to the SpeedTree vegetation rendering system (see Figure 3). We integrated the SpeedTree renderer with the OpenSceneGraph and FreeVR libraries to bring realistic looking trees into the virtual world. Work remains, however, to increase rendering efficiency and therefore the number of trees that the application can render in real-time. We have also begun incorporating realistic-looking fire and smoke models into the scene.

Atmospheric Visibility Analysis

Atmospheric gaseous and particulate chemicals can impact visibility significantly. The thrust of our work in this area is to take the atmospheric chemistry model's output data and combine it with visibility transfer functions to accurately portray atmospheric pollution's impact on visibility. Atmospheric scientists at DRI have used the Comprehensive Air-quality Model with extensions (CAMx) to study atmospheric chemicals' behavior in a number of cases. We reformatted CAMx output so that it is compatible with Cave5D, which we then use for the visualizations. Figure 4 shows an example of this application's visualization capability.

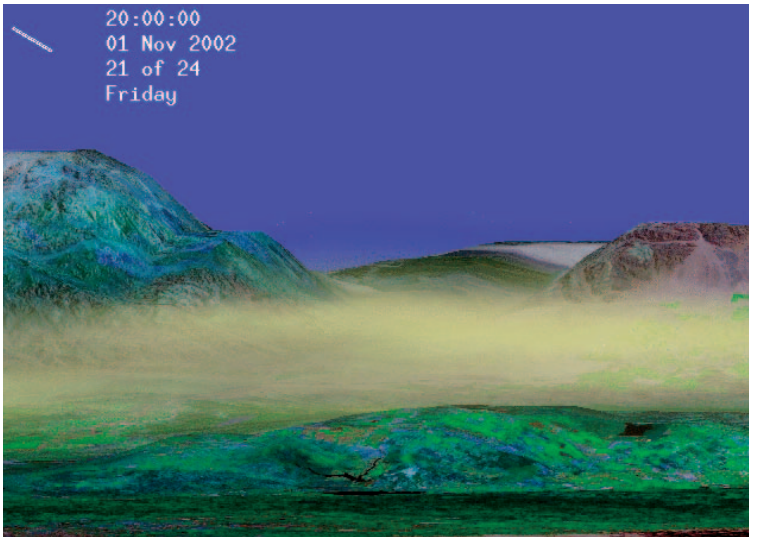
The Future

As we continue to work on these projects, along with new projects that we recently selected, we expect to build up a suite of tools that will be useful for environmental visualization, training, and exploration. Some of our future application work will include visualizing sand dunes' internal structures and immediate forecasting, or nowcasting, with a portable VR system for the user interface. Our goals for future development include increasing the realism of world rendering in real-time, immersive display systems. We are also working toward interfacing our immersive displays with an existing, robust, open-source visualization toolkit, such as SCIRun from the University of Utah or Kitware's VTK.

Conclusion

The Center for Advanced Visualization, Computation, and Modeling is a new lab, having brought online a CAVE immersive display and most of the staff in mid-2005. By starting with open-source tools and extending them to meet our needs, we have been able to bring data from DRI researchers into our CAVE in only a few months.

Our lab is committed to developing open-source tools that will help expand the community of collaborators with whom we can work. We look forward to being able to interface other generic visualization and GIS tools into our framework to let us quickly view data of many other researchers at DRI and elsewhere. ■



4 **Volume rendering in Vis5D showing visibility.**

References

1. W. Hibbard et al., "Exploring Coupled Atmosphere-Ocean Models Using Vis5D," *Int'l J. Supercomputer Applications*, vol. 10, no. 2, 1996, pp. 211-222.
2. C. Cruz-Neira et al., "The CAVE: Audio Visual Experience Automatic Virtual Environment," *Comm. ACM*, vol. 35, no. 6, 1992, pp. 64-72.
3. W. Schroeder, K. Martin, and B. Lorensen, *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*, 3rd ed., Kitware, 2003.

Contact authors William R. Sherman at wsherman@dri.edu, Simon Su at ssu@dri.edu, Philip A. McDonald at philipm@dri.edu, Yi Mu at ymu@dri.edu, and Frederick Harris Jr. at fredh@cse.unr.edu.

Contact the department editors at cga-vr@computer.org.