Immersive Visualization and Collaboration with LS-PrePost-VR and LS-PrePost-Remote

Todd J. Furlong
Principal Engineer - Graphics and Visualization
Inv3rsion, LLC – http://www.inv3rsion.com

Abstract

This paper describes two new branches of LS-PrePost that are designed to work together to extend LS-PrePost with immersive visualization and collaboration capabilities.

LS-PrePost-VR supports immersive visualization on a wide range of immersive displays, including CAVE-like devices, large-screen displays, and head-mounted displays. The software can run on either a single computer, a visualization cluster, or an SMP machine. By itself, LS-PrePost-VR supports command-line reading of supported file types as well as playback of command files generated by desktop versions of LS-PrePost. A VR input device provides an intuitive interface that includes animation control, an interactive clipping plane, and selection capability.

LS-PrePost-Remote is a client application that connects to the LS-PrePost-VR application and allows input to the application through the traditional LS-PrePost GUI. Multiple remote clients can connect to and synchronize with a VR session, allowing collaborative analysis on a corporate intranet.

The paper discusses software design and implementation, as well as possible future directions for this software. LS-PrePost-VR and LS-PrePost-Remote are developed and supported by Inv3rsion, a software engineering firm located in Goffstown, New Hampshire.

LS-PrePost VR

Software Overview

LS-PrePost-VR supports immersive visualization (also known as virtual reality) systems. It is based on the LS-PrePost source code, but the code has been modified to use VR Juggler to control the 3-D display. Additionally, some 2-D elements of the LS-PrePost display have been converted to 3-D objects. By itself, LS-PrePost-VR is limited to command-line arguments and the wand-based interactions described below. However, when used in conjunction with LS-PrePost-Remote, almost all features of LS-PrePost are accessible in VR.

VR Juggler is an open-source virtual reality application development framework developed by Dr. Carolina Cruz-Neira and a team of students at the Virtual Reality Applications Center (VRAC) at Iowa State University. VR Juggler supports multi-wall displays on either multi-pipe machines or visualization clusters. It also supports VR input devices such as motion trackers, gloves, and wands. LS-PrePost VR is written using version 2.0, specifically to take advantage of the clustered visualization capabilities offered in that version. Version 2.0 of VR Juggler is in an alpha testing phase as of the writing of this paper.

Through VR Juggler, LS-PrePost-VR can support traditional virtual reality devices, such as head-mounted displays (HMD's) or CAVE™ systems. Traditional systems are driven by multi-processor, multi-pipe machines such as an SGI Onyx. For each window drawn on a multi-pipe

system, VR Juggler creates a new application thread, thereby taking advantage of multiple processors if they are available on the system.

Bringing 2-D into 3-D

To bring the LS-PrePost display into VR, some display elements had to be converted from 2-D to 3-D. This primarily applies to text and to the fringe plot color key. All text in LS-PrePost-VR is drawn using "stroke" fonts from the GLUT library, which is freely available on most platforms. Stroke fonts are drawn using OpenGL lines, so the character sizes will recede with distance but the line widths will not. Text was then divided into two categories: 2-D and 3-D. In this context, 2-D text refers to the text that borders the traditional LS-PrePost window, and 3-D text refers to text that is attached to a part of a model. 2-D text in VR is drawn on a user-defined plane that can either be can be fixed in space, or it can be drawn in a heads-up display mode, where it appears fixed in the viewer's field of view. When the plane is fixed in space, it is useful to if it coincides with a wall of a CAVE™ or PowerWall™. The heads-up display mode may be useful when using a head-mounted display for the application. Additionally, a feature was added to print the last command processed in the lower right corner of the window, which mimics the behavior of the command display area in LS-PrePost.

Wand-Based Interaction

Basic interaction with a model in LS-PrePost-VR is supported through the use of a wand. Full interaction is supported though LS-PrePost-Remote, which is discussed later in this paper. The wand interactions described below apply to the InterSense IS-900 wireless wand, but they may be extended or adapted for other input devices. The IS-900 wand is tracked in 6 degrees-of-freedom in real-time, and it has 5 buttons and a hat joystick for input. Figure 1 shows an example of this type of wand and its button mappings for LS-PrePost-VR.

