Collaborative Virtual Environment Standards: A Performance Evaluation

Jauvane C. de Oliveira*, Shervin Shirmohammadi, and Nicolas D. Georganas
Multimedia Communications Research Laboratory
School of Information Technology and Engineering
University of Ottawa, Ottawa, Canada
[jauvane | shervin | georgana]@mcrlab.uottawa.ca

Abstract

Collaborative Virtual Environments are virtual reality spaces that enable participants to collaborate and share objects as if physically present in the same place. These collaboration spaces require strict performance characteristics. Today there are many systems developed specifically for collaboration, including DIVE, CALVIN, and COVEN. At the same time, some relatively new standards that address multiuser virtual environments and shared spaces have become available. In this paper, we're going to evaluate available and emerging standards such as the High Level Architecture (HLA) and Open Community to determine if they appropriately address the needs of such environments.

1. Introduction

Collaborative Virtual Environments (CVE) are used for applications such as collaborative design, training, and tele-robotics. Unlike other types of simulations, there are not many people participating in such environment at the same time. Typically it is expected to have about 5 to 15 people in one such session, these are engineers designing a new engine, or new department employees virtually training with equipment.

Today there are some dedicated systems supporting these types of environment. The Distributed Interactive Virtual Environment (DIVE) is an Internet-based multi-user VR system developed at the Swedish Institute of Computer Science which allows its participants to navigate in 3D space and see, meet and interact with other users and applications [5]. COVEN is a four-year European project that was launched with the objective of comprehensively exploring the issues in the design, implementation and usage of multi-participant shared virtual environments, at scientific, methodological and technical levels [11]. CAVERN, the CAVE Research Network, is an alliance of industrial and research institutions equipped with immersive equipment and high-performance computing resources all interconnected by high-speed networks to support collaboration in design, training, visualization, and computational steering in virtual reality [10].

Almost in parallel, standards are being developed by the VR/simulation community in order to facilitate the development of CVEs in a systematic manner. As we will see next, some of these standards/libraries are very generic while others make certain assumptions. Also the area of interest for each standard varies from graphics and rendering to simulation and network communication. Standards are necessary in order to construct a hybrid collaborative VR system, reason why the currently available standards have to be evaluated aiming the identification of problems to be addressed in future revisions of the standard or the development of new standards. The rest of this document is organized as follows: section 2 gives a brief introduction to some of the CVE and related standards, section 3 introduces the prototypes that were developed for testing, and section 4 describes the performance evaluation methods and results.

2. The Standards

The standards can be divided into two categories: *Rendering and Graphics* and *Communications Middleware*.

2.1 Rendering and Graphics

These standards are mainly concerned with graphics and rendering of 3D scene graphs. They have very little or no consideration about communication between users and other networking related issues.

* Sponsored by CAPES, the Brazilian Ministry of Education Agency: Bolsista da CAPES - Brasilia/Brasil

OpenGL[1]

OpenGL is a graphical library introduced by Silicon Graphics Inc. (now SGI) in 1992 to allow developers to write a single piece of code, based on the OpenGL API, which is supposed to run in various platforms (as long as they have an OpenGL library implementation). Since its inception OpenGL has been controlled by an Architectural Review Board [15] whose representatives are currently from the following companies: 3DLabs, Compaq, Evans & Sutherland, Hewlett-Packard, IBM, Intel, Intergraph, NVIDIA, Microsoft, and SGI.

Most of the Computer Graphics research (and implementation) broadly uses OpenGL which has became a de facto standard. Virtual Reality is no exception to this rule.

There are many options available for hardware acceleration of OpenGL based applications. The idea is that some complex operations may be performed by specific hardware (an OpenGL accelerated video card such as those based on 3DLab's Permedia series or Mitsubishi's 3Dpro chipset for instance) instead of the CPU which is not optimized for such operations. Such acceleration allows low-end workstations to perform quite well yet at low cost.

