Proceedings of the 2002 Winter Simulation Conference
E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, eds.

VRML CLIENTS LINKED THROUGH CONCURRENT CHAT

Lee A. Belfore II

Department of Electrical and Computer Engineering
Virginia Modeling Analysis and Simulation Center
Old Dominion University
Norfolk, VA 23529, U.S.A.

Sudheer Battula

YnotLearn, Inc.
Virginia Beach, VA 23452, U.S.A.

ABSTRACT

Internet based virtual reality offers the opportunity to render content in three dimensions. In addition, the Internet provides a medium to support collaborative activities. In this work, we describe how collaborative capabilities are integrated into the Interactive Land use VRML Application (ILUVA). ILUVA is a VRML based application that supports highly interactive functionality, live updates, and the dynamic generation of VRML content. The collaborative functions have been added in the context of an Internet chat session in which multiple users may participate from the Internet. In addition to the usual functions supported by chat applications, user information sharing is supported. The union or intersection of sessions from different users may be produced and reviewed in the world.

1 INTRODUCTION

Internet technology provides both the fabric and function for collaborative enterprise. Providing the ability to effectively communicate from any location on the Internet is an important consideration given the impact the Internet has on many people's personal and business lives. Several technologies provide useful bases for exploration. Our VRML applications have evolved, first through development of the client part (Belfore 2001) and then have been extended to include client server applications (Belfore and Chitithoti 2000, Belfore and Chitithoti 2001). The client applications are stand-alone applications that support live updates, or the ability to add content to the world without reload. A variety of models can be added to a client session including, for example, buildings, roadways, and areas of interest. Most models include the ability to be moved and edited to suit the particular circumstances. Collectively, the session represents an area of interest that includes improvements and modifications that are visualized. The application integrates GIS and planning information appropriate for the theme of the application. To expand the capabilities, server processing has been developed to increase the flexibility and capabilities (Belfore and Chitithoti 2000, Belfore and Chitithoti 2001, Belfore 2002). The server processing provides the opportunity to enforce access control, enable selection of the content to be viewed, and to generate content dynamically. Indeed, in Belfore (2002), GIS VRML content is generated by translating native GIS content, in the form of ESRI Shapefiles (ESRI 1998), on demand.

To support information sharing, a mechanism must be in place to allow users to determine what information can be shared and how that information is shared. As described in prior works (Belfore and Chitithoti 2000, Belfore and Chitithoti 2001, Belfore 2002), a user has several usage options including starting a new session, revising an old session, or including a session made available by another user. Sessions from other users are read only to protect the integrity of the original user's session. The application supports multiple simultaneous logins from a single user who is restricted to reviewing sessions not actively being reviewed. Interaction among users is limited since information can be shared only at login. The results described in this paper extend prior efforts by enabling the creation and management of virtual worlds that allow interactive collaboration through a chat session. In addition to the expected capabilities associated with a chat application, users may share information from sessions each has created. Furthermore, simple operations for combining information from different user sessions have been implemented through intersection and union operations. The operations give participants the ability to determine commonality between their respective efforts through the intersection operation and combine their efforts using union operations.

Several other researchers have made contributions in the development of VRML based collaborative virtual worlds (Testani et al. 1999, Brutzman et al. 1997, The Web3D Consortium, Incorporated 2000, Kirner et al. 2001, Damer et al. 1999). The work most similar to ours includes a concurrent chat that links clients in an educational application (Kirner et al. 2001). A chat that links clients and their

worlds to a chat server enables communication and modification of the worlds. One characteristic that distinguishes our application is the manner in which object properties are changed. In Kirner et al. (2001), a dialog appears for editing object location and appearance where as in our application, a user modifies objects through drag and drop actions on objects within the world. The Nerve Garden architecture (Damer et al. 1999) describes how algorithms mimicking life processes are used to create and plant artificial plants. Nerve Garden is a collaborative laboratory allowing users to create and germinate plants in an applet window with subsequent insertion into a scene graph common to all users. The vrtp (Brutzman et al. 1997) describes a general approach for linking VRML based worlds and several example deployments (The Web3D Consortium, Incorporated 2000). Testani et al. (1999) describes a virtual training environment allowing participants at distant locations to collaborate. Interaction among users is accomplished using a multi-user domain (MUD). A key contribution of this work is the description of a software architecture.