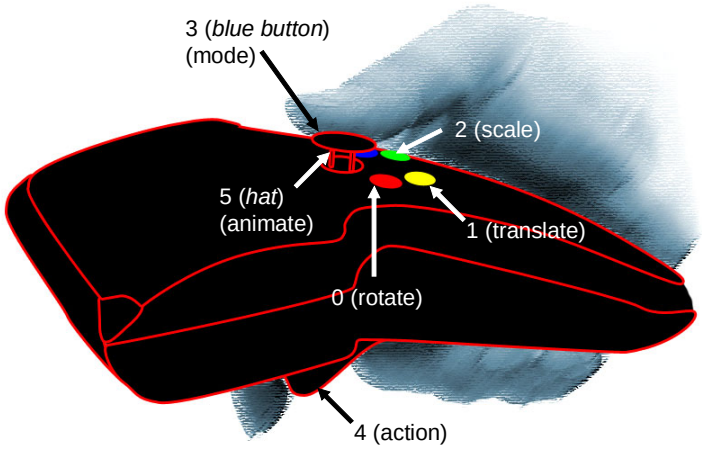

Figure 1 – IS-900 Wand Button Layout for LS-PrePost-VR

Model Manipulation

Manipulation of a model is designed to be intuitive for desktop LS-PrePost users. The red, yellow, and green buttons on the wand, which are organized from left to right, have similar functionality to the left, middle and right buttons of a mouse. Holding the red button and rotating the wand rotates the model the same amount. Holding the yellow button and translating the wand translates the model in the same direction with an acceleration factor applied. The acceleration factor makes it easy to move a model without having to overextend one's arms.

Holding the green button and moving the wand up or down scales the model uniformly up or down. Finally, pressing the red and yellow buttons together will reset the transformation.

Animation Control

Animation of a model is accomplished with the hat joystick. Rocking the hat right or left increments or decrements the animation by one frame. Holding the hat to the right or left causes the animation to play forwards or backwards respectively. Rocking the hat forward or backward increases or decreases the playback speed. Pressing the hat down stops the animation. If a command file is loaded rather than a plot file, the hat will control progression through the command file instead of the animation.

Selection & Clipping

The blue button on the wand, which is furthest to the right, is the "mode" button. Pressing this button repeatedly cycles through modes that are controlled by the trigger button. The modes are described below:

1. "Clip mode" is indicated graphically by a box drawn at the end of the virtual wand tip. In this mode, holding the trigger button creates a clipping plane at the position and orientation of that box. As long as the trigger is held, the clipping plane can be repositioned in real-time. Moving the clip plane through a model reveals the internals of that model interactively (see Figure 2). Releasing the trigger leaves the clip plane in place, and a quick pull on the trigger removes the clip plane. At this time, the interactive clipping plane only removes what is currently being drawn by OpenGL. Therefore, moving the plane through a solid will reveal it to be a shell. Future plans for the clip plane include combining the functionality with that of the typical section plane view of LS-PrePost so that the cross-sections appear as expected by the end-users.

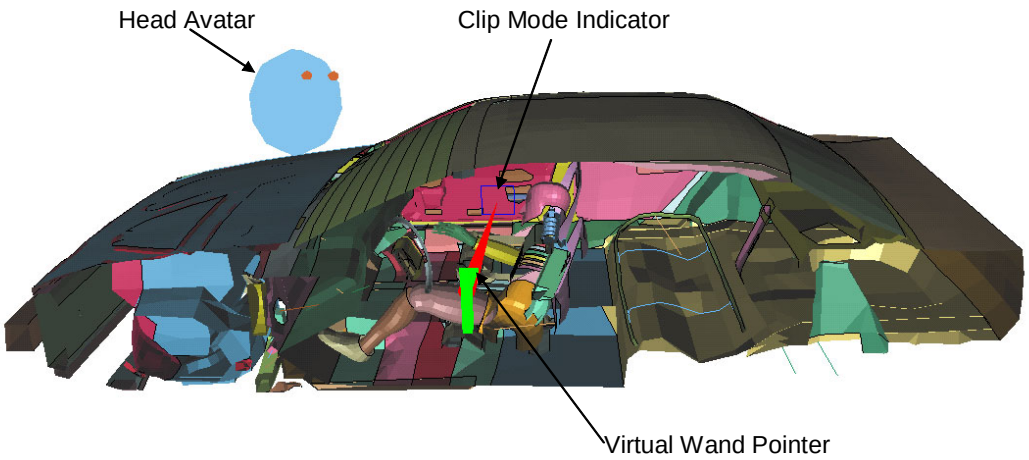

Figure 2 – Using the Interactive Clipping plane in LS-PrePost-VR

2. "Selection mode" is indicated by a virtual laser pointer projecting from the wand tip. By default, pressing the trigger in selection mode will perform an "ident node" command. Used in conjunction with LS-PrePost-Remote, the trigger will perform the selection action that matches the mode of the remote software. Currently, there is no equivalent of polygon or area selection in LS-PrePost-VR.