We will see that such is the importance of OpenGL that both VRML and Java3D are built on top of it, i.e. if a given workstation has hardware support for OpenGL the VRML browser and Java3D, discussed below, will also benefit from it.

DirectX

Microsoft DirectX® is a group of technologies designed by Microsoft to allow Windows-based computers to run and display applications rich in multimedia elements such as full-color graphics, video, 3-D animation, and surround sound. DirectX is an integral part of Windows 98 and Windows 2000, as well as Microsoft® Internet Explorer 4.0. DirectX components may also be installed in Windows 95 as an optional package.

DirectX allows a compliant application to run in any Windows based system, independent of particularities of hardware of each system. In some sense it seems similar to OpenGL; however there is a logical limitation in disponibility as it is a Windows specific component. DirectX accomplishes its task via a multilayered structure. The Foundation layer is responsible for resolving any hardware dependent issue. DirectX also allows developers to deploy creation and playback of multimedia content via DirectX's Media layer. A third layer, Component, complets the high level protocol layer stack.

Some VRML browsers also provide a Direct3D based version (as well as the common OpenGL). A good example is blaxxun's Contact 4.0 [17].

VRML

VRML 2.0, which is the latest version of the well-known VRML format, is an ISO standard (ISO/IEC 14772-1:1997). Having a huge installed base, VRML 2.0 has been designed to support easy authorability, extensibility, and capability of implementation on a wide range of systems. It defines rules and semantics for presentation of a 3D scene. Using any VRML 2.0 compliant browser, a user can simply use a mouse to navigate through a virtual world displayed on the screen. In addition, VRML provides nodes for interaction and behavior. These nodes, such as *TouchSensor* and *TimeSensor*, can be used to intercept certain user interactions or other events which then can be *ROUT*ed to corresponding objects to perform certain operations. Moreover, more complex actions can take place using *Script* nodes which are used to write programs that run inside the VRML world. In addition to the Script node, VRML 2.0 specifies an *External Authoring Interface* (EAI) which can be used by external applications to monitor and control the VRML environment. These advanced features enable a developer to create an interactive 3D environment and bring the VRML world to life.

Java 3D

Java3D is part of the Java Media APIs developed by Javasoft. Providing developers with high level constructs for creating and manipulating 3D geometry in a platform-independent way, the Java 3D API is a set of classes for writing three-dimensional graphics applications and 3D applets. Since a Java3D program runs at the same level as any other Java program/applet in the virtual machine, controlling the virtual world becomes very easy through calling the Java3D API from any Java program. Although Java3D and VRML both appear to target the same application area, they have fundamental differences. VRML is aimed at a presentational application area and includes some support for runtime programming operations through its External Authoring Interface and the Script node, as mentioned earlier. Java3D; however, is specifically a Java language API, and is only a runtime API. Java3D does not define a file format of its own and is designed to provide support for applications that require higher levels of performance and interactivity, such as real-time games and sophisticated mechanical CAD applications. In this sense, Java3D provides a lower-level, underlying platform API. Many VRML implementations can be layered on top of Java3D. In fact, it is possible to

[1] OpenGL™ is a registered trademark of SGI.

write a VRML browser using Java3D, such as the browser developed by VRML consortium's *Java3D and VRML Working group* [14].

2.2 Communications Middleware

Standards in this category focus on the issues concerning connecting users together on the network to create shared worlds. Although some of them also provide graphical capabilities, their main target is networking, world status updating, and object-sharing capabilities.

Living Worlds

Living Worlds (LW) [13] is a Working Group of the VRML Consortium, supported by a large number of organizations. The LW effort aims to define a set of VRML 2.0 conventions that support applications which are multiuser and interoperable. "Scene-sharing", which is concerned with the coordination of events and actions across the network, is one of the main elements of LW. In short, LW is a first attempt to devise a common VRML 2.0 interface to support basic interaction in multi-user virtual scenes and enables each participant to know that someone has arrived, departed, sent a message or changed something in the scene. Although LW specifies rules for object sharing and exchanging update messages across the network, it is not a communications middleware and the reason it is being presented in this section is that it is also not concerned with graphics and rendering. In fact LW does not care about the actual technical implementation of the communications system that enables world sharing. Referred to as the Multi User Technology (*MuTech*), the actual system that runs on the network and is responsible for message passing among clients can be developed by any technology as long as its interface to the VRML world follows the LW specifications. Since currently no implementations of LW are publicly available, it was not used in our performance tests.

