

Realizing multi-user interactive shared space using VRML.

Kouichi Matsuda, Rodger Lea, Yasuaki Honda
Software Laboratory, Architecture Laboratories, Sony, Tokyo, Japan
Tel: +81 3 5448 5437 email: matsuda@arch.sony.co.jp

Abstract

The Virtual Society project is a research initiative to investigate the future electronic society. Our vision for such a society is based on the notion of shared multi-user 3D virtual worlds where people can meet, interact and work. People will also be able to enjoy multiple lives in the world and take on any of a range of personalities and roles. We have developed the CommunityPlace system to realize an initial implementation of our vision for the Virtual Society. It is targeted at the WWW and uses the Virtual Reality Modeling Language (VRML). We believe that such a system will become the next generation user interface over the Internet. In this paper, we discuss the architecture and implementation of the CommunityPlace system, its performance and our future plans.

Keywords: Distributed Virtual Environment Internet VRML

1. Introduction

The term "Cyberspace" although popularized by the media exists today only in science fiction novels. However, the recent evolution of computer technologies and network technologies provides the enabling infrastructure for cyberspace. The goal of Virtual Society project is to provide the cyberspace, or a massive shared multi-user 3D virtual world on WWW. People can access the world from anywhere in the Internet, from homes, schools and offices. We wish to create worlds where many people can meet, interact and work together and investigate how the future electronic society will evolve. Also people will be able to enjoy multiple lives in the world. The investigation of such people's activities in the world will become a cyberspace cultural anthropology in the future. We also believe that such a world will become the next generation of the user interface over the Internet.

This project started in the middle of 1994 at Sony Computer Science Lab. The first goal was to build a support infrastructure that allowed many people to participate in a shared, interactive 3D world. Such interaction will include the ability to see each other, talk to each other, visit locales with each other and share the events happen in the shared world. The CommunityPlace system has been developed in the project as the support infrastructure for an initial version of Virtual Society. The first release (based on VRML1.0) was at the end of 1995; subsequently we have migrated to VRML2.0 and made a version of the system available since early 1996.

2. CommunityPlace

CommunityPlace (previously known as CyberPassage) is the name of a suite of software developed by Sony's Architecture Laboratories to realize the Virtual Society. It is targeted at home PC's, dial up lines and the Internet. It consists of four software components; a 3D browser known as CommunityPlace Browser, a 3D scene and behavior construction tool, known as CommunityPlace Conductor, a multi-user server known as CommunityPlace Bureau and an application environment known as AO. To construct virtual worlds on WWW, we chose the VRML as a 3D file format.

CommunityPlace Browser is a PC based browser to display and navigate through virtual 3D worlds. It supports VRML1.0 and VRML2.0 as a 3D file format and Java as a scripting language to describe behaviors. In addition, it communicates with the CommunityPlace Bureau to support shared 3D worlds by using a client server protocol Virtual Society Client Protocol (VSCP) to send and receive updates about changes to scene geometry. In the shared worlds, The CommunityPlace browser supports on-line text chat and audio chat. The CommunityPlace browser can work as both a helper application and plug-in for Netscape Navigator. Currently Windows 95/NT versions are freely available for download from our web site. Figure 2.1 shows the CommunityPlace Browser connecting to a multi-user world.