Configuring the VR Software

Customization of the VR software is done with a ".lsprepost_vrrc" hand-editable configuration file. In a cluster configuration, the rc file must be accessible to each node of the cluster to make sure that the nodes use the same configuration. Multiple rc files may be used, and the application will use the settings found in the most recently loaded rc file. Each time a model is loaded, the application looks for an rc file setting in that model's directory to allow per-model configurations. Some of the rc file settings are described below:

Scaling & Centering

Scale of a model can be either automatic or user-defined. User-defined scales can be useful for showing a model at a 1:1 scale. In this case, a scale factor is specified in the rc file that converts from model units to VR environment units. For example, a scale factor of 0.00328 is used to convert a model created in millimeters to display in a VR system that is configured to use feet. For auto-scaling, a rectangle is defined in space, and models are scaled such that their extents are confined to that rectangle. The model is also translated so that the model center coincides with the center of that rectangle. With auto-scaling turned off, auto-centering may still be used.

Text Drawing

A number of settings control how text is displayed in the VR environment. Such attributes as stroke width and text size may be customized, and the plane for "2-D" text is defined in the rc file as well. Depth cueing may also be enabled for text, which causes dimming of the text color as text recedes into the distance.

Wand Settings

Settings in the rc file control behavior of the wand buttons, such as how long a button must be held down to be considered held down by the application. Additionally, the user has control over the virtual pointer that is attached to the wand. It can be disabled, scaled, or its color may be changed. Similar settings apply to the pointer icons, which are the clip indicator box and the laser pointer in the current version of the software.

Application Behavior Settings

The VR application may be configured to ignore zoom, pan, and rotate commands, which gives the VR user full control over model manipulation. Some users may prefer full control from an LS-PrePost-Remote session, while others prefer to manipulate using the VR wand.

The VR application has the ability to transition smoothly from one transformation to the next rather than snapping into each new position as in command file playback in regular LS-PrePost. This was implemented to prevent viewers from feeling jarred or disoriented when viewing in an immersive environment. The number of frames of interpolation may be set in the rc file, and setting the number to zero defaults to regular LS-PrePost behavior.

LS-PrePost-Remote

LS-PrePost-Remote is a version of LS-PrePost that is designed to connect to a system running LS-PrePost-VR to provide input through the traditional LS-PrePost GUI. LS-PrePost-Remote was designed to run on a tablet PC that may be carried into a CAVE™-like environment. Using LS-PrePost-Remote on a tablet PC gives users an interface that they are familiar with while inside the virtual environment, and it provides a much wider range of input that the wand-based

VR controls. LS-PrePost-Remote can also be run on a standard workstation as long as there is a network connection between the remote computer and the server.

Networking Configuration

In the current version of the software, LS-PrePost-VR acts as a server, and LS-PrePost-Remote is a client application. Using TCP/IP sockets, the applications can transmit commands to one another. The main difference between LS-PrePost and LS-PrePost-Remote is in how commands are processed. Actions in standard LS-PrePost generate a command that is processed and written to an lspost.cfile for later playback. In LS-PrePost-Remote, commands are instead sent across a TCP/IP socket connection to the server. The server processes the commands and transmits them back to the client. The LS-PrePost-Remote client only executes commands that are received from the server.

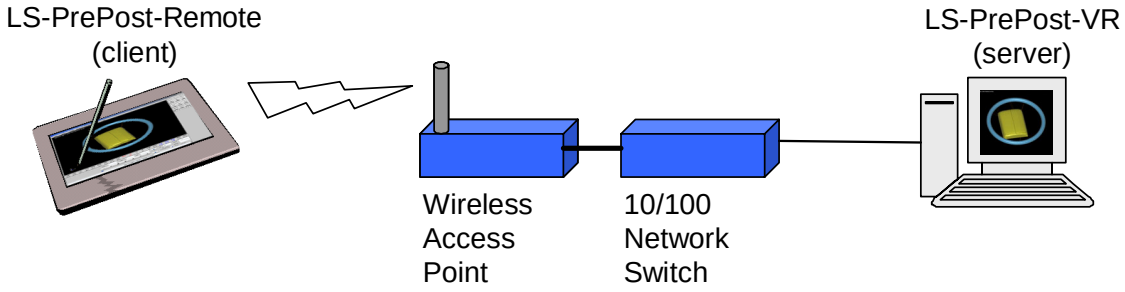

Figure 3 – Standard Network Configuration for LS-PrePost-Remote and LS-PrePost-VR