Open Community

Open Community (OC) is a proposal of standard for multiuser enabling technologies from Mitsubishi Electric Research Laboratories. SPLINE (Scalable Platform for Large Interactive Networked Environments) [6] is an implementation compliant with OC which provides a library with ANSI C and soon Java API. Such library provides very detailed and essential services for real-time multi-user cooperative applications. For its communication, SPLINE uses the Interactive Sharing Transfer Protocol (ISTP) [7] which is a hybrid protocol supporting many modes of transportation for VR data and information, namely:

- 1-1 Connection Subprotocol: Used to establish and maintain a TCP connection between two ISTP processes;
- Object State Transmission Subprotocol: Used to communicate the state of objects from one ISTP process to another. Such updates may be sent via 1-1 Connection or UDP Multicast (1-1 Connection is used as a backup route);
- Streaming Audio Subprotocol: Used to stream audio data via RTP;
- Locale-Based Communications Subprotocol: This is the core of ISTP and supports the sharing of information about objects in the world model; and
- Content-Based Communication Subprotocol: Supports central server style communication of beacon information. Beacons are tags through which is it possible to retrieve a given object (similar to an URL).

The last two subprotocols are build upon the other three. ISTP does not provide video-streaming capability to date, however such support could be provided by extending ISTP with an extra appropriated subprotocol.

High Level Architecture (HLA)

HLA is a framework for distributed simulation systems developed by the U.S. Defense Modeling and Simulation Office (DMSO). HLA attempts to provide a very generic environment that any virtual object can attach to in order to participate in a simulation. It is a very well-thought architecture that defines standard services and interfaces to be used by all participants in order to support efficient information exchange. HLA is adopted as the facility for *Distributed Simulation Systems 1.0* by the Object Management Group (OMG) and is now in the process of becoming an open standard through the IEEE. HLA's Runtime Infrastructure (RTI) is a set of software components that implement the services specifies by HLA. Today, a few RTI implementations for different platforms are available. For this paper, we're using version RTI 1.0 release 3.

Java Shared Data Toolkit (JSDT)

Although not specifically designed for 3D simulations and virtual environments, JSDT is part of the Java Media APIs developed by Javasoft and provides real-time sharing of applets and/or applications. JSDT provides many facilities such as *token*s that can be used for coordinating shared objects. It also provides different modes of transportation including a reliable *socket mode*, *RMI* mode which uses Remote Method Invocation, and a multicast mode that makes use of the *Lightweight Reliable Multicast Protocol* (LRMP) and is useful for shared application with a large number of participants.

3. Prototypes

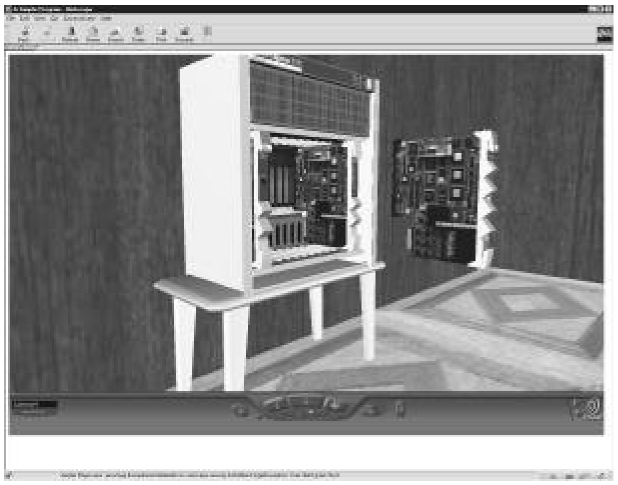

Figure 1. A Virtual ATM switch with one of its cards pulled out, running in Netscape Navigator.

