Enhanced Collaborative Visualization with LS-PrePost-VR and LS-PrePost-Remote

Todd J. Furlong

Inv3rsion, LLC, Goffstown, New Hampshire, USA

Summary:

This paper describes the implementation of a new communications architecture that expands the capabilities of LS-PrePost-VR (virtual reality) and LS-PrePost-Remote. Using Inv3rsion's SinterPoint architecture, any combination of Remote and VR clients can share an LS-PrePost session. For Internet-based collaboration, we describe configuration of a VNC server to allow remote control of an LS-PrePost session.

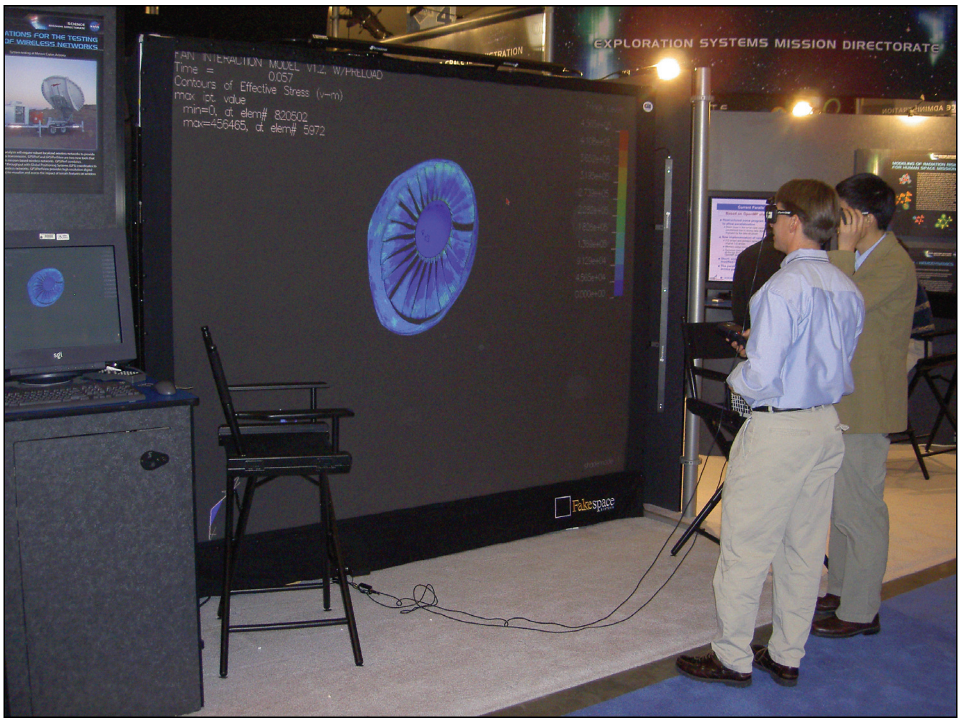
Keywords:

LS-PrePost, visualization, virtual reality, collaboration

1 Introduction to LS-PrePost-VR and LS-PrePost-Remote

1.1 LS-PrePost-VR

LS-PrePost-VR [1] is a branch of LSTC's LS-PrePost developed by Inv3rsion that supports immersive visualization systems such as CAVE^{TM} devices, large-screen displays, tiled displays, and head-worn displays. The software supports both SMP visualization systems and visualization PC clusters. A wand input device provides an intuitive interface that includes animation control, an interactive clipping plane, and selection capability. Fig. 1 shows users examining data in a single wall stereoscopic environment.



EXPLORATION SYSTEMS MISSION DIRECTORATE
FAN INTERACTION MODEL V1.2 W/PRELOAD
Time = 0.057
Contours of Effective Stress (v-m)
max ipt. value
min=0, at elem# 820502
max=456465, at elem# 5972
Fakespace

Fig. 1: LS-PrePost-VR Users Examine Data on a Fakespace ROVR System

1.2 LS-PrePost-Remote

LS-PrePost-Remote is a desktop version of LSTC's LS-PrePost with the capability to communicate with LS-PrePost-VR. Both LS-PrePost-Remote and LS-PrePost-VR can send and receive commands over a network to syncronize the state of any connected. LS-PrePost-Remote is designed to be used as input to an LS-PrePost-VR session.

2 SinterPoint Networking Architecture

The original versions of LS-PrePost-VR and LS-PrePost-Remote employed the VR software as a server and any number of Remote versions as clients. This had a few disadvantages:

1. VR-to-VR communication was not possible,
2. VR-independent Desktop-to-Desktop communication was not possible, and
3. VR perfomance could suffer for large numbers of Remote clients

To remedy those shortcomings, we developed the SinterPoint architecture, which consists of two components:

1. SinterPoint server, and
2. SinterPoint client API

Both the SinterPoint server and client API are cross-platform and are currently available on Windows, Linux and Irix.