This paper is organized into five sections including an introduction, an overview of the ILUVA architecture, a presentation of the chat architecture, an example session, and a conclusion.

2 OVERVIEW OF THE ILUVA ARCHITECTURE

The application architecture is assembled to link four primary components (Belfore and Chitithoti 2000, Belfore and Chitithoti 2001, Belfore 2001, Belfore 2002). The first component is the domain data that is used to set the context for the world. The second component is the server that satisfies user requests, converts between data formats, and logs sessions. The third component is the client application that renders the world. The fourth component, the chat component, is described in the next section. The interrelationship among the first three components is illustrated in Figure 1. At startup, the user requests the URL for the world. In response, the server provides both static information and dynamic content that is assembled to create the requested world. During a session, the user has the ability to make live updates, i.e. modify the world, save application information on the server, and restore a prior session.

2.1 Server Architecture

A key challenge in creating visualizations is being able to provide timely, up to date information to clients automatically. In order to achieve this flexibility, the server needs to be able to determine what information is available and then present the appropriate choices to the user upon logging in. Furthermore, the maintenance procedures for updating and adding information should be simple and reliable. Table 1 shows a high level sequence of the activities that occur in a

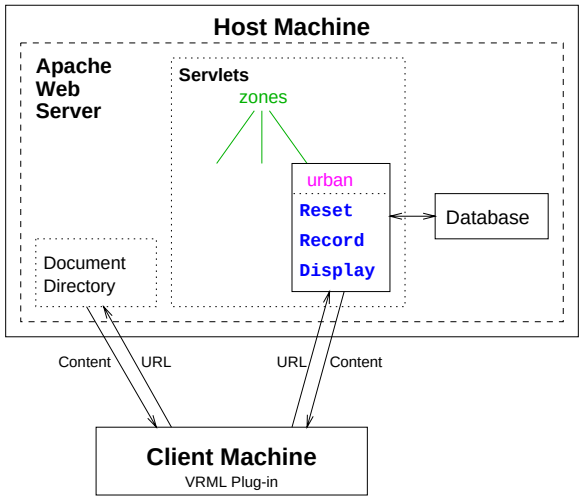


Figure 1: Top Level Architecture

session. The client and server have specific functions that occur in each step. In addition, the administrative aspects for each activity are described in the third column. The server architecture is built upon the Apache web server and ApacheJServ servlet engine. Apache provides all of the necessary capabilities to serve files on the Internet. ApacheJServ enables the coding of servlets, light weight Java methods, for access control, session control, generation of dynamic content, and logging user activity. The server architecture is designed to make update of current and development of new deployments as easy as possible. The current work builds on server functionality described in Belfore and Chitithoti (2001), Belfore (2002) and adds significant capabilities for integrating dynamic content. Users may log in and after

Table 1: User, Client, and Administrative Aspects

Client	**Server**	**Administrative**
Login	Login page, user authentication	User information
Configure	Provide choices, take responses	Data repositories
Generate	Content, assembly, generation, conversion	Methods to support generation of worlds
Explore	Monitor, log, content	Associate logging with user

logging in, the user has the opportunity to select information to view and other aspects related to the configuration of the world. From this configuration information, the requested information is formatted and presented to the user for verification. At this point, the server assembles the information and makes whatever conversions are necessary to include the information in the world. Once the client has rendered the world, the user explores the world. At various times,

information is passed back to the server to provide a record of certain events, such as the addition or modification of a feature.