We built a few prototypes based on the above middleware/libraries. The application of interest was a training application where an instructor teaches one or more trainees how to configure a specific ATM switch. This application requires coordination of interactions with different components of the ATM switch. In this example the shared objects were the ATM cards that had to be inserted or removed for configuration. A screen shot of a sample session is shown in Figure 1. Figure 2 shows a SPLINE session of the same prototype using OpenGL as a rendering engine.

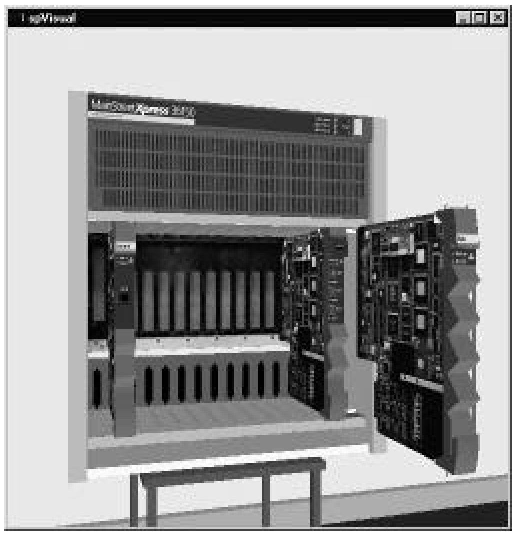

Figure 2: Virtual ATM Switch through SPLINE using OpenGL for rendering

Figure 1 shows the virtual training environment running in Netscape Navigator's VRML 2.0 plug-in. To join the session, participants navigate to a pre-defined URL address. The browser then downloads the necessary HTML files, containing the Java applets and the VRML world. The VRML world is controlled by a Java applet trough EAI, as explained before. Since both Open Community and RTI 1.0 release 3 provide a C API at this time, the Java Native Interface (JNI) was used to connect the Java applet to the communications middleware.

3.1 Java3D/VRML - RTI

Technically speaking, RTI is a distributed system comprised of two global processes, the RTI executive (*rtiexc*) and the Federation Executive (*fedex*). *rtiexec* manages the creation and destruction of the federation executions. *fedex* is a global process per federation execution that manages the joining and resigning of federates in an execution. A Java applet is used to control the VRML scene via EAI and to communicates with the RTI through JNI. Different objects are used to embody a certain entity in the virtual environment at each of the three layers of the application – VRML, Java and RTI. At the VRML layer, a VRML *PROTOTYPE* node defines the avatar's geometry and the animations of its actions/gestures which are implemented by VRML sensors, events and scripts. At the Java level, a Java object consists of attributes like position, rotation and gesture to describe the avatar in CVE. Finally, simulated entities in the RTI context are represented by RTI objects. All possible interaction between the federates of a federation must be pre-defined in a so-called Federation Object Model (FOM). Figure 3 illustrates the model of the system.

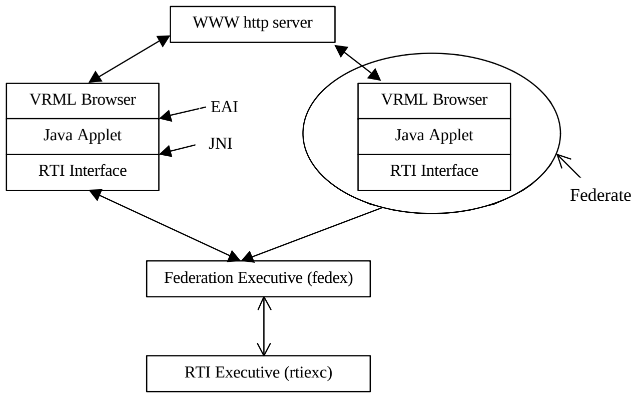

Figure 3. The RTI-based hybrid system model of the collaborative environment.