2.1 SinterPoint Server

The SinterPoint server is an application that is designed to accept client connections and forward command strings to the appropriate connected clients. It is designed to facilitate networked collaboration between instances applications that are controlled by a command or scripting language. The server uses the concept of "sessions" and "connections" to manage clients. Fig. 2 illustrates the flexibility of server and client combinations.

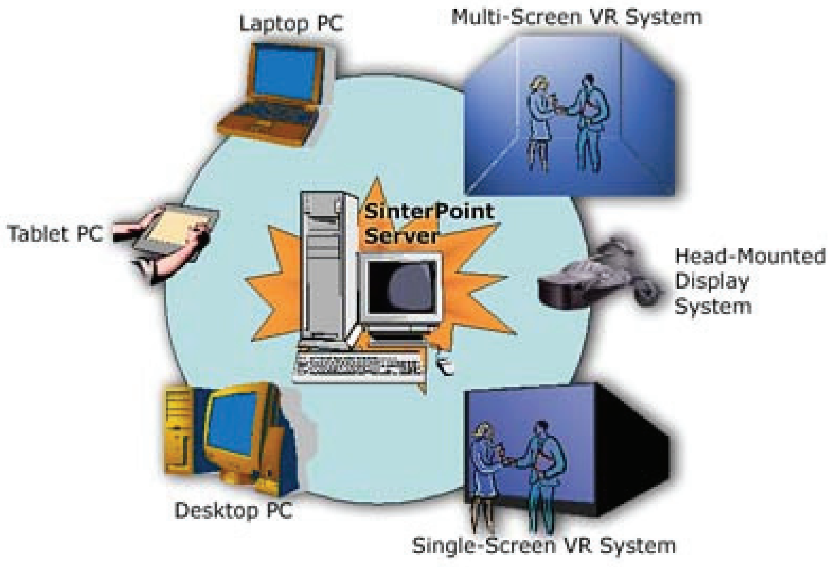


Fig. 2: Illustration of SinterPoint connectivity in a mixed VR and desktop environment

2.1.1 Session Management

When a client application of a certain type (e.g. LS-PrePost) attempts to connect to the server, a "session" is created to manage that connection. When other clients of the same type attempt to connect, those connections are assigned to the session created by the first connection. If a client of a different type attempts to connect, a different session is created to manage that type of client. If all clients of a type disconnect, then the associated session ends. A different session may represent a different stream of commands for a single application, or it can represent different applications that are independently using the server. This server design means that the server application is simply a "session manager" because the sessions manage the various connections, and connections manage the actual data.

2.1.2 Connection Management

A connection to a SinterPoint session can be either a "send" or a "receive" connection, meaning the connected client will either send or receive command strings. A single application can connect as both a sender and a receiver, allowing two-way communication through the server. The number of send or receive connections may be limited by the properties of that session type. The session connection limitations are established when the first client of any type connects. Limitations are enforced by the server to accomodate a range of applications, such as those that may only support a single send connection but multiple receive connections. LS-PrePost-VR and LS-PrePost-Remote do not limit the number of connections.

Command strings that are sent to the server over a send connection are recorded and transmitted to any receive connections. When a receive connection is established to an active session, all commands issued during that session (the "session history")are first sent over the connection. That is done to allow receive clients to connect at different times, disconnect, and reconnect with the ability to "catch up" to the state of the active session. For most applications, this is the least intrusive way to relay the current state of the application when a new client connects.

2.2 SinterPoint client API

The SinterPoint client API is a C++ API designed to be a simple interface for connecting to a server and sending and receiving data. The API consists of two primary objects: sinter::Sender and sinter::Receiver. The code example below demonstrates creation and usage of a sender class to send a "hello world" message to a SinterPoint server:

```
sinter::Sender sender;              // SinterPoint sender class
sender.SetType( "DEMO" );           // Declare Session Name
sender.SetVersion( "1.0" );         // Declare version of this application
if(!sender.Connect( “server“ ))     // Connect to server
   sender.Send(“Hello World“);      // send a command to server
```

The next piece of code demonstrates creation and usage of a receiver class to wait for commands and print them:

```
sinter::Receiver receiver;          // SinterPoint receiver class
receiver.SetType("DEMO");           // Declare Session Name
receiver.SetVersion("1.0");         // Declare version of this application
if(!receiver.Connect( “server“ ))   // Connect to server
  while( 1 )                        // receive & print until Ctrl-C
    if(receiver.Receive( -1 ))      // -1 means wait forever to receive
      fwrite( (void *) receiver.Data(), 1, rc, stdout );  // print received command to stdout
```

The simplicity of the client API allows simple networked applications to be created quickly. More advanced applications can be created by managing multiple send & receive connections.