2.2 Client Architecture

The details of the client architecture are described elsewhere (Belfore and Chitithoti 2000, Belfore and Chitithoti 2001, Belfore 2001). The client is written entirely in VRML and salient aspects of the architecture are described here. The purpose of the client is to provide a platform upon which to render the world and allow the user the ability to navigate through and interact with the world. In addition to managing the direct interactions with the user, the client architecture must also communicate with the server. The client architecture is shown in Figure 2. GIS information

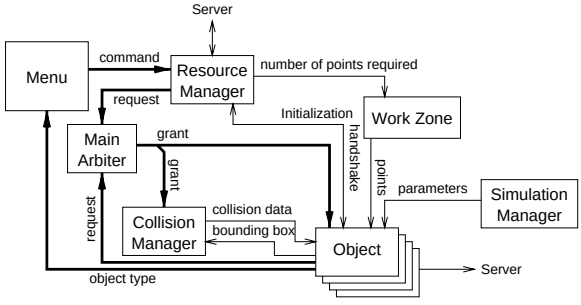


Figure 2: Client Architecture (Belfore and Chitithoti 2001)

can be integrated in several ways. In the base visualization, GIS provides a basic frame of reference for the application, describing terrain, imagery and other desired features. In addition, the GIS can be a component in the application that a user can change the appearance of or otherwise manipulate (Belfore 2002). To control the different capabilities and features, a user interface is included to help guide the user through the world. The user interface is implemented as a head's up display in the world that always follows the user. The user interface can include such features as navigation aides to assist the user in identifying important features or locations. A live update is a process whereby new content is integrated into an existing world. Depending on the complexity of the updated content and its linkage to the world, the process of performing a live update can be complicated. In ILUVA, added content can be moved and manipulated requiring a separate module, the resource manager, to coordinate. Furthermore, care must be taken in defining the architecture of the content to be added. Figure 3 gives an overview of the object architecture that is deployed in ILUVA (Belfore 2001). Browser applications typically have limited capabilities to protect the user from malicious or inadvertent modification of data on the client machine. Thus, in order to capture information from a user session, communication with the server is necessary.

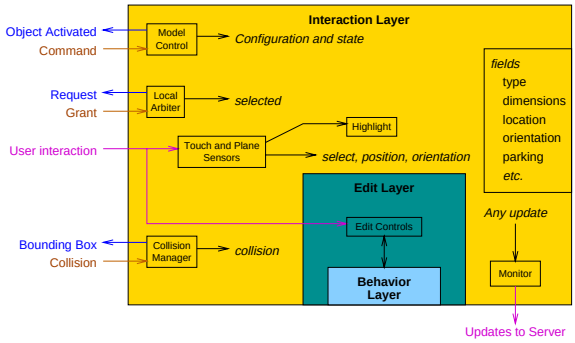


Figure 3: Object Architecture Supporting Live Updates (Belfore 2001)

The client application communicates with the server by requesting URLs for servlets. These servlets may direct the server to log the information or may also be a request for VRML content that can be subsequently be added to the world. These communication issues are described in detail in Belfore and Chitithoti (2001).

3 THE CHAT ARCHITECTURE

Building on prior work, ILUVA chat is a client server architecture that implements the chat that runs concurrent with the session. The technology for constructing chat applications is well defined (Harold 2000). In this work, users have two methods for communicating. Previously, ILUVA provided information sharing only through the sharing of sessions in a non-interactive fashion as described in Belfore and Chitithoti (2001). A chat application has been integrated into the ILUVA to extend its multiuser capabilities. The chat application, implemented in Java, provides a method allowing immediate communication among clients through a chat server application. The chat supports the exporting and importing of sessions between users. The latter part of this section explains the chat client, chat server, importing a session and exporting a session. The chat application architecture is illustrated in Figure 4. In order to assure session owners retain control over their session, sharing of sessions is accomplished only with their consent. The users negotiate sharing by exchanging chat messages. Export of a session occurs after the provider exports the requested session through the chat client. Upon notification of the export, the first user reloads the scene that includes the combination of the current user's session and the exported session.