3.2 VRML - Open Community

SPLINE consists of the SPLINE Universal Server (*spurs*) and the SPLINE library. *spurs* manages the World Model with all its *locales* and enables ISTP compliant communication by controlling limited resources such as multicast addresses. Locales define a partition of the World Model and a given process only receives updates regarding objects within its own locale as well as the immediate neighborhood locales, which is accomplished by attributing a different multicast group for each locale. This way the amount of information a given station has to deal with is limited to those contained in few locales. Locales are pretty much like federations in HLA. SPLINE library is linked to the local client code and implements the client side of the ISTP protocol as well all functions/methods of the API.

Currently only an ANSI C API is available, hence the same procedure previously mentioned for RTI has to be followed in order to achieve rendering of VRML content via a VRML browser; i.e., a Java Applet controlling the VRML content via the External Authoring Interface communicates with SPLINE's ANSI C code (linked to SPLINE library *spLib*) via JNI. To keep the World Model up to date one must call the *spWMUpdate* method (World Model Update). Calling this method forces every action performed in objects owned by the current process to be sent to the appropriated multicast address and at the same time every process's modification to its objects is received. As this calling is made periodically the WM is reasonably up to date for every process. Figure 4 shows this model.

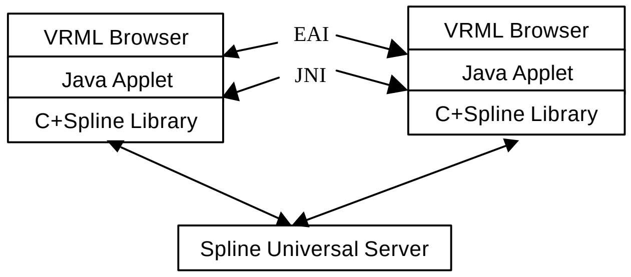

Figure 4. The SPLINE-based hybrid system model.

4. Performance Evaluation

In terms of a performance model, a virtual environment can be divided into four high-level components [8] each adding a delay to the overall system: 1) display – the “on-screen” delay created by monitors, projectors, stereo goggles, and other displaying equipment; 2) graphics – the delay caused by the graphics card/drivers and other graphical component of the system; 3) simulation: the lag introduced as a result of processing/computing performed by the core multi-user engine hardware and software; 4) Interconnections: networking and communication delays.

4.1 Parameters of Interest

Delay and jitter are the most important parameters in CVEs. Among these, delay has been studied more extensively with actual numbers available which indicate acceptable levels. The acceptable simulation + network delay is less than 200 msec [3], with some studies suggesting a more restrictive 100 msec [4]. Jitter is also an important factor, sometimes believed to be more important than delay when it comes to coordinating collaborative tasks [2]. But no actual numbers for it are available in terms of what constitutes an acceptable level of jitter.

In our evaluations, we combine simulation and network delays as the *client-to-client delay* (CCD). This is the average time it takes for a given update message at one client to be assembled by the simulation, sent over the network, received by another client, and disassembled to be processed by the graphics/display components of the system - it doesn't include the rendering and graphics delay. In the case of the RTI prototype, for example, this delay includes all layers of Java, JNI, C++, RTI, and physical network delays.

Another parameter of interest is the *Ownership Transfer Delay* (OTD). Both SPLINE and RTI allow only the owner of an object to manipulate it although everyone is the session will be able to see those manipulations. Ownership of an object must first be acquired in order for someone to manipulate the object. This is similar to the concept of *floor control* in conferencing [9] or 2D collaborative applications where only one person at a time can interact with a shared entity in order to avoid event collision and unwanted conflicts [1]. This transfer of ownership creates an additional delay. When the user clicks on an object to manipulate it, the system must first obtain ownership. This delay should be small enough to maintain a natural feeling in the virtual world.