3 LS-PrePost and SinterPoint

LS-PrePost is a command-driven application, which means that every event is translated into a string that is processed by a command interpreter. Those commands are saved to a command file so that a user can play back an LS-PrePost session. Using SinterPoint, LS-PrePost-VR and LS-PrePost-Remote transmit and receive those commands over a network connection to keep application states in sync.

In the current incarnations of LS-PrePost-VR and LS-PrePost-Remote, each application is both a SinterPoint send and receive client. That means that any connected client can "take control" of the visualization. The VR nodes only send a limited set of commands, so the visualization is primarily controlled by the Remote clients. Because any node can send commands to all other nodes, the question is often raised as to how the sequence of commands can be maintained. The SinterPoint server arranges commands in the order they are received and send the same command stream back to all the connected clients. The clients are designed to only process commands issued by the server instead of those issued locally, so they all process the same command stream.

4 Collaboration

The word collaboration has many connotations. In this text, it is taken to mean a shared visualization state. Multiple users at multiple locations can see and interact with the same data. In the case of LS-PrePost, the shared data is a 3-D representation of LS-DYNA output and the associated view mode, which includes such attributes as fringe data, view settings, and animation state.

4.1 Intranet Collaboration

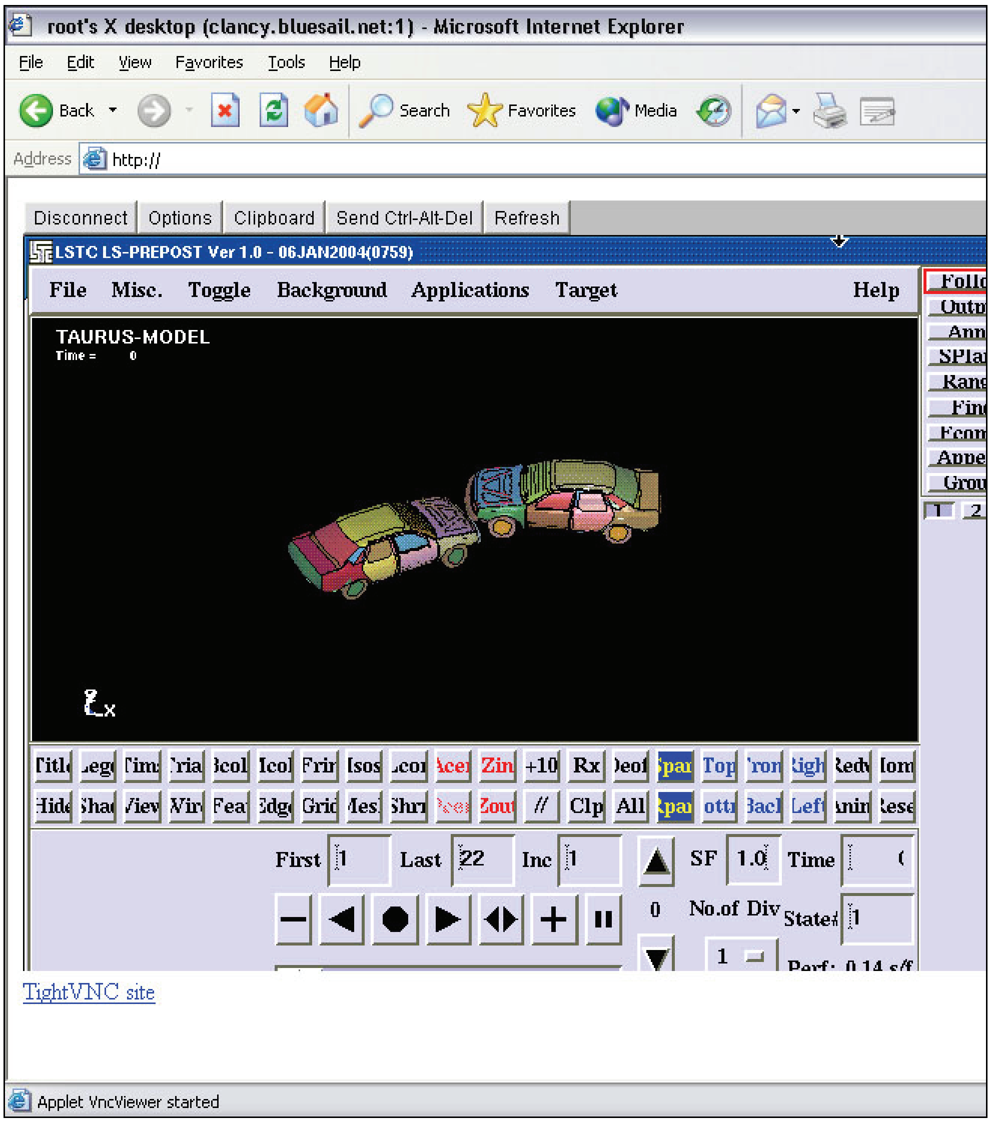
Using SinterPoint-enabled LS-PrePost over an intranet, each client is required to have the same files (e.g. d3plot files) available either locally or over a shared drive. Beyond that, there are no restrictions on which types of clients can connect and communicate. For VR users, this means that multiple VR systems can be set up to share an LS-PrePost environment. For desktop LS-PrePost users, this means that users at multiple desktop locations can participate in a shared session. Any user can start up the SinterPoint server and ask colleagues to connect to the session. There is no requirement for a homogeneous computing environment. Therefore, a visualization system may run Linux or Irix, and desktop users can still connect from their Windows machines.