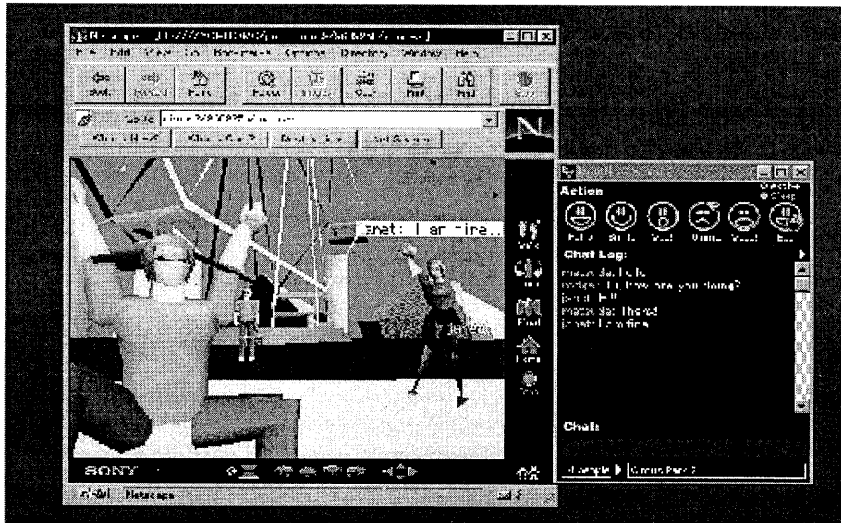


Figure 2.1 CommunityPlace Browser connecting to a multi-user world

CommunityPlace Conductor is a PC based authoring tool to create virtual 3D worlds. It is a kind of integration tool and does not support 3D modeling features. Rather it is used to compose scenes from 3D objects created in modeling tools to which can be added multimedia data (sound and video), textures and behaviors. This approach allows the creation of interactive, multi-media 3D scenes or worlds. Community Place conductor supports VRML1.0 and VRML2.0 as its 3D file format and uses Java as a scripting language. Currently Windows 95/NT versions are available.

CommunityPlace Bureau is a PC and UNIX based server system to share virtual 3D world among users. It receives 3D geometric changes and messages from user's browser or external applications for the shared 3D worlds and distributes these changes and messages to other users so that they can visualize the changes. Currently Windows 95/NT and various flavors of UNIX versions are available. The UNIX version can support up to 600 users simultaneously for one world.

The application environment (AO) is a separate component that provides an API suitable for building 3D applications. The AO connects to the server and registers any application objects created via the AO API. These are then dynamically added to the shared 3D scene.

Both the browser and conductor can run on ordinary home PCs and do not require any special hardware like a data glove, HMD(Head Mounted Display) or 3D rendering hardware. So user can navigate a 3D world with a mouse and the 3D world is rendered in real-time to the computer display by 3D rendering software engine. We refer to this type of Virtual Reality experience as fish tank virtual reality.

3. CommunityPlace system architecture

It is possible to build multi-user 3D applications by using the CommunityPlace system. In this section, we discuss how the CommunityPlace system supports such application development.

In addition to the three components described above, the CommunityPlace system assumes a WWW server and HTML browser. The WWW server manages VRML files which the HTML browser retrieves over the Internet. The VRML files contain a description of the virtual 3D world and multi-user applications. It also has a IP address and port number of the computer where CommunityPlace Bureau is running.

When the HTML browser accesses the VRML file, it is passed to the CommunityPlace browser. Then the CommunityPlace browser analyzes the file and visualizes it. If the VRML file is designed as a multi-user content, the CommunityPlace browser tries to connect to the CommunityPlace Bureau specified in the VRML file.

After connecting to the CommunityPlace Bureau, the rest of the communication between the Bureau and the browser is performed using VSCP and only uses HTTP to access files from WWW servers. The basic function of VSCP is to notify the server about any changes made by the users via their browser and to be notified of any changes made by other browser. This allows any browser which access the same VRML file to connect to the same CommunityPlace Bureau and so share the VRML scene. Using this approach, any VRML scene can be shared.