Both RTI and SPLINE only allow the owner of a given object to handle it and the ownership transfer is performed in a two steps fashion, i.e. a given process requests it and the current owner grants it or not. In fact SPLINE allows unidirectional transfer when the owner gives the object to another process without that one requiring it; however this procedure is not common and is commonly used when a given process will end and wishes to leave in the world some objects it owns. The ownership transfer goes through four different queues which increase considerably the delay. Again taking SPLINE as an example the first queue is located at the requester’s station when the request is enqueued to be sent in the next cycle as only when spWMUpdate is called outcoming messages are effectively sent. The same thing happens when the request arrives at the current owner of the object and stays in the queue until the next cycle, when a callback is performed to handle the request. Once the ownership request is granted this message has to be sent back to the requester, not before staying in two queues again in its way back. So, the ownership transfer involves 4*queue delay + 2*communication delay + 2*processing time. Taking this into account if spWMUpdate is set to be called every 50ms the average waiting time should be of about half of that (25ms) in average. So, one should expect an average 100ms just concerning queueing delay.

4.2 Other parameters

There are some other parameters that affect the performance of the overall system. We don't measure them in this paper either because we can't control them or because they are not very significant. *Object Insertion Delay* measures how long it takes for a newly created object to appear on every participant's screen. This is however more a function of the number and size of the attributes of the object than anything else since the object must initially be downloaded over the network. If the object is local, than this delay will normally be insignificant.

Framerate measures how quickly a user can navigate and interact with objects from a graphical stand-point. It depends on the complexity of the virtual environment and the hardware support: a large room with lots of texture maps runs slower than a small room represented by primitive objects. Subjectively, we observed that the virtual world ran faster in the Java3D environment than in the VRML plug-in. Again taking SPLINE as an example the frame rate is also set via the update rate as usually at every cycle the display is refreshed. I.e. at 100ms update one may expect to have about 10fps, which is not guaranteed because SPLINE runs to completion (if it needs to take longer to finish all outstanding tasks it will do so, reducing the "framerate").

4.3 Testing Procedure

For the CCD test, we had an "initiator" client create an object and perform an update on this object, such as a rotation or translation. A second client then receives this update and extracts its information. Then, the second client performs a similar update on its object which is "seen" by the initiating client. The initiating client again performs an update on its object, and so on. This procedure is repeated for a given number of times or duration, which was about 5 minutes in our tests.

For the OTD tests, we had two clients obtaining and releasing the ownership of an object consecutively and in turn over a period of time. The average time to obtain ownership was then calculated. The application consisted of a couple of stations running the simulation where one started as the owner of one of the cards in the switch. The other client started the loop by asking for the ownership of the card. From this point onwards the current owner granted automatically the request in the appropriated callback and asking for the ownership of the same card right away. Such loop was run for a pre determined time interval and the delay averaged afterwards.

For RTI the reliable transport mode was used. Although slower than the "best effort" mode, reliable transportation of update data to each client is a necessary requirement of collaborative environments. The reason is that during the session, any user interaction with an object occurs in a short duration. In our example, the action of pulling out a card from the switch occurs quickly and must be reflected to all other users reliably. The well-known arguments of "dead-reckoning" and "frequent sending of position-updates" that justify best-effort communications in other simulations do not apply to coordinating collaborative tasks as described above. Even in the case of multicast, reliable multicast such as LRMP must be used for CVEs.

The ISTP protocol used in SPLINE didn't allow us to choose a transportation mode as it automatically uses an appropriate mode with regards to the data being sent. However ISTP provides the necessary reliability and error correction for its operations.

4.4 Results

All tests were performed on Pentium II class PCs running Windows NT 4.0 and connected to a 100 Mbps Ethernet. The graphics was set at 1280X1024 24bit color screen resolution with 3D accelerator cards. All machines were running typical operating system and networking background processes.

As previously mentioned SPLINE allows one to set the update rate via spWMSetDesiredInterval(). We then performed an extensive evaluation of delays based on his feature. The Figure 5 below shows the delays for the various update rates. The darker area shows delays larger than the usually acceptable 100ms [3] delay. However some [4] suggest 200ms as a limit for the end-to-end delay. It is important to remember that the graph of Figure 5 shows the communication delay, where additional delays, as commented in section 4, still need to be considered.

