

Space-sharing architecture for a three-dimensional virtual community

This article has been downloaded from IOPscience. Please scroll down to see the full text article.

1998 Distrib. Syst. Engng. 5 101

(<http://iopscience.iop.org/0967-1846/5/3/003>)

View [the table of contents for this issue](#), or go to the [journal homepage](#) for more

Download details:

IP Address: 38.107.179.210

The article was downloaded on 20/02/2012 at 07:44

Please note that [terms and conditions apply](#).

Space-sharing architecture for a three-dimensional virtual community

Hiroaki Harada[†], Naohisa Kawaguchi[‡], Akinori Iwakawa[§],
Kazuki Matsui^{||} and Takashi Ohno[¶]

Personal Systems Laboratories, Fujitsu Laboratories Ltd, 64 Nishiwaki,
Ohokubo-Cho, Akashi 674-8555, Japan

Received 17 March 1998

Abstract. This paper describes our proposal for a new virtual community architecture that employs a technique for dividing and managing virtual space, a message-model, and a space-sharing technique. We also describe certain concepts that can be used to apply our architecture to PCs and modems: the pack-delivery method for messages, and an asynchronized-rendering method. Finally, we review the results of our experimental service: 'CyberCity96'.

1. Introduction

In this paper, we describe an architecture designed for virtual communities [1–3] in three-dimensional (3D) spaces. Our goal was to develop a flexible and generalized communication system by combining two characteristics: the variety of sites in the world-wide web (WWW) and the ability of near real-time communication, such as that exemplified in chat rooms.

Distributed virtual environment (DVE) is a typical technology employed for realizing virtual communities. DVE has to address more serious problems, such as the control of network traffic, latency, and reliability, than simple virtual reality systems do [4–7].

The control of network traffic does not depend on the load on the user's terminal or on the central server system, but on the amount of messages. A trade-off relationship exists between improving the latency and preserving the reliability of networks. Recently, various prototypes have been developed to solve this problem by using such techniques as IP-multicast or point-to-point communications [8, 9].

These problems must be solved by a combination of techniques, and the solution must consider the real environment of the network or utility purpose. We set the following technical goals in our approach to this problem.

—The ability for the general public to communicate freely in DVE systems.

—To provide DVE system support via PCs, modems, and telephone-lines in the home.

—To achieve a balance between the functions used for creating communities and dispersing the network load.

In relation to this, we considered three technical points.

—Simultaneity means that changes of status of each client-terminal are synchronized.

—Scalability means that the network load does not increase with the number of users.

—Consistency indicates the same virtual world exists in each client-terminal.

In this paper, we propose a new architecture that includes a technique for dividing and managing virtual space, a message-model, and a space-sharing technique. We also explain certain concepts that can be used to apply our architecture to PCs and modems: the pack-delivery method for messages, and an asynchronized-rendering method. Finally, we describe the result of our experimental service 'CyberCity96'.

We refer to our proposed architecture as 'AGORA', which means 'market' or 'public place' in ancient Greek.

2. Architecture

2.1. Construction model

AGORA was constructed as a server–client model, which provides the benefits of security and collective management. The server that manages communities, a client that can browse communication channels, and the communication protocol between the server and the client, are called the 'community server' (CS), the 'community browser' (CB), and the 'community protocol' (COMMP), respectively. The database for the community spaces resides not only on the CS and WWW, but with each client, if necessary. However, only the server can maintain the consistency of the data. In other words, AGORA is a DVE system that is classified as a shared, centralized database system. Figure 1 shows the display of a user terminal.

