LIMBO/VTK: A Tool for Rapid Tele-Immersive Visualization

Jason Leigh
jleigh@eecs.uic.edu
National Center for Supercomputing Applications
and the Electronic Visualization Laboratory

Paul J. Rajlich, Robert J. Stein
prajlich@ncsa.uiuc.edu, rstein@ncsa.uiuc.edu
National Center for Supercomputing Applications
University of Illinois at Urbana-Champaign

Andrew E. Johnson, Thomas A. DeFanti
ajohnson@eecs.uic.edu, tom@eecs.uic.edu
Electronic Visualization Laboratory
University of Illinois at Chicago

Abstract

This "Late Breaking Hot Topic Paper" describes LIMBO/VTK a tool that allows developers to quickly build collaborative visualization applications for CAVE, ImmersaDesk as well as desktop workstations. LIMBO/VTK is based on two broadly used technologies: CAVERNsoft, a library for supporting collaborative Virtual Reality; and the Visualization Toolkit, an extensive library for supporting 3D graphics and visualization.

Keywords: Virtual Reality, VTK, VR, Collaborative, Distributive, CAVE, CAVERN

1 Introduction

We define the term Tele-Immersion as the integration of audio and video conferencing with collaborative virtual reality (VR) in the context of data-mining and significant computation. When participants are tele-immersed, they are able to see and interact with each other in a shared environment. This environment persists even when all the participants have left. The environment may control supercomputing computations, query databases autonomously and gather the results for visualization when the participants return. Participants may even leave messages for their colleagues who can then replay them as a full audio, video and gestural stream.

As an example CAVE6D is a tele-immersive extension of CAVE5D[8], a tool based on Bill Hibbard's Vis5D[2]. Vis5D is a system for interactive visualization of large 5-D gridded data sets such as those produced by numerical weather models. One can make isosurfaces, contour line slices, colored slices, volume renderings, wind trajectory tracings etc, of data in a 3-D grid, then rotate and animate the images in real time. CAVE5D integrated Vis5D with the CAVE[1] allowing those visualizations to be created in VR. CAVE6D allowed multiple remotely located CAVEs, ImmersaDesks and desktop workstations to collaborate in this visualization. Participants are able to jointly turn volume renderings on or off, move wind trajectory tracings and turn animations on and off. This marked a significant improvement over CAVE5D because it allowed scientists from different research institutions to engage each other in a *virtual* research laboratory where they could share their insights on the data they were visualizing. This form of collaborative research is but only part of the goal of Tele-Immersion.

CAVERN (CAVE Research Network) is a collection of participating industrial (such as General Motors) and research institutions equipped with CAVEs, ImmersaDesks, and high-performance computing resources all interconnected by high-speed networks for the purpose of supporting Tele-Immersive- engineering and design; education and training; and scientific visualization and computational steering.

In 1992 there was a single CAVE in Chicago, in 1998 there are over 80 CAVE and ImmersaDesk installations around the world. One of the problems facing this growing community is how to best provide a mechanism to support long term collaborative work. This is a challenging problem because, unlike traditional multimedia applications, tele-immersion requires a tight, realtime coordination of a broad range of networking, database, user-interface, visualization, and virtual reality technologies. The goal is to create the illusion of co-presence amongst participants who may be physically located at geographically distant locations around the world. Developing tele-immersive applications can therefore be a daunting task as it requires expertise in all of these areas of computing.

A temptation and common mistake, made by application developers building tele-immersive applications for the first time, is to first build a non-collaborative application and then attempt to retrofit it for tele-immersive capabilities. It is important to provide tools that encourage application developers to envision Tele-Immersive scenarios at a high-level so that they can determine how such capabilities would be most useful in their own applications. However, a high-level set of tools does not help those trying to retrofit legacy applications. A high-level library of well-integrated tools often assumes a specific software design that may be incompatible with the software that is being retrofitted.