4.2 Internet Collaboration

It is not always feasible to get data to all Internet-connected participants in a collaboration session. Therefore, we employ the following components to facilitate Internet collaboration:

1. VNC server
2. Mesa-compiled LS-PrePost-Remote

Readers may be familiar with VNC. It is free software that allows desktops to be shared across networks. On Linux and Unix, VNC creates a virtual desktop, which prevents the use of a hardware-accelerated version of LS-PrePost. Therefore, LS-PrePost must be compiled with Mesa [2], a software-rendering OpenGL work-alike, to render to a VNC desktop. To create an Internet-capable collaboration session, a user must start VNC server and open LS-PrePost-Remote on the VNC virtual desktop. An Internet-connected participant may then connect to the VNC server with either a Java-enabled web browser or a VNC client application. A Java-enabled browser can automatically download a VNC client applet to allow a user to connect to the session. See Fig. 3. A normal VNC client may be faster, but the end-user has to download and install it manually. Popular incarnations of VNC server and client include RealVNC [3] and TightVNC [4].


root's X desktop (clancy.bluesail.net:1) - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://
Disconnect Options Clipboard Send Ctrl-Alt-Del Refresh
LSTC LS-PREPOST Ver 1.0 - 06JAN2004(0759)
File Misc. Toggle Background Applications Target Help
TAURUS-MODEL
Time = 0
First 1 Last 22 Inc 1 SF 1.0 Time
No.of Div State# 1
TightVNC site
Applet VncViewer started
Fig. 3: Screen Shot of Java applet connected to a VNC session running LS-PrePost

VNC can also be run on an intranet to support desktop collaboration without SinterPoint. Using VNC, users do not have to share a drive in order to connect to a session. However, we have found that performance and visual quality are better if a user can run LS-PrePost-Remote locally rather than connecting to a VNC session. Performance is better because of hardware-accelerated OpenGL rendering, and responsiveness is better because mouse and keyboard events happen locally. Visual quality is also better because it is not subject to compression or color reduction that VNC users usually employ to increase performance.

5 Summary and Future Directions

With the transition to the SinterPoint architecture for LS-PrePost-VR and LS-PrePost-Remote, we have opened up more collaboration possibilities. The functionality now exists to create VR-to-VR or Desktop-to-Desktop collaboration sessions in addition to the Desktop-to-VR control that was originally possible with these applications. The new architecture, with support for multiple command streams, can be used to further enhance the collaborative experience in the future. Future goals with this software include the ability to transmit user information so that participants can visualize the user ID's and head, hand, and/or mouse positions of other connected clients. Another future goal is to extend SinterPoint to support binary file transfers to remove the requirement of a shared or mirrored filesystem. A shorter-term goal is to get users to start working with the current capabilities and to incorporate feedback form that eperience into future versions of the software.

6 Literature

[1] Furlong, T. J.: "Immersive Visualization and Collaboration with LS-PrePost-VR and LS-PrePost-Remote", 8th International LS-DYNA Users Conference, 2004, 15-1 - 15-8
[2] Mesa 3D Homepage, http://www.mesa3d.org
[3] RealVNC Homepage, http://www.realvnc.com
[4] TightVNC Homepage, http://www.tightvnc.com