Distributed Collaborative Virtual Environment: PaulingWorld

Simon Su[1], R. Bowen Loftin[2], David T. Chen[1], Yung-Chin Fang[1], Ching-Yao Lin[1]

[1]Department of Computer Science
University of Houston
Houston, Texas, USA

[2]Virginia Modeling Analysis & Simulation Center
Old Dominion University
Suffolk, VA 23435

{ssu, chingyao}@cs.uh.edu, bloftin@odu.edu, dave@chen.net, yfang2@bayou.uh.edu

Abstract

This paper introduces Distributed PaulingWorld, a Distributed Virtual Environment application that supports collaborative visualization of molecular structures among multiple users within the same virtual environment. All the participants in the virtual environment have the same level of interaction in the application. In the application, a virtual menu that is attached to the left hand of the user is used to manipulate the molecule and the environment. The user that has the virtual menu has total control of the environment and the viewpoint of the users in the virtual environment. However, the virtual menu can also be transfer to another user in the virtual environment. Only the user that has the menu can chose to transfer the menu. At that point, the other user, upon receiving the virtual menu, will have the capability to manipulate the molecule and the virtual environment. Users are represented by avatars to indicate their location within the virtual environment.

Key words: Distributed Virtual Environment, Virtual Reality, Responsive Workbench, Collaboration.

1. Introduction

An individual computer system can no longer provide sufficient computing power to support the increasing requirements and complexity required in creating a realistic Virtual Reality application. Even on some single user Virtual Reality applications, multiple computer systems are required to create a Virtual Reality application that looks accurate and behaves realistically. Single user Virtual Reality applications have benefited from distributing their sub-processes on different processors to increase their performance. In a Distributed Virtual Reality application, multiple computer systems are used to accommodate multiple users regardless of their locations as long as those computer systems are networked. This communication will provide collaborators with a tool to work together without having to be physically present in the same location.

PaulingWorld (PW) is a Virtual Reality (VR) application that simulates and visualizes molecular structures [4]. It also supports acceptable soft real-time interaction and manipulation performance. PW uses static local two-dimensional control widgets to interact with molecular data. PW allows one user to examine the structure of a molecule via five different representations: ball-and-stick, vanderWaals' spheres, coded sticks, backbone, and icons that replace repetitive structures. Figures 1 and 2 show snapshots of the application using the vanderWaals' representation and partially expanded icons that replace the repetitive structures. The user is free to fly through the virtual environment while examining the molecule representation from different viewpoints. The application also allows the user to scale, translate, rotate, or attach the molecule to his hand to inspect the molecule at different levels of details.

Distributed PaulingWorld (DPW) is a Distributed Virtual Environment (DVE) application that allows more than one distributed operator to interact with the same molecular structure by sharing the same virtual world. DPW allows collaborative visualization of a molecular structure among distributed users. DPW introduces a multi-user mode into PW described in the previous paragraph.

PaulingWorld allows a single user to visualize and investigate in detail the structure of the molecule. However, if the user wants to conduct a collaborative study with another person, the user will be limited by the functionality provided by PaulingWorld. It will also be impossible to share a finding with another person since the user cannot take a snapshot of the view of interest at that moment and share the findings with a second person. DPW provide a perfect solution to the collaboration problem in PaulingWorld by supporting multiple users in the virtual environment. In addition to provide support for multiple users, DPW can also bring together users that are physically dispersed into the same virtual environment without having the users to travel to the same physical location.

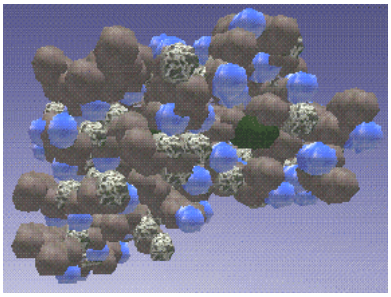

Figure 1 vanderWaals' Representation

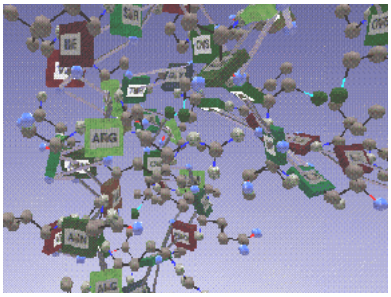

Figure 2 Iconic Representation

2. Related work

Research laboratories funded by both the government and private institutions have developed several practical and promising Distributed Virtual Reality applications [1][5][7][9][11][12][13][14][16][17]. Most Distributed Virtual Reality applications have some common properties. They are comprised of computer systems located at the same site or at geographically distant sites that are networked together, they use multiple processes, and they are used simultaneously by multiple people. The users interact with one another, and they are represented by an abstract representation to notify each other of their positions in the virtual environment. Un-Jae Sung et al [15] outlined some of the general characteristics of a DVE application. In general, DVE applications can be classified into large-scaled and small-scaled applications. A large-scaled DVE application may consists of several hundred nodes or participants, whereas a small-scaled DVE may consists as few as two participants within the same virtual environment.