Regarding the four-queueing delay expected to incur in each Ownership Transfer we have modified the simulation to grant the ownership of the object (the card from the switch) and ask for it immediately. I.e. both messages were called between spWMUpdate, which makes both to be enqueued and sent together at the next cycle, reducing the end-to-end latency. One may argue that commonly such requests are performed separately, which would increase the presented delay by 2* queueing time in average. A last remark about the graph above is that instead of falling smoothly in the 0-25 interval the graph should keep declining at similar rate until the minimum is reached at about 12-15 ms refresh interval (about 66 frames per second). I.e. 1ms update certainly doesn't mean that 1000 frames are show every second but the maximum possible remembering the run to completion rule of SPLINE mentioned in section 4.2. The thin line in the graph displays the correction according to this rule.

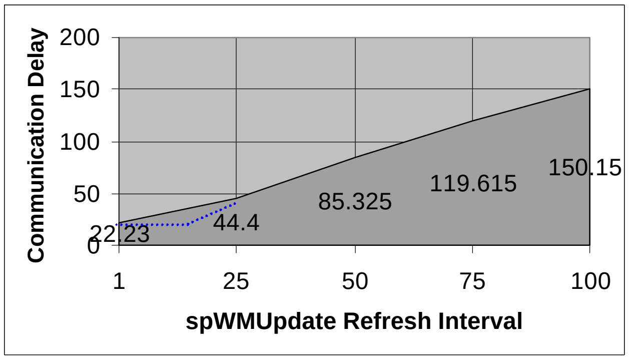

Figure 5: The ownership transfer delay for SPLINE as a function of update rate

The results shown in Table 1 below shows the remaining performance results we got from our measurements.

Table 1. Performance results for various middleware (All delays are in msec)

	CCD	Com. Delay*	OTD
RTI with VRML	184	103	235
RTI with Java3D	152	103	204
JSDT	-	60	5
SPLINE with VRML 100ms update	129	85	217
SPLINE with VRML 1ms update	68	12	78
* Communication Delay: this is CCD minus the delays between the middleware and the rendering part.			

The version of SPLINE used (3.0 beta 2) uses multicast extensively, reducing delays in all operations as the messages are sent through the appropriated multicast group directly to all relevant participants. The latest Beta 3 does not use Multicast, as it aims to run through the Internet as of today with many routers without multicast support. The lack of multicast adds considerably extra delays as ownership transfers, for instance, need to be sent to a server and then forwarded to the destination. I.e. four spWMUpdate cycles will be required with 8 queueing delays. Some early evaluation has show that at 100ms update rate (the server uses a fix 50ms update rate) there is almost 500ms delay instead of the 150ms observed with the multicast enabled implementation of it. This shows the huge importance of multicast, which is be enabled in future implementation of SPLINE.

5. Conclusion and Future Work

From our delay evaluation we showed that using available middleware and standards-based applications, it is feasible to construct CVEs with acceptable quality at least for a small group of users in a controlled environment. However as the number of users rises to large or very large numbers it will be necessary to have appropriated mechanisms to support them efficiently. We are going to write a simulation tool to evaluate the performance of the existent standards for very large environments where bottlenecks will be identified and avoided via a modified model suitable for such use.

By adjusting the updating rate in SPLINE, it is possible to reduce the amount of computing and networking resources required at the cost of higher delays and vice-versa. SPLINE does provide the flexibility to achieve a desired delay by allowing the designer to adjust the updating rate. At the common 100 msec interval the observed delay was 129 msec, which is acceptable. RTI also exhibits acceptable performance.

JSDT seems to be the fastest at a reasonable refresh rate; however as mentioned earlier, JSDT is not designed for 3D virtual environments and lacks many of the necessary services for such applications. As a result, the developer needs to design and implement those services, usually in an ad-hoc manner.