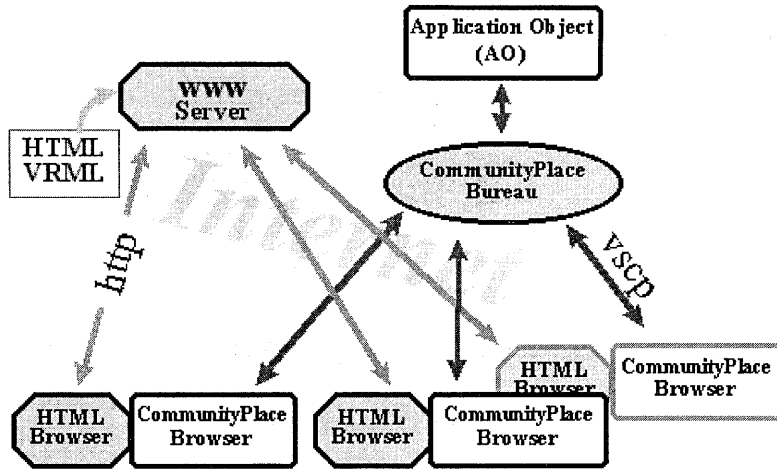


Figure 3.1 CommunityPlace system architecture

In the shared 3D world, each user accessing to the same world is represented as an avatar, or a 3D representation of the user, in the world. Thus, each user's behaviors, e.g. navigating through the 3D scene is reflect by his/her avatar in all other browser sharing the scene.

When a user navigates around the 3D world by moving and rotating his/her avatar, this movement and rotation information is sent from the CommunityPlace browser to the connected CommunityPlace Bureau using the VSCP protocol. Then the CommunityPlace Bureau propagates the information to all other CommunityPlace browser connected to the Bureau. When a CommunityPlace browser receives the information from the server, it updates the corresponding user's avatar according to reflect the changes made by the source browser.

Multi-user applications in the shared world are described using the multi-user application programming model designed for CommunityPlace system. We will discuss the model later.

4. CommunityPlace Bureau

The CommunityPlace Bureau acts as a position tracker and message forwarder in CommunityPlace system. Its primary role to receive the user's position and the information made by the browser and them distribute them to the other browsers. In the current system, due to performance and scalability consideration, the Bureau also has the role to restrict the distribution of information to those browsers that need to know. It uses an Aura [6] algorithms to decide which other browsers need to be sent the information. An Aura is the area around a user that is deemed to be interesting, anything outside the aura is not considered interesting. So the Bureau does not send the information of the browsers outside the aura to the browser which has the aura. This spatial technique allows us to scale the system. Figure 4.1 shows how Aura works. In this case, user A and user B can see each other but they can not see user C and user C can not see them.

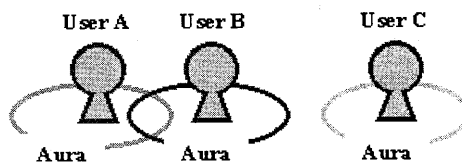


Figure 4.1 Aura

In addition to the aura, the CommunityPlace system supports World Location Server (WLS) to enable to switch multiple Bureaus dynamically. WLS enables to divide one shared world into several sections geometrically and assign the Bureaus to each section. Figure 4.2 shows how WLS works. A user(browser) is in the area1 where the Bureau1 manages and then enters into the area2 where the Bureau2 manages. Bureau1 sends a leaving message to the browser (1) and the browser asks the WLS about the next Bureau to be connected (2) and it sends the information of the Bureau2 (3). Then the browser connects to it (4).

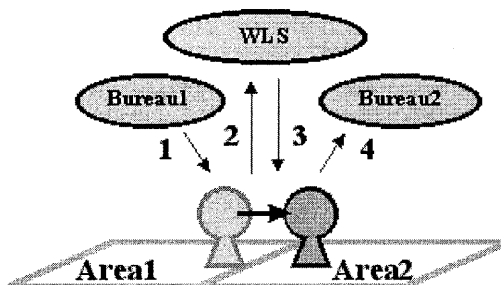


Figure 4.2 World Location Server

5. Multi-user application programming model

The CommunityPlace system provides two types of application programming model for building multi-user applications. The first one is called the Simple Shared Script (SSS) model and the second one is called the Application object (AO) model.

5.1 Simple shared script

The SSS is a simple mechanism designed for small shared application in the 3D world. It uses a replicated script model. So each browser uses the same script and executes it locally. The script can send a message to all other browser sharing the world using VSCP via CommunityPlace Bureau.