3.1 Chat Protocol

The chat protocol operates in a client-server mode with one server and several clients. Sockets are opened between a single server and each client as each user logs in and users are

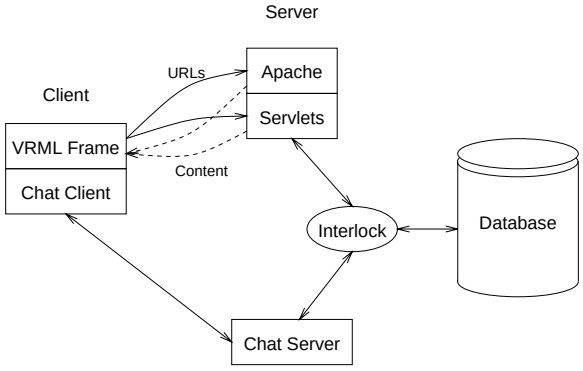


Figure 4: Chat Application Architecture

added. The chat server is capable of accepting connections from chat clients and relaying messages amongst the clients. The chat client takes input from the user and communicates this information to the server. In addition, the client displays any updates from the server. This forms the groundwork for the chat application. The chat protocol is simple and supports a handful of basic commands summarized in Table 2.

Table 2: Chat Command Summary

Command	Description
Start User List	informs the chat client that it is about to receive is the list of users
Add	adds a user to the user list box in all the chat
client Remove	removes a user from the user list box of all the chat clients
End of User List	no more user to be updated in the users list box of the chat clients of all users
export union	export operation with union the of two sessions has to be performed
export intersection	export operation with intersection the of two sessions has to be performed

3.2 Chat Client

The chat client that is embedded into ILUVA is a Java applet (Harold 2000). The chat client appears in a browser frame and has the appearance shown in Figure 5. Information relating to individual users is stored as browser parameters, previously set during the login process by servlets, that the chat client uses to negotiate a socket connection with the chat server. The chat client interface includes three buttons, two radio buttons, a list, and two text boxes. In one text box, messages from other users are displayed, serving as the bulletin board for the chat server. In typical "chat" fashion,

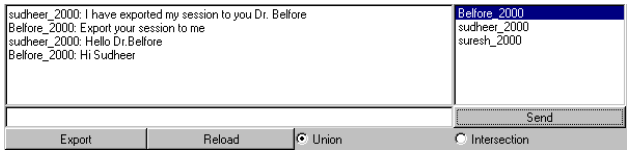
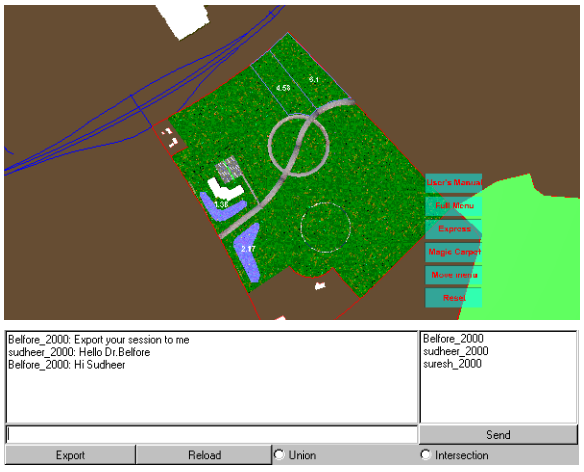