To address this issue, a software infrastructure called CAVERNsoft is under development. CAVERNsoft supports the rapid creation of new Tele-Immersive applications, and eases the retrofitting of previously non-collaborative VR applications with Tele-Immersive capabilities. A full discussion of CAVERNsoft is beyond the scope of this short-paper. This can be found by reading the following papers: [5, 4, 3]. This paper will describe the lessons learned in adapting the Visualization Toolkit to a multi-pipe VR environment and the work in CAVERNsoft to provide a re-usable software tool called LIMBO that offers application developers a template for building tele-immersive applications. In addition it describes the integration of LIMBO with the Visualization Toolkit

(VTK) to support the rapid development of collaborative scientific visualization applications.

Although this work was initially targeted at CAVE and ImmersaDesk users, a CAVE Simulator is available that will run on desktop Silicon Graphics workstations. This allows non-CAVE/ImmersaDesk owners to also benefit from the results of this work.

2 Adapting the Visualization Toolkit for Virtual Reality

The Visualization ToolKit (VTK) is a freely available C++ class library that supports 3D graphics and visualization [6, 7]. VTK was used because of its flexibility and wide array of supported visualization algorithms. The entire system consists of more than 400 classes, so the range of options for visualization are wide and varied. The core computational objects are compiled C++ code. These objects can be accessed using wrappers for scripting languages such as Tcl/Tk, Java, and Python.

The Visualization Toolkit is based on the data-flow paradigm and works by creating what is referred to as a VTK pipeline. A basic pipeline consists of three objects: a Source for data, a Filter to convert that data into geometry, and a Mapper to map the geometry to graphics. An object called a vtkActor contains a reference to the Mapper object and is the access point for any particular graphical object.

One particular aspect of VTK's design that contributes to its power is that it incorporates an implicit control of execution. This means that whenever something changes in one of the stages of the pipeline, the changes will automatically propagate down through the pipeline the next time output is requested from the Mapper. This methodology makes visualization applications easy to write and maintain.

In the process of integrating VTK with the CAVE environment, we found that VTK was not designed to be able to render the same geometry multiple times simultaneously to different graphics pipes. Specifically, geometry in VTK is stored in a data structure called vtkPolyData. This structure cannot be traversed by multiple processes simultaneously.

Our solution to this problem is to compute geometry using VTK and render using IRIS Performer. Doing so allows an application developer to harness all of the visualization algorithms of VTK and all of the rendering advantages of Performer, such as multi-process rendering to multiple views. Performer is a Silicon Graphics rendering library that also provides other features not found in VTK such as scenegraphs and intersection testing.

However, making VTK work together with Performer is not trivial. The library that was developed to allow these two toolkits to work together is called vtkActorToPF. Specifically, the code translates any vtkActor into a corresponding pfGeode in Performer. A vtkActor represents a graphical object in VTK. This includes geometry as well as surface properties and color information. A pfGeode is the analogous structure in Performer.

Using vtkActorToPF, the developer will build a VTK pipeline and a Performer scenegraph. Whenever the VTK pipeline re-executes and vtkActors change, the new geometry in these vtkActors will be translated and plugged into corresponding pfGeodes living in the Performer scenegraph. The Performer scenegraph is automatically rendered every frame (Figure 1.)

The vtkActorToPF code distribution includes the vtkActorToPF function and the vtkActorToPFTranslator. The function will translate any vtkActor into a corresponding pfGeode. However, if used alone, this function needs to be called explicitly whenever the vtkActor changes. When using the class, a translator object has a handle to a particular vtkActor as well a handle to a particular