The delay caused by the Java applet-to-JNI-to-native interface measured about 50 to 80 msec. An architecture such as Java3D running on top of JSDT would eliminate this delay and would be an alternative for time-critical and other low-delay demanding applications.

Although the test results for the delay parameter are less than the 200 msec requirement mentioned before, there is some room for concern. Considering the fact that the tests were performed between only 2 users and on a local-area network with light local traffic, and that the results are sometimes very close to the allowable thresholds, scalability and delay for a larger number of users connected by internetworks is questionable. Further tests focusing on the scalability aspect of the prototypes are required.

Another issue which shall be taken into account is that there can be more than one object change at the same time. In an assembling scenario, one person can be moving parts out of the box while another is tightening the screw between two parts, or simply moving its avatar for one position to another. It is then necessary to measure how many simultaneous object changes can be supported with still acceptable delays. These changes must be relevant to the specific collaboration of course and will be addressed in future work. The simulation tool will allow plenty of parallel modifications in the world, where much detailed results will be available.

Today, computing equipment available to ordinary users is becoming more advanced and more affordable. This means that it will be more feasible to run CVEs on typical computing systems. Adhering to standard enables a broad range of collaborative application to be accessed by the general public.

Acknowledgements

The authors acknowledge the research and development contributions of Xiaojun Shen and Pierre Desmarais, as well as the financial assistance of Newbridge Networks and kind contributions of Mitsubishi Electric Research Lab. We also acknowledge the financial assistance of the Brazilian Ministry of Education Agency's CAPES scholarship, and the Natural Sciences and Engineering Research Council of Canada (NSERC) Scholarship Program.

References

[1] H. P. Dommel and J. J. Aceves, "Floor Control for Multimedia Conferencing and Collaboration", ACM Multimedia Systems, Vol. 5, No. 1, 1997, pp. 23-38.

[2] Jason Leigh et al, "Preliminary STAR TAP Tele-Immersion Experiments between Chicago and Singapore", High Performance Computing Asia Conference & Exhibition, Singapore, 1998.

[3] K. S. Park, *Effects of Network Characteristics and Information Sharing on Human Performance in COVE*, Master's thesis, Electronic Visualization Laboratory, University of Illinois at Chicago, 1997.

[4] Mathias M. Wloka, "Lag in Multiprocessor VR", Presence: Teleoperators and Virtual Environments (MIT Press), Vol. 4, No. 1, Spring 1995.

[5] Olof Hagsand, "Interactive Multi-user VEs in the DIVE System", IEEE Multimedia, pp. 30-39, Spring 1996.

[6] R.C. Waters and J. Barrus, "The Rise of Shared Virtual Environments", IEEE Spectrum, March 1997.

[7] R.C. Waters, D.B. Anderson and D.L. Schwenke, ``Design of the Interactive Sharing Transfer Protocol", Proc. IEEE Sixth Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 1997.

[8] V. E. Taylor, R. Stevens, and T. Canfield, "Performance Models of Interactive, Immersive Visualization for Scientific Applications", Proc. International Workshop on High Performance Computing for Computer Graphics and Visualization, Swansea, United Kingdom, 1995.

[9] Oliveira, J. C., "TVS: A Videoconferencing System", M.Sc. Thesis (in Portuguese), Computer Science Department, Pontifical Catholic University of Rio de Janeiro, Brazil, August 1996.

[10] CAVERN , http://www.evl.uic.edu/spiff/covr/

[11] COVEN, http://chinon.thomson-csf.fr/projects/coven/

[12] HLA, http://www.dmso.mil/hla/

[13] Living Worlds, http://www.vrml.org/WorkingGroups/living-worlds/

[14] The Java3D and VRML Working Group, http://www.vrml.org/WorkingGroups/vrml-java3d/

[15] OpenGL Architecture Review Board, http://www.opengl.org

[16] Microsoft DirectX, http://www.microsoft.com/directx

[17] blaxxun Contact 4.0, http://www.blaxxun.com/products/contact/