Figure 5: Chat Client Appearance

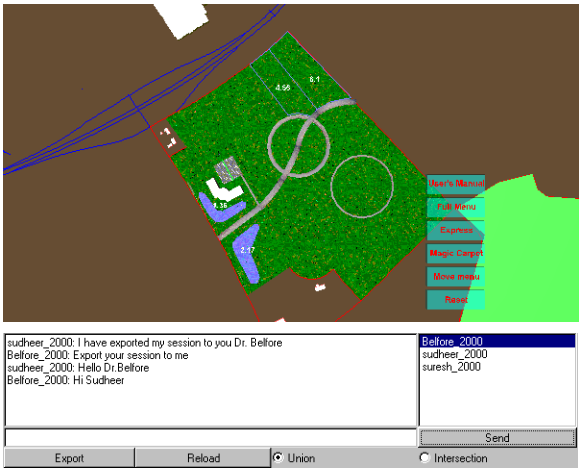
the user responds to the messages in the bulletin board by entering a message to send. The two remaining buttons are used for exporting a session to a user and importing a session from a user. The list box displays all the users who have been logged on this virtual environment. If a session is to be exported, the user has to select a user (to whom the user likes to export a session) from the list. A mouse click on the user in the list sets the environment for exporting a session to him/her. The export button has to be pressed in order to transfer a session to the selected user. Through the chat bulletin board the users can negotiate the exchange of information. Once the export is confirmed, the recipient selects the reload button to view all the objects that have been exported for the selected operation. The radio buttons for union and intersection control the export function. Every applet has a thread (Harold 2000) that reads to and writes from the chat server. The commands that are exchanged are listed in Table 2. In addition, the chat client handles functions such as adding a user to the user list upon logging in, deleting the user when logged out, and managing the client chat display. User information for the chat server is provided through servlet calls on the server that maintain and update list of concurrent chat clients.

3.3 The Chat Server

The chat server operates on the same machine that serves Internet content and servlet methods. This is necessary in our application because both access a common data base on this machine. The chat server will respond upon receiving any of the client commands summarized in Table 2 and perform the required processing. When a new user logs in, the user is connected to the chat server by issuing an add command. Since several users may be logged in simultaneously, multi-threading (Harold 2000) is necessary. Requests from the client are processed and the chat server provides several results including a list of chat participants, available exported sessions, and session operations. The chat server must interact with the same database as the servlets described above, so an interlock mechanism is implemented to ensure that the database is not corrupted. When either the export-union or export-intersection commands are issued, the server will combine the sessions in the requested fashion. Currently the union and intersection operations are determined by identifying coincident objects in the scene



(a) User 1 Request



(b) User 2 Export Operation

(c) Resulting Union

Figure 6: Example Chat: Union Operation

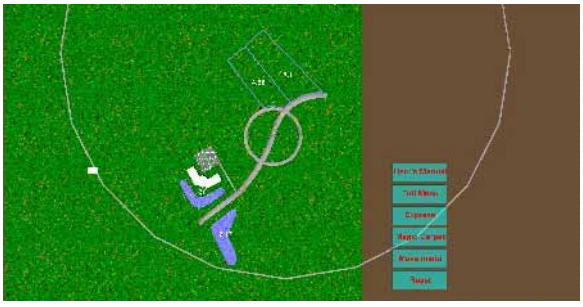
as listed in the respective session files for the users. The union operation enables collaborators to combine their work into a single session. The intersection enables users to examine differences between the sessions, useful if both are comparing modifications to the same area.

4 SESSION

In this section, an example of an interchange between users is presented. The ILUVA application supports sharing of information between users in two methods. The information sharing mechanism differs in whether or not the direct and immediate interaction with another user is necessary. In the first case, when users log in, sessions exported by other users are available for review (Belfore and Chitithoti 2001). By this method, sessions are exported in a read-only fashion to protect the integrity original user's session. In this work, a second, more interactive sharing method is presented that relies supported by a chat application. Using the chat, the users can merge their sessions using a union operation or identify the common components through the intersection operation. Figure 6 shows the process and result of a union operation. As noted earlier, the significance of the union operation is in the ability to combine work between users. In Figure 6.a, a user requests a session from another user. Figure 6.b shows the state and chat client of the second user. This user selects the operation, either union or intersection, and exports it to the first user. After the export, the second user informs the first that the export has occurred. With the confirmation, the first user reloads the scene as shown in Figure 6.c. In addition, Figure 7 shows the intersections between the initial session and the modified session.