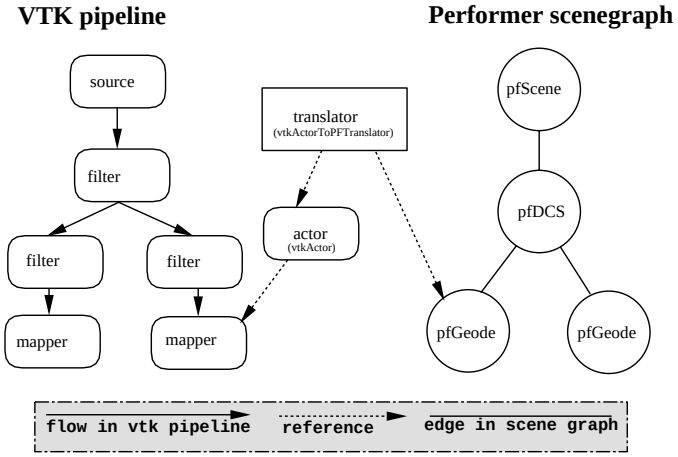

Figure 1: Relationship between a VTK pipeline and a Performer scenegraph.

pfGeode. Whenever the vtkActor changes, the pfGeode is automatically re-translated. This approach preserves the implicit control of execution model that VTK uses. By instantiating translator objects, any changes in the VTK pipeline will automatically propagate into the Performer scenegraph.

The vtkActorToPFTranslator object achieves this effect by registering a callback with Performer that is associated with the translator's pfGeode. Each time the scenegraph is traversed, the callback is made when the pfGeode is reached. The callback checks to see if the vtkActor has changed since the last translation and will re-translate if necessary.

Using the vtkActorToPFTranslator class, the developer instantiates a translator object for each vtkActor in the VTK pipeline. For each vtkActor a corresponding pfGeode is created and added to the Performer scenegraph. Once these objects are created, the two toolkits work together seamlessly and the translation is transparent. The translation itself is very quick compared to the execution time of the VTK pipeline. Since the translation is done only when portions of the VTK pipeline re-execute, there is little loss in interactivity as compared to a standard VTK application.

3 A Tele-Immersive Visualization Framework

LIMBO is an application framework or template that provides the basic capabilities of Tele-Immersion. LIMBO can be launched by connecting to a persistent LIMBO server or by connecting to another LIMBO client application. Other clients may join in at any time by either connecting to the server or latching onto another known client. Participants are depicted by virtual representations (or avatars) which have their head and hand tracked so that they are able to convey natural gestures such as nodding and pointing. Participants are able to fly through the space using a joystick and interact with the space using buttons on a spatially tracked pointing device called a wand. A separate tool may be used to deposit three-dimensional objects (possibly scientific data-sets, mechanical engineering parts, architectural structures, etc) in the shared space. Each participant can pick, move, and delete any of the objects. As the environment is persistent, participants may exit and re-enter the environment at any time. Finally participants are able to speak to each using a virtual intercom system.

LIMBO/VTK merges LIMBO with vtkActorToPF to allow application developers to use the rich set of visualization tools

built into VTK to generate sharable three-dimensional objects in LIMBO. The provided example in the LIMBO/VTK distribution consists of a tele-immersive environment to load and generate isosurface visualizations of volume data. VTK is used to load the volume data and generate the polygons of the isosurface. A user may instantiate a persistent copy of the isosurface at any time by pressing one of the wand's buttons. This persistent copy is distributed amongst all the remotely connected participants who are then able to pick and move the objects for inspection. Other participants may also join in changing the isosurface threshold to produce more persistent instantiations. Figures 2, 3, 4, 5 show participants actively engaged in LIMBO/VTK.

All data distribution between LIMBO clients is supported by CAVERNsoft. CAVERNsoft is a novel integration of networking and active database techniques to produce a distributed shared memory whose data objects possess three key capabilities necessary for supporting Tele-Immersion: Firstly the data objects may be shared via a variety of networking interfaces (unreliable UDP, reliable TCP, and multicast) with customizable networking Quality of Service (ability to specify a desired bandwidth, latency and jitter). Secondly the data objects can be made persistent by committing to a built-in database. Lastly the data objects can trigger user-defined actions whenever the data is updated.