[†] E-mail address: harada@flab.fujitsu.co.jp

[‡] E-mail address: nkawa@flab.fujitsu.co.jp

[§] E-mail address: iwakawa@flab.fujitsu.co.jp

^{||} E-mail address: kmatsui@flab.fujitsu.co.jp

[¶] E-mail address: ohno@flab.fujitsu.co.jp

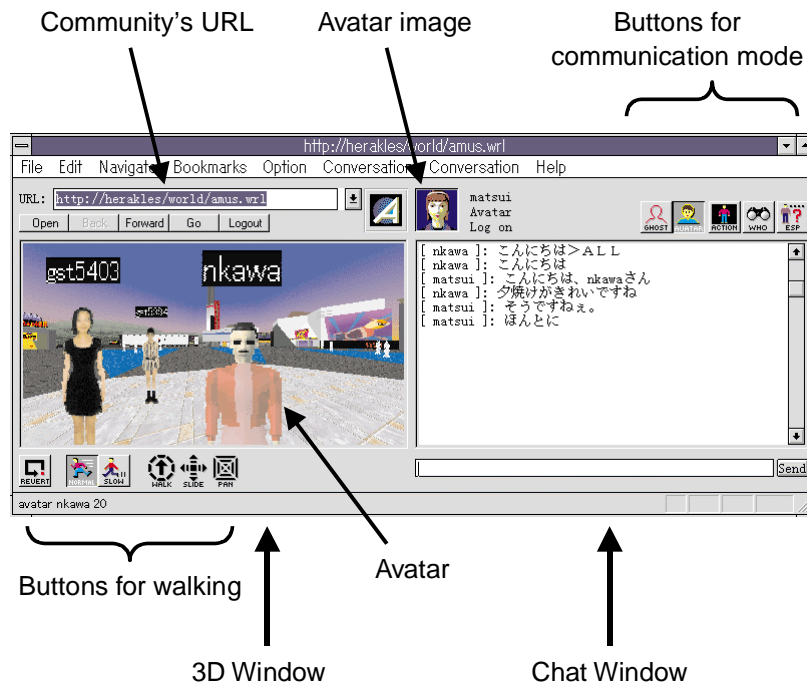


Figure 1. Example of CB screen.

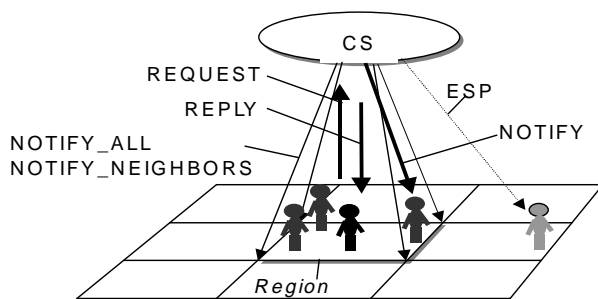


Figure 2. Illustration of region and message model.

Table 1. Message types.

REQUEST	Request from CB to CS CS replies to CB
REPLY	Reply from CS to CB
NOTIFY	Notifies a special CB
NOTIFY_ALL	Notifies all CBs in a region
NOTIFY_NEIGHBOURS	Notifies all CBs in a region (CS does not notify the requested CB.)

2.2. Message model

Figure 2 depicts a message model between a CS and a CB. The types of messages are shown in table 1.

A significant characteristic of AGORA's message model is that the exchange of messages is enclosed within one small space called a 'region'. In public communities, there is the danger of a CB sending illegal messages to the

CS. Therefore, the CS does not always swallow messages from CBs and data in CBs. For example, the CS does not query a CB on its information. A CB sends a REQUEST or NOTIFY message to the CS when hoping to change some attributed values in the database. After the CS receives the request, it changes the attributed values and notifies all CBs of the change. If a CB has changes for a client's attributes, an illegal change cannot spread to the other clients. The CS manages all information regarding users and therefore confirms all queries from all CBs.

3. Space-sharing techniques

In a virtual community, it must be possible for everybody to obtain any object as a subject for communication, and for everybody to be able to manipulate the object and see it in real time. In other words, objects and time both need to share virtual space. This concept of consistency and synchronization between clients is called 'space sharing'. Space sharing denotes the notification of a phenomenon in one virtual space to all clients immediately.

This technology is known as DVE, and there have been several studies carried out in this area. However, most of the proposed technology, such as IP-multicasting, is inconvenient and has yet to be implemented in practical, wide-spread network services.

The goal of AGORA is to provide a communications technology for communities by sharing space on home-computing equipment (PCs, modems, and telephone lines). This section describes two techniques for solving the trade-off between convenience for the community and the network load.

—A technique that divides communication space into fixed sizes.

—Using an ‘interactive VRML server’ to maintain the communication space.

3.1. Region

‘Aura’ or distance filtering, is a famous technique for managing virtual space [10]. Employing this idea, Barrus *et al* have proposed the use of combinations of small areas called ‘locales’ [11].

This idea is effective enough to prevent network traffic from rapidly increasing in a wide virtual space. Conversely, the combining of auras in the aura technique may add overhead processes, and the size of auras in crowded spaces may not be determined properly. It requires a strict control of communications and traffic forecasts for point-to-point communications between several clients.

We presume that the size of the area does not need to be so wide in a virtual community, because the chance of meeting and the number of participants are both small. Thus, a community does not need a wide area controlled by an aura. AGORA applies the ‘region’ technique, which is a small static area of a fixed size instead of the dynamic aura technique. A region corresponds to a uniform resource locator (URL) and is the smallest unit available for sharing space and communications. A community space is divided into several regions.

A direct effect of the region concept is that it can decrease the traffic in the community. When there are N people in a space speaking together simultaneously, the order of messages is $O(N^2)$. If a space divided into M regions and persons can only speak within a region, the order of messages becomes $O(N/M * N) = O(N^2/M)$. It can reduce the network load by a suitable size of M . In addition, the region does not require a special control technique and does not create any overhead. This simple region technique can be applied to home-computing environments.

3.2. Interactive VRML server