Figure 5.1 shows how SSS model works. When a user selects a 3D object(1), a local script associated with the object is executed (2). Then the script converts the event into a message and sends it to CommunityPlace Bureau(3). The Bureau distributes the message to all other browser (4). The browser convert the message into an event that causes the local script execution (5).

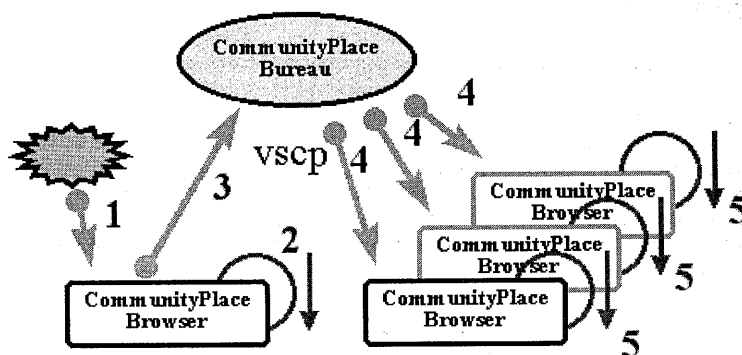


Figure 5.1 Simple Shared Script model

The drawback of the SSS model are based on ownership and persistence. Since all scripts are equal, they need to

communicate among themselves to resolve ownership and persistence. To solve these problems, we introduced a mechanism to assign master ownership to one browser among the browsers connecting to the same shared world. It can receive messages from other browsers and send messages to them. We tend to use the SSS for simple shared applications that do not need sophisticated synchronization or persistency requirements.

5.2 Application object

The application objects are the entities which are responsible for handling application specific behavior of 3D objects in the 3D world. It allows application creators to create 3D objects and inject them in the existing shared scene dynamically. It also supports to delete the objects dynamically. The AO sits out of the CommunityPlace Bureau and communicate with it using Virtual Society Application Protocol(VSAP) protocol to register the AO and send requests and receive a message to/from the Bureau.

Figure 5.2 shows how AO model works. When AO is attached to the Bureau, it requests the Bureau to create and inject 3D objects (1). The Bureau sends the request to all browser and then the browser do that (2).

When a user selects a 3D object(3), a local script associated with the object is executed (4). The script sends a message to the CommunityPlace Bureau (5). The Bureau sends the message to the AO managing the selected object (6). The AO performs internal processing and then typically sends back a message (7) via the Bureau to all other browser (8).

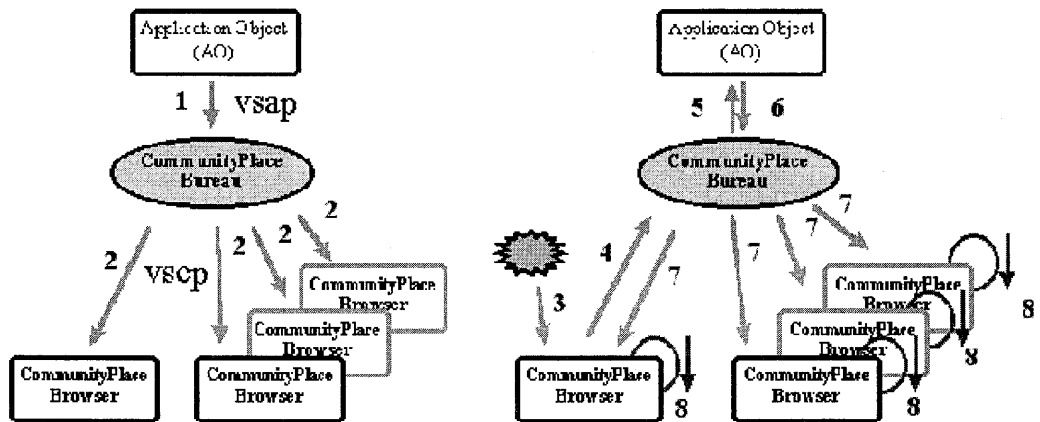


Figure 5.2 Application object model

6. Performance

We conducted five experiments to show the basic performance of the CP system. In this experiments, the Bureau is running on a NEWS 5000 and the browsers are DEC Celebris PCs (Pentium 133MHz 64MB - Windows 95).