This type of client-server design has a number of advantages:

Asynchronous Execution

The VR and Remote applications are completely asynchronous. A tablet PC usually has fewer capabilities than the computers used in a VR system, so it is important that the VR system not be held back by the slowness of a tablet PC. Some VR applications that have an external GUI display process actions from the GUI within an execution frame of the VR application. In that type of setup, the GUI can completely stop execution of an application. Additionally, the contents of the GUI display in that type of setup must be simple enough that rendering them does not perceptibly slow the execution of the virtual environment. With the client/server approach, the tablet PC can show the whole LS-PrePost window, including the 3-D portion where much of the interaction takes place.

Command Sequencing

At first glance, it may seem odd for the remote computer to wait to receive a command from the server when that command is most likely the one that the remote computer just issued. However, the VR server application can issue commands (initiated by interactions with the wand) at the same time as the client. To deal with multiple computers issuing commands, the commands are put in order on the server and sent to the client(s) in that order. That way, each computer processes the same sequence of commands. Also, the commands are small, so that network bandwidth isn't taxed by this approach.

Multiple Clients

Finally, the client-server design can support multiple clients. This means that multiple tablet PC's can be used in a VR session, and users can connect from any number of locations across a

corporate campus as well. When the server receives a command from a client, it issues that command back to all clients. The command sequencing described above ensures that each client processes the same set of commands.

Late Connection Support

In a multiple client configuration, each client might not connect at the same time. People may arrive late to a design review, or they might accidentally close LS-PrePost-Remote and have to re-connect. To handle those eventualities, the server maintains a list of all commands executed during a session. When a new client connects, the commands from that list are issued to the client until is has caught up to the server. After that point, the new client can issue commands of its own, and all clients will be in sync with the server.

Remote-Related VR System Settings

All settings for LS-PrePost-VR and LS-PrePost-Remote are in the .lsprepost_vrrc file. The settings that apply to LS-PrePost-Remote include the hostname of the server computer, which tells LS-PrePost-VR which cluster node will be the command server.

Another important Remote-related rc file setting is a mapping of a remote mount point to the corresponding location on the VR system. This setting is required so that the application can find the files it needs to load. For example, a VR system might be a Linux system with files stored under /vr/models. That same directory may be mounted on a Windows machine on which LS-PrePost-Remote is run, and it might be mapped as Z:\vr\models or \\server\vr\models on that machine. The VR system uses the rc file to determine if it should look somewhere else when it is told to look at a Windows path. Note that the paths do not need to point to the same directory, so a local mirror of the "models" directory as described above may be preferable to a network-mounted drive, especially if the network connection to the server is slow.

Limitations

LS-PrePost-Remote contains almost all of the features of regular LS-PrePost, but a few features are disabled. One such feature is the multi-window display. When attempting to implement that feature, it was difficult to determine which commands should be transmitted to the VR system because some commands are only local to the active window and others affect all windows. Additionally, area and polygon selection are disabled in the current version, as is the "pcen" function. Those features use the LS-PrePost window dimensions to perform their functions, making it difficult to get consistent selection behavior across multiple walls of a VR system. To re-enable these features, a 3-D equivalent of area and polygon selection must be designed and implemented.

VR Cluster Implementation

Programming for clustered visualization can be more involved than developing for another type of system. In a visualization cluster, a separate instance of the same application runs on each node of the cluster. A network is often used to control swap-lock of the various nodes and to transmit data from input devices to all the nodes of the cluster. VR Juggler automatically handles distribution of motion tracking data and wand input, as well as frame lock and swap lock of the cluster displays. When LS-PrePost-VR runs in a cluster, the node that acts as the command server is responsible for distributing commands to the cluster nodes in addition to the remote clients. Commands are distributed to the nodes using VR Juggler's ApplicationData feature. See Figure 4 for a network diagram for a visualization cluster.