Stytz at the Air Force Institute of Technology has done much work involving large-scaled DVE training applications that can support hundreds of participants in shared virtual environments [10]. The Synthetic Battlebridge gathered information from both the computer generated actors and human participants in real time and rendered a 3D image of the battlespace and its contents. This DVE application uses the Distributed Interactive Simulation (DIS) protocol [2] to manage its complex and active virtual environments. The DIS protocol governs the communication between hosts participating in the virtual environment.

Close Combat Tactical Trainer (CCTT) is another large-scaled DVE joint US Army-Loral project [14]. CCTT is a US Army training program that will help train ground combat tank and mechanized infantry forces within a realistic virtual environment. This DVE application also utilizes the DIS protocol to manage its complex and real time virtual environment. The simulator and individual workstations exchange data about their state information with respect to the virtual environment over the Fiber Distributed Data Interface (FDDI) using the DIS protocol. This application can support up to several hundreds of manned participants, computer-generated forces, and simulated vehicles.

In a more recent work by C. R. Karr et al [5], Synthetic Soldiers is a US military Joint Simulation System that was intended to create a single distributed virtual environment. The system is intended to provide joint training for all four branches of the armed services. As with most large-scaled DVE, Synthetic Soldiers also employs DIS to manage the communication of the hundreds of entities within the virtual environment.

Other than military research projects, most of the research done by academic institutions can be classified as small-scaled DVE. R. Bowen Loftin's work on the Hubble Space Telescope (HST) training project demonstrated a cross continental collaborative training in a shared virtual environment by astronauts in Houston and Darmstadt, Germany [3]. The virtual environment consists of a model of Space Shuttle payload bay and the HST. In the application, the training took the form of a simple extra vehicular activity (EVA) simulation that enable two astronauts on opposite sides of the Atlantic ocean to train within the same virtual environment. During the training, the two astronauts practiced the changeout of the HST's Solar Array Drive Electronics (SADE) and the real time hand off of the SADE within the virtual environment. The exchanging of state data was managed by IGD-developed communication software, and the virtual environment was rendered by NASA-developed graphics software. An Integrated Services Digital Network (ISDN) line was used to connect the sites together. Since absolute synchronization of the participants was required, no dead-reckoning algorithms were used in the application. A duplicate copy of the 3D environment database was also kept at all participants site to minimize the network traffic to the state change among the participants of the virtual environment.

Leigh's work in Collaborative Architectural Layout Via Immersive Navigation (CALVIN) shows the use of a DVE application to perform an architectural design and

collaborative visualization [6][7]. In this DVE application, Leigh emphasized the use of heterogeneous perspectives in viewing an architectural design to aid in the design and the collaborative visualization processes. With heterogeneous perspectives, CALVIN also demonstrated the use of virtual reality technology in the active design phase rather than the just as a walkthrough of the finished design.

Mourant's work in the Distributed Driving Simulator provided another example on a small-scaled DVE application [11]. Distributed Driving Simulator simulated the driving of a multiple driver within the same virtual environment. As in the case of HST training program, no dead-reckoning algorithms were used since the state change of one driver must be propagated immediately to the other driver to simulate a real time driving simulation. To minimize the network traffic, duplicate databases for the 3D environment and vehicles were also stored at the participants' local site.

Concurrency control within the shared virtual environment is also an important issue that needed to be addressed in a DVE application. The Collaborative Immersive Architecture layout (CIAO) paper described how concurrent actions are coordinated in a multi-user DVE application [15]. It achieved optimal response and notification time without compromising consistency through a new multicast-based, optimistic concurrency control mechanism.

3. Hardware and Software Environments

DPW is currently implemented between sites that have interactive workbenches. At each site, an SGI Onyx2 with multiple graphics channels drives a projector that produces display on the workbench. Tracking of the participants are accomplish by using Polhemus Fastrack™ each with a stylus and two other sources. Both the user's hands and the viewpoint are tracked at interactive rates.

Although we chose workbench as the display device, the application can easily be modified to use homogeneous socket communication protocol to display on a multiple-wall CAVE™ display device or a head mounted display device. The use of Polhemus Fastracks™as the tracking device can also be replace with Ascension's Flock of Birds tracking device. VrTool was the software toolkit used to develop this application [8] (not to be confused with Vr-tools developed by Christian Michelsen for NorskHydro). Figure 3 shows the workbench setup that was used in the application.