The first experiment is performed to give readers the general understanding of the browser event processing costs. In the first experiment we measured the cost of calling Java from VRML (the cost of eventIn) and posting event from Java to VRML (the cost of eventOut). The actual experiment is done in the following way: we have two script nodes in a VRML file, each of which refers to a different Java class file. On some initial trigger event, one of the script nodes starts to generate events in the Java code, which is routed to the other script node. Execution path looks like: Java -> eventOut -> routing -> eventIn -> Java. The destination Java script code performs nothing. The cost of this execution path is 0.99ms. (average of 3000 times).

The second experiment is performed to give readers the cost of basic networking functions. In the second experiment we measured the cost of sending a message from Java program in a browser to a server which sends back the message to the same client. The actual setting is as follows: we connects to a server a browser which loads a simple VRML world. On the initial trigger event in the world, a Java code is called (measurement started) and sends a message (using our own API) to the server. The server automatically sends back the message to the browser. On receiving the message, browser generates an event and routed to the same Java code. The execution path looks like: Java -> messaging API -> network -> server -> network -> eventIn -> Java. The destination Java script code performs the

same message sending to repeat the execution path for specified number of times. The cost of this execution path is 6.48ms. (average of 3000 times).

The other three experiments show the different costs for different application models in a practical settings. As described in section 6, we have two different application programming models, Simple Shared Scripts and Application Object. We created very simple shared behavior programs, one in SSS and the other in AO. In either programs, an initial event is generated on one client browser and delivered to a master (in case of SSS model) or AO (in case of AO model). On receiving the message, the master or AO sends the same message to all the clients (including the original event generator browser). The execution path looks like: Java -> messaging API -> network -> server -> network -> AO or master -> network -> server -> network -> eventIn -> Java. The destination Java code sends the same message to repeat the execution path for the specified number of times. In addition, in order to realize the practical user situations, we connect additional clients to the server and see how it affects the cost of the execution path. The cost of the execution path is shown in the graph (Figure 6.1).

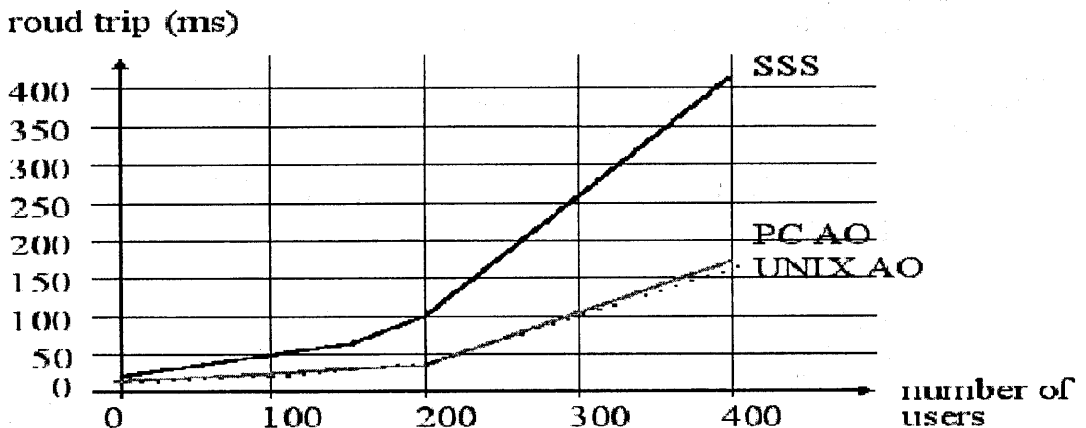


Figure 6.1 Total system communication costs

The graph contains three lines: one for SSS and the other two for PC AO and UNIX AO. Currently we develop AOs on a PC platform a UNIX platform. For comparison, we used both PC AO and UNIX AO for the measurement. The graph shows clearly that the SSS model is both more costly and more sensitive to server load. The reason for this is that the SSS model does not use the server internal Aura mechanism.