5 SUMMARY

In this paper, we have presented extensions to the ILUVA architecture that support a concurrent collaborative chat session. The chat session supports immediate collaboration



(a) User 1

(b) User 2

(c) Intersection

Figure 7: Intersection Operation

and sharing of information among users that are currently logged in. Union and intersection operations are supported to show modifications and cumulative efforts respectively. Future work will focus on developing a wider range of chat operations, closer integration of the chat server with the web server, and also to support concurrent VRML worlds where modifications by one client are immediately reflected in other clients' worlds.

REFERENCES

Belfore, L. A. 2001, September. An architecture for constructing large VRML worlds. In *Transactions of the Society for Computer Simulation*, 24–40.

Belfore, L. A. 2002, April 14-18. An architecture support live updates and dynamic content in VRML based virtual worlds. In *Military, Government, and Aerospace Symposium*, 138–143. San Diego, California.

Belfore, L. A., and S. Chitithoti. 2000, December. An interactive land use VRML application ILUVA with servlet assist. In *2000 Winter Simulation Conference Proceedings*, 1823–1830. Orlando, Florida.

Belfore, L. A., and S. Chitithoti. 2001, April 22-26. Multiuser extensions to the interactive land use VRML application ILUVA. In *Thirty-Fourth Annual Simulation Symposium*, 159–166. Seattle, Washington.

Brutzman, D., M. Zyda, K. Watson, and M. Macedonia. 1997, June 18-20. Virtual reality transport protocol (vrtp) design rationale. In *Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE): Sharing a Distributed Virtual Reality*. Cambridge Massachusetts: Massachusetts Institute of Technology.

Damer, B., K. Marcelo, and F. Revi. 1999, January. Nerve garden: A virtual terrarium in cyberspace. In *Proceedings of the 1999 Virtual Worlds and Simulation Conference (VWSIM'99)*, 131–135. San Francisco, CA.

ESRI 1998, July. *ESRI Shapefile technical description*. ESRI.

Harold, E. R. 2000. *Java network programming*. O'Reilly & Associates, Inc.

Kirner, T. G., C. Kirner, A. L. S. Kawamoto, J. Cantao, A. Pinto, and R. S. Wazlawick. 2001, February. Development of a collaborative virtual environment for educational applications. In *WEB3D 2001*, 61–68. Paderbon, Germany.

Testani, S., E. Wagner, and K. Wehden. 1999, January. CIMBLE: The CADETT interactive multi-user business learning environment. In *Proceedings of the 1999 Virtual Worlds and Simulation Conference (VWSIM'99)*, 105–109. San Francisco, CA.

The Web3D Consortium, Incorporated 2000. The distributed interactive simulation DIS-Java-VRML working group.

<http://www.web3d.org/WorkingGroups/vrtp/dis-java_vrml>.

AUTHOR BIOGRAPHIES

LEE A. BELFORE II is an Assistant Professor of Electrical and Computer Engineering at Old Dominion University. He received the B.S. degree in electrical engineering from Virginia Tech in 1982, the M.S.E. degree in electrical engineering and computer science from Princeton University in 1983, and the Ph.D. degree in electrical engineering from the University of Virginia in 1990. From 1982 to 1985, he was a Member of Technical Staff at AT&T Information Systems. From 1987 to 1988, he was a research scientist in the Department of Electrical Engineering, Center for Semicustom Integrated Systems at the University of Virginia. From 1990 to 1997, he was with the Department of Electrical and Computer Engineering at Marquette University, Milwaukee, Wisconsin. Since 1997, he has been with the Department of Electrical and Computer Engineering at Old Dominion University, Norfolk, Virginia. His research interests include neural networks, data compression, and Internet based virtual reality applications. He is a Senior member of the IEEE and a member of the Computer Society and the Systems, Man and Cybernetics Societies. His e-mail address is <lbelfore@odu.edu>.

SUDHEER BATTULA is currently employed with YnotLearn, Inc., Virginia Beach, Virginia. In addition, Mr. Battula is currently working towards an M.S. Degree in electrical engineering from Old Dominion University. Mr. Battula received his B.E. in electronics and communications from Osmania University, Hyderabad, India.