4. Application Design

4.1 Application Architecture

The DPW application is controlled by a main VrController process that manages and synchronizes all the states among the participants of the virtual environment. In addition, all the processes on a participant site are managed by their own local VrController. The main VrController is responsible to process and communicate with all the local VrControllers running at the participant site. Figure 4 shows the connection between the local VrController and all the processes at a participant site.

Figure 3 Workhench and Polhemus Fastrack

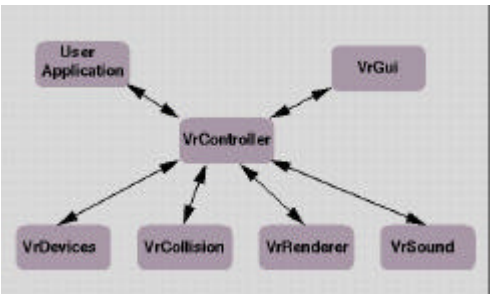

Figure 4 Application Architecture

Each participant process in the virtual environment can be divided into VrDevice, VrCollision, VrRenderer, VrSound, and VrController. The local VrController manages and synchronizes all other 5 processes of the local participant in the virtual environment. VrDevice is a process that is responsible for reading the raw data from the hardware devices and pre-process the data into the format that the application can use. VrCollision is responsible for detecting any collision among the objects that have been registered for collision within the virtual environment. VrRenderer is responsible for traversing through the scene graph that has been continuously updated by the VrController and render the object on the scene graph onto the display device. VrSound is responsible for playing any sound even in the virtual environment. The user application is the process that actually implements all the features of DPW. Commands that the user executes will be process

in this process and the state change is sent to VrController to be updated accordingly.

DPW employs a distributed database model to sustain the DVE. Every site retains a copy of all the models used in the DVE. This replication allows the DPW to be implemented over a regular Internet connection with no dedicated network connectivity with acceptable lag. Since all sites have copies of all the models, only the necessary state change information is propagated to the sites. State changes lost due to communication error is insignificant, since the actual state, not the relative state, are transmitted to all the distributed environments.

4.2 Collaboration Issues

All existing DVE applications allow certain degrees of collaboration among distributed users. Distributed users have been able to see and signal each other through visual gestures in a virtual world [15]. Virtual environments have been synchronized to render the same content in all sites. State changes in one site are propagated to the rest of the connected sites to refresh all local state. One of the most powerful features of a DVE application is the exchange of objects among distributed users. It allows true collaboration among distributed users [10]. However, most of the DVE work supports a sole manipulator and passive observers only. Allowing only one user to control the DVE imposes a great limitation on the level of interaction and collaboration.

Our system provides all users with an equivalent interaction priority in the shared virtual environment. This feature allows a more free and equal collaboration capability for all distributed users to share their opinions about certain objects in the DVE. At any moment, one user can interact with the visualized data by using the two-dimensional control widget while the other distributed user can observe the manipulation process. The current manipulator of the DVE can pass the control widget to other user in the DVE. This enables the receiving user to manipulate and interact with the DVE. Only the user who has the control widget in hand can transfer its control to other users in the DVE.

During the transfer process, the controlling user relinquishes the control to the other user in the virtual environment. After the transfer command has been issue, the controlling user's application will send a command to the other user's application to activate the virtual menu of that user. The other user's application, upon receiving the command, will attempt to turn on the virtual menu of the user. If the virtual menu is successfully turned on, the application will then send a command to disable the controlling user's virtual menu. The disable command must be acknowledged by the controlling user's application. If the acknowledgment was not received from the controlling user, the disable command will be resent until an acknowledgment is received. The protocol ensures that at least one user will have a virtual menu on the left hand. This guarantee is important because all interactions with the application are accomplished through the virtual menu. This protocol also guarantees that only one user can have access to the virtual menu at any given time.

4.3 Viewpoint Control

After the initial testing of the prototype application, we found that when distributed users were exchanging opinions about an object, they were occasionally discussing two different objects. This situation occurs because every user has different viewpoints. To eliminate this problem, we designed our system to provide a feature that will give all users a coherent viewpoint. The manipulator of the virtual environment can synchronize all viewpoints to a temporarily coherent viewpoint.

This methodology can guarantee that all distributed users are observing and discussing exactly the same object in the DVE. The controlling user can activate temporarily coherent viewpoints and restore the original viewpoints through the virtual menu. This feature enables the user who has the virtual menu on hand to show the other distributed users the viewpoint of interest and guarantees that distributed users are observing the object from the exact same viewpoint and discussing about the same issue. However, the user that has the control of the widget will also have to reset the viewpoint to its original settings before relinquishing the virtual menu to another user.

5. Future work and Conclusion

The current implementation of DPW supports two simultaneous users. However there is no pre-determined limitation of the number of simultaneous users that DPW can support. The user process of DPW can be easily modified to support more than two simultaneous users. The limitation on the number of users will depend only on the network bandwidth that is available to support an acceptable real time interaction among the distributed users.