LIMBO shares avatar data using an unreliable UDP channel. Since avatar data is transmitted frequently the loss of a packet is acceptable as it is soon followed by another. Each new packet of data will trigger an action to render the appropriate change in the corresponding avatar. VTK actors in LIMBO/VTK are translated into Performer pfGeodes that are then exported as 3D objects in Performer's binary model format. The 3D models are then distributed to all the participating clients via reliable TCP channels. On receipt LIMBO uses CAVERNsoft's database capabilities to render the 3D objects persistent. Absolute data (position and orientation data) about each of the instantiated objects are broadcasted via a reliable channel only when the objects are manipulated. This data is also stored in the database so that objects always remain in the correct location in the space even when participants exit and re-enter the environment.

In addition to sharing VTK visualizations, LIMBO can be used to share and visualize 3D models in any of the currently popular formats: Inventor, VRML1, DXF, OBJ, etc. This allows external applications to generate visualizations of their data and deposit it inside LIMBO for collaborative visualization.

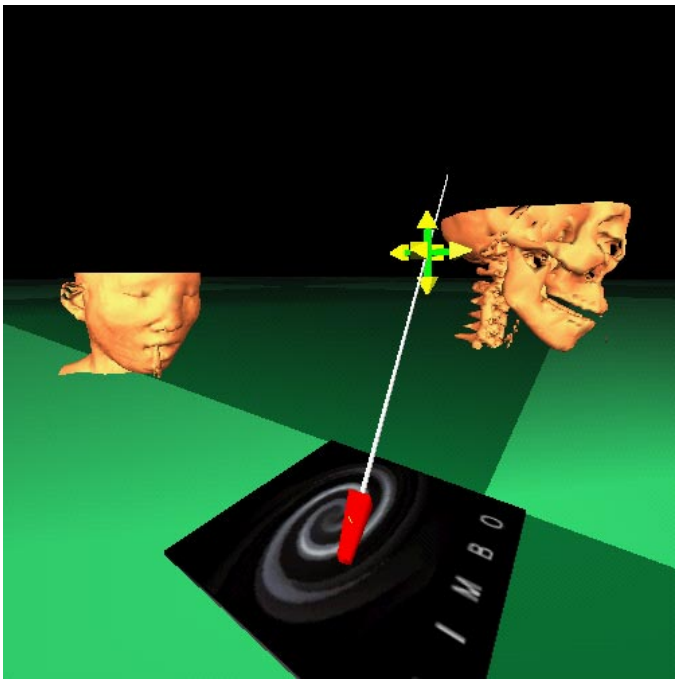

Figure 2: Inside the sample LIMBO/VTK world a user is able to extract isosurfaces from a volume data-set by setting different isosurface threshold values. Then a separate copy of the object can be created and moved.

4 Closing Remarks

VtkActorToPF allows the Visualization Toolkit and IRIS Performer to work together. This facilitates the quick development of interactive visualization applications for virtual environments such as the CAVE. The vtkActorToPF library alone can be downloaded from `http://hoback.ncsa.uiuc.edu/group/vtkActorToPF/`.

The LIMBO/VTK integration allows developers to build collaborative visualization applications without having to be experts in networking, databases and virtual reality. LIMBO/VTK is built on top of the CAVERNsoft Tele-Immersion architecture, IRIS Performer and the CAVE library. This allows remote participants to collaborate synchronously and asynchronously on CAVEs, ImmersaDesks and desktop workstations.

LIMBO will continually be improved as our CAVERNsoft effort develops new modules for Tele-Immersion. Future modules will include video conferencing tools. There is an effort currently underway to develop a virtual annotation module for LIMBO to allow participants to record synchronized audio and gestures of avatars while interacting in the environment.