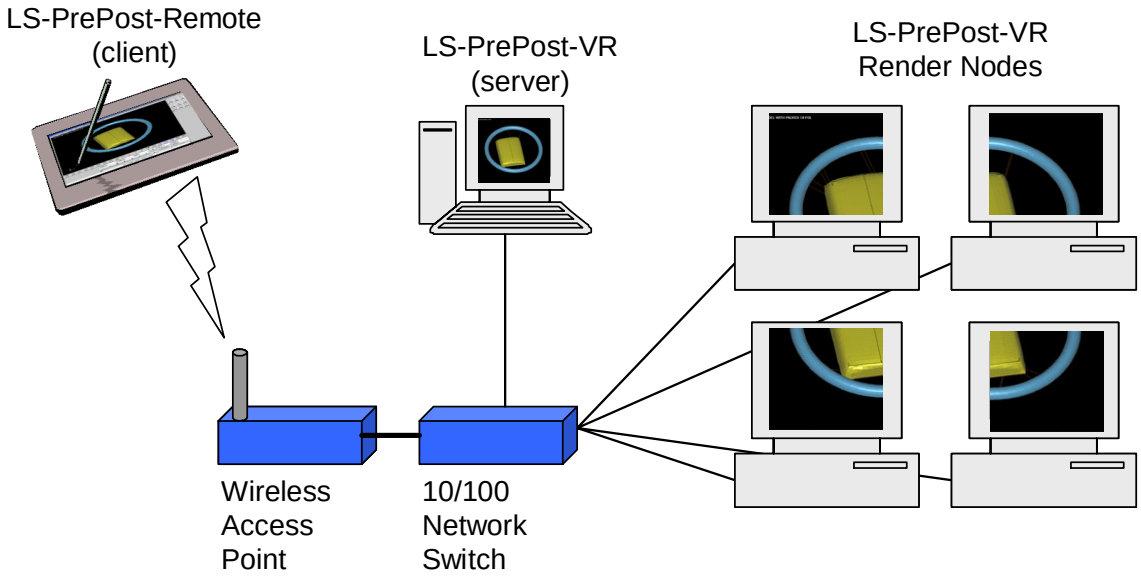

Figure 4 – Cluster Visualization with LS-PrePost-VR and LS-PrePost-Remote

In a cluster, users may experience a slowdown when loading files. This is because each node of the cluster loads a file at the same time from a network-mounted drive. The load times for a cluster have not been quantified yet, but experience tells us that loading data into a 9-node cluster does not take 9 times longer than a single-node implementation. Rather, it is closer to 3 or 4 times slower than loading from a single machine. This slowdown is also experienced the first time running through the frames of an animation in LS-PrePost-VR. However, once the data is local to each node, the system is very responsive.

Future Directions

Separate Server Application

In developing this software, it became evident that the server component would be better as a separate application rather than as a part of LS-PrePost-VR. With that type of configuration, the VR and Remote versions would each be clients that connect to a common server. That would enable multiple VR environments to share an LS-PrePost session. Alternatively, multiple remote versions could work together independently of a VR environment, creating a non-VR collaborative version of LS-PrePost.

Collaborative Tools

With the emergence of LS-PrePost as a collaborative tool, a desirable direction to take would be to add features that facilitate collaboration. One such feature would be to add awareness of other remote or VR users to the software. The mouse cursor location of a remote client could be propagated to the other clients and displayed as a ray projected into the scene using color and/or a name to identify individual users. Similarly, the wand position and orientation in a VR environment could be propagated to the other clients as a similar ray.

To further aid in using LS-PrePost as a collaborative tool, some markup features would be desirable. For example, the ability to highlight and annotate areas of a model in need of re-design for all participants to see and for later review would be a valuable tool.

Acknowledgements

Thanks to Joseph Kitching at Procter & Gamble for funding this work and for having the vision to push the development in the right direction for a successful VR application. Thanks also to Philip Ho and LSTC for making this project possible by allowing us to use the LS-PrePost source code. Special thanks to Scott Coppen, whose equal share in the development work contributed to many of the results presented in this paper. Thanks also to the VR Juggler team for providing a quality framework for application development.

References

Kilgard , Mark J., "The OpenGL Utility Toolkit (GLUT) Programming Interface, API Version 3", Silicon Graphics, Inc., 1996.

VR Juggler website, http://www.vrjuggler.org