Future development planned for this DVR application includes the verification of the usefulness of the features supported in DPW. We plan to run human subject to determine the usefulness of the viewpoint control and virtual menu transfer features.

DPW provides a homogenous collaborative working environment for remotely located scientists to cooperate designing a new drug, new gas ...etc. With team members scattered all over the globe, remote

collaboration should substantially reduce the turnaround time in the design phase. DPW can also be use as a distance-learning tool. Both an instructor and a student can be immersed in a virtual environment at the same time to examine an object. The teacher can illustrate the construction of a molecular structure in a way never before possible to the trainee even in a virtual world.

Acknowledgement

The first author would like to thank Dr. R. Bowen Loftin for his support and guidance over the past years.

This material is based upon work supported in part by the National Science Foundation under Grant No. NEC95-55682, NASA grant NAG9-985, and funding from the Institute of Somatic Sciences. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding agencies.

Reference

1. A. Johnson, J. Leigh, and J. Costigan, "Multiway tele-immersion at Supercomputing 97", *IEEE Computer Graphics and Applications*, pp 6-9, (1998).

2. B. Blau, J. M. Moshell, and B . McDonald, "The DIS (Distributed Interactive Simulation) Protocols and Their Application to Virtual Environments," *Proc. Of the Meckler VR 93 Conf*, Mecklermedia Press, Westport, Conn,. (1993).

3. Bowen R. Loftin, "Virtual Environment for aerospace training," *Proceedings of WESCON 1994*, pp. 384-387, (1994).

4. C. Dede, M.C. Salzman, and R.B. Loftin, "ScienceSpace: virtual realities for learning complex and abstract scientific concepts," *Proceedings of Virtual Reality Annual International Symposiom*, pp246-252, 271, (1996).

5. C. R. Karr, D. Reece, and R. Franceschini, "Synthetic soldiers [military training simulator]," *IEEE Spectrum*, pp39-45, (March 1997).

6. J. Leigh, A.E. Johnson, "CALVIN: an immersimedia design environment utilizing heterogeneous perspectives," *Proceedings of the Third IEEE International Conference on Multimedia Computing and System*, pp. 20-23, (1996).

7. J. Leigh, A. E. Johnson, C. A. Vasilakis, and T. A. DeFanti, "Multi-perspective collaborative design in persistent networked virtual environments," *Proceedings of the IEEE 1996 Virtual Reality Annual International Symposium*, pp. 253-260, 271-2, (1996).

8. LinCom Corporation, VrTool User's Guide and VrTool Application Programmer's Interface Functions, LinCom Corporation, 1020 Bay Area Blvd, Suite 200, Houston, TX 77058-2682, pp. 196-201, (1995).

9. Martin. R. Stytz, J. Vanderburgh, and S.B. Banks, "The Solar System Modeler," *IEEE Computer Graphics and Applications*, pp. 47-57, (Sept.-Oct. 1997).

10. Martin R. Stytz, E. G. Block, B. B. Slotz, and K. Wilson, "The Synthetic Battle bridge: a tool for large-scale Ves," IEEE Computer Graphics and Applications, pp. 16-26, (1997).

11. R. R. Mourant, N. Qiu, and S.A. Chiu, "A distributed virtual driving simulator," *IEEE Virtual Reality Annual International Symposium*, pp 208, (1997).

12. S. Stansfield, D. Shawver, N. Miner, and D. Rogers, "An application of shared virtual reality to situational training," *Proceedings of Virtual Reality Annual International Symposium 1995*, pp 156-161, (1995).

13. S. Stansfield, D. Shawver, and A. Sobel, "MediSim: a prototype VR system for training medical first responders," *Proceedings of IEEE Virtual Reality Annual International Symposium 1998*, pp 198-205, (1998).

14. T.W. Mastaglio, and R. Callahan, "A large-scale complex virtual environment for team training," *Computer*, pp 49-56, (July 1995).

15. Un-Jae Sung, Jae-Heon Yang, and Kwang-Yun Wohn, "Concurrency control in CIAO," *Proceedings of IEEE Virtual Reality 1999*, pp 22-28 (1999).

16. V. D. Lehner and T. A. DeFanti, "Distributed virtual reality supporting remote collaboration in vehicle design," *IEEE Computer Graphics and Application*, Vol. 17, Issue 2, pp. 13-17, March-April (1997).

17. W.D. McCarty, S. Sheasby, P. Ambrun, M. R. Stytz, and C. Switzer, "A virtual cockpit for a distributed interactive simulation," *IEEE Computer Graphics and Applications*, pp49-54, (Jan. 1994).