CAVERNsoft, LIMBO and LIMBO/VTK can be downloaded from `http://www.evl.uic.edu/spiff/ti`. VTK can be

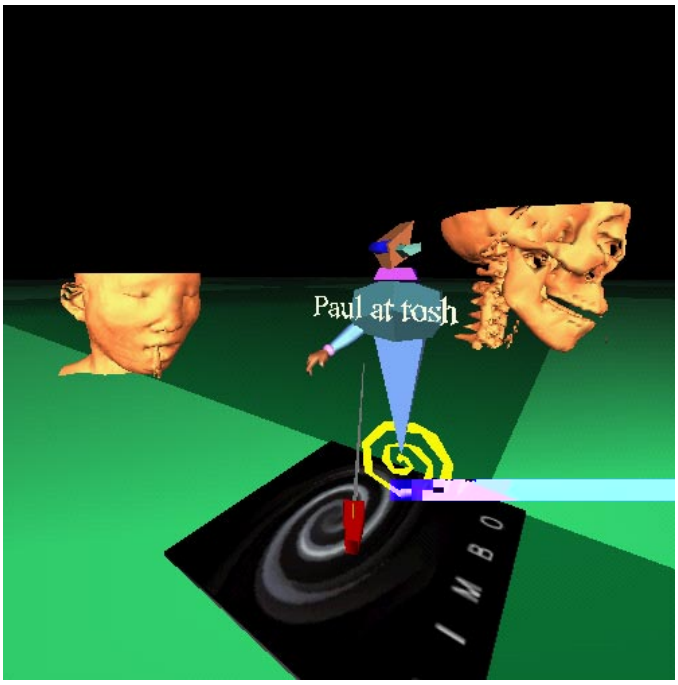

Figure 3: Another participant enters the shared space.

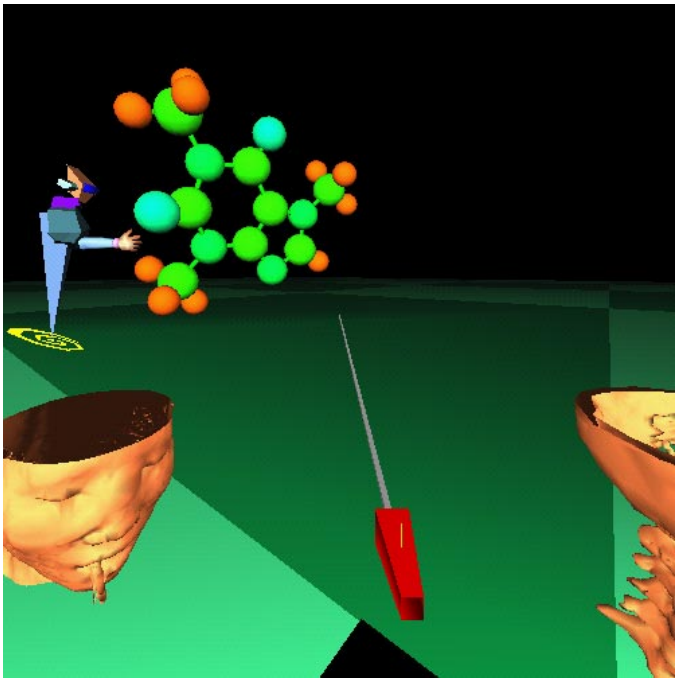

Figure 4: The participant will be able to generate his/her own set of isosurfaces or load in a completely separate 3D model. A Caffeine molecule is seen in the distance.