In contrast, the AO model uses the Aura mechanism in the Bureau. Thus, the maximum number of messages generated from the AO is limited by the number of users who can enter the Aura. In these experiments, it is set at 8.

7. Current status

The CommunityPlace system has been in public beta release since December 1995. The downloadable browser, conductor and server are available. The system is fully functional and supports distributed shared world, multi-user chat feature (both text and audio) and shared behaviors. It is based on the WWW, VRML2.0 and Java standard. The product version has been available since September 1996.

We have tested larger servers at two public sites, one is in Japan and one in the USA since February 1996. Both sites are freely accessible from anywhere on the Internet to connect to and share a world. We have made available a set of sample shared worlds. Figure 7.1 shows one of the sample shared worlds called "Circus Park" and the shared behavior. In the figure, we can see two browsers showing the same scene from two different viewpoints. In the center is a sea lion whose behavior is to flip the ball up on his nose and put it down again. This behavior is activated by user selection and shared. So the all browsers within the sea lion's aura can see that the behavior is happening simultaneously.

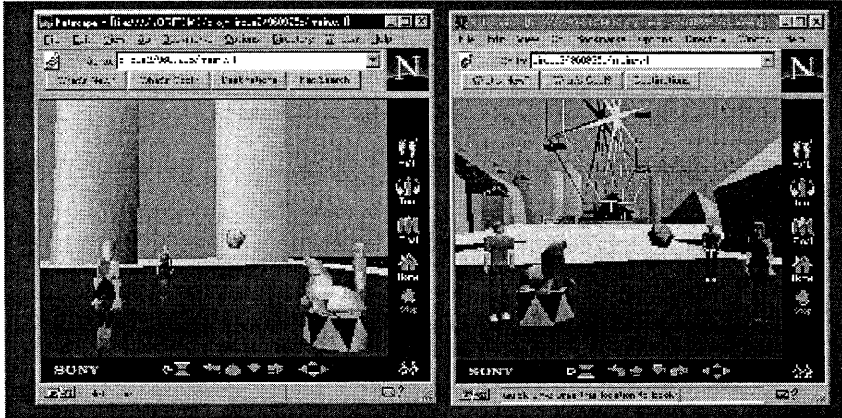


Figure 7.1 Shared behavior

Our initial experiences are encouraging, we have fostered a small but active community of regular visitors at the public site in Japan. Users have had no difficulty driving the system and making use of the interaction mechanisms we have provided.

We have limited experience of collaboration within our 3D framework having built only a simple shared whiteboard. However, we are now developing more sophisticated shared applications and hope to deploy them within several months.

8. Related work

Recent work targeting the WWW and using full 3D shared spaces has mainly been confined to the VRML community. Within that community there are several projects of note.

The **Cybergate** system from Blacksun Inc.[7] has built and experimented with shared 3D spaces similar to the CP project. However, CyberGate is based on CosmoPlayer and Live3D and so supports only static scenes. **Moondo**[8] from Intel is a similar system to CyberGate in that it currently supports only static scenes. However, Moondo has experimented with a shared object model as a basis for shared consistent objects. In addition, Moondo has added support for audio chat.

The **Pueblo** project from Chaco Communications [9] has evolved out of earlier work on social MUDs (Multi-User Dungeon). Recently it has augmented the MUD server with VRML support and provided a VRML1.0 browser that allows MUD authors to build 3D scenes. This approach allows world builders access to the rich mechanism of the MUD database, but again only supports static scenes.

An interesting experiment in 3D spaces that supports audio chat is the **Traveler** system from OnLive![10] Again, Traveler is VRML1.0 based, and so static, but the primary focus of the group has been on audio interaction. To support this, they have built a low-bandwidth audio codec suitable for the Internet. In addition, they have augmented user avatars with facial motion, and in particular lip synch, that is driven from the audio stream. This technique offers a computationally cheap, but rich and compelling interaction mechanism.