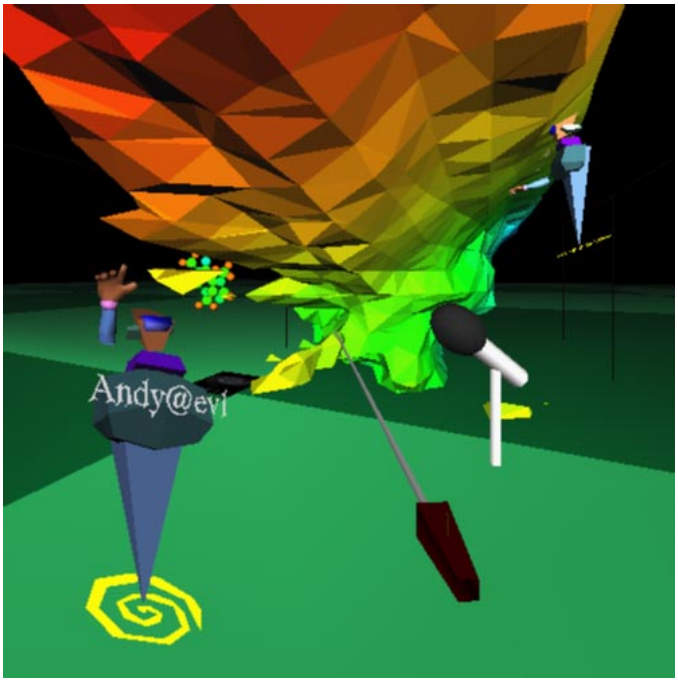

Figure 5: LIMBO and LIMBO/VTK can support several participants at the same time (network bandwidth permitting.) Each participant may bring in, move, and delete objects at will. Here one participant is speaking to another using the intercom.

obtained from `http://www.kitware.com/vtk.html`. The CAVE and CAVE Simulator library can be obtained by contacting `http://www.vrco.com`.

Acknowledgements

Major funding is provided by the National Science Foundation (CDA-9303433.) The virtual reality research, collaborations, and outreach programs at EVL are made possible through major funding from the National Science Foundation, the Defense Advanced Research Projects Agency, and the US Department of Energy; specifically NSF awards CDA-9303433, CDA-9512272, NCR-9712283, CDA-9720351, and the NSF ASC Partnerships for Advanced Computational Infrastructure program. The CAVE and ImmersaDesk are trademarks of the Board of Trustees of the University of Illinois.

We would also like to thank Randy Heiland for his early contributions to the development of the vtkActorToPF translator.

References

[1] Carolina Cruz-Neira, Daniel J. Sandin, and Thomas A. DeFanti. Surround-screen projection-based virtual reality: The design and implementation of the CAVE. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 135–142, August 1993.

[2] Bill Hibbard and Brian Paul. Vis5D http://www.ssec.wisc.edu/~billh/vis5d.html, 1998.

[3] Andrew E. Johnson, Jason Leigh, Thomas A. DeFanti, and Daniel J. Sandin. CAVERN: the cave research network. In *Proceedings of 1st International Symposium on Multimedia Virtual Laboratories*, pages 15–27, Tokyo, Japan, March 1998.

[4] Jason Leigh, Andrew E. Johnson, and Thomas A. DeFanti. CAVERN: a distributed architecture for supporting scalable persistence and interoperability in collaborative virtual environments. *Journal of Virtual Reality Research, Development and Applications*, 2(2):217–237, 1997.

[5] Jason Leigh, Andrew E. Johnson, and Thomas A. DeFanti. Issues in the design of a flexible distributed architecture for supporting persistence and interoperability in collaborative virtual environments. In *Proceedings of Supercomputing'97*, San Jose, California, Nov 1997. IEEE/ACM.

[6] William J. Schroeder, Kenneth M. Martin, and William E. Lorenson. The design and implementation of an object-oriented toolkit for 3D graphics and visualization. In *IEEE Visualization*, pages 93–100, 1996.

[7] William J. Schroeder, Kenneth M. Martin, and William E. Lorenson. *The Visualization Toolkit*. Prentice Hall PTR, 1996.

[8] Glen H. Wheless, Cathy M. Lascara, A. Valle-Levinson, D. P. Brutzman, W. Sherman, W. L. Hibbard, and B. Paul. Virtual chesapeake bay: Interacting with a coupled physical/biological model. *IEEE Computer Graphics and Applications*, 16(4):52–57, July 1996.