9 Conclusion and future plans

Our current system is sufficient for medium scaled shared social spaces. We are pursuing several developments to allow our platform to evolve into a general purpose shared 3D platform supporting the Virtual Society concept.

Our first area of concern is scaling. Our current server supports up to 700 users. We have a goal to support several thousand users for mass events such as shared concerts etc. To support this we are currently developing a distributed multi-user server. The distributed server uses multicast communication to allow replication of server data.

A second area of development is tools and applications to enrich the shared worlds. Our first development is support

for audio chat in addition to textual chat. Subsequently we will add support for streaming audio and video into shared worlds. In parallel we are developing a general purpose application environment with a rich set of application API's to allow anybody to author shared 3D scenes.

Lastly we are exploring techniques to foster on-line communities, these include:

- Supporting a social infrastructure
- Supporting social rules
- User interface to interact with other people
- Concept of ownership in the shared virtual world

Our goal is to provide a rich 3D collaborative experience through a combination of base level technology and higher level tools. This will encourage third parties to use and develop new technology for our platform.

The Community Place system is freely downloadable from two web sites:

<http://vs.sony.co.jp>

<http://www.spiw.com/vs>

10 Acknowledgments

We are indebted to our colleagues in the Sony Computer Science Lab. and Sony's architecture Lab. for their help in the definition of this project. We also wish to thank our colleagues at the Swedish Institute of Computer Science who have contributed indirectly to the CP design as part of our joint research project, Wide area virtual environments (WAVE). Lastly, our thanks, as always, go to Akikazu Takeuchi, Mario Tokoro and Toshi Doi for their continuing support.

References

- 1 The VRML2.0 specification. Currently available as <http://vag.vrml.org/VRML2.0/FINAL>
- 2 Honda, Y., Matsuda, K, Rekimoto, J and Lea, R. Virtual society. Procs. of VRML'95, San Diego. USA. Dec. ACM press 1995 pp. 109-116, Order no. 434953 Available at:
<http://www.csl.sony.co.jp/project/VS/VRML95.ps.Z>
- 3 Lea, R., Raverdy, P.G, Honda, Y, and Matsuda, K. Issues in the design of a large scale VE. Sony Computer Science Lab Tech Report XX.XX.95 available from <http://www.csl.sony.co.jp/>
- 4 Lea, R., Raverdy, P.G, Honda, Y, and Matsuda, K. (Virtual Society: Collaboration in 3D spaces on the Internet. available from <http://www.csl.sony.co.jp/>
- 5 Carlsson, C. and Hagsand, O. DIVE - A platform for multi user virtual environments. Computer and Graphics Vol. 17. No. 6 1993 pp. 663-669
- 6 Benford.S, and Fahlen, L. A spatial model of interaction in large virtual environments. September 1993, In proceedings of G. DeMichelis et al (Eds.) Third European Conference on Computer Supported Cooperative Work (pp. 109-124), Kluwer Academic Publishers.
- 7 Blacksun Inc. home page. Currently available as <http://www.blacksun.com/>
- 8 Intel Moondo home page. Currently available as <http://www.intel.com/iaweb/moondo/index.htm>
- 9 Chaco Communications home page. Currently available as <http://www.chaco.com/DD>>
- 10 OnLive! home page. Currently available as <http://www.onlive.com/index.html>
- 11 Binary format proposal. Currently available as <http://www.austin.ibm.com/vrml/binary>
- 12 SGI's external authoring interface proposal. Currently available as <http://vrml.sgi.com/moving-worlds/spec/ExternalInterface.html>
- 13 Dimension X's external interface proposal. Currently available as <http://www.dimensionx.com/people/wing/extapi.html>
- 14 "Living Worlds" proposal. Currently available as http://www.livingworlds.com/draft_1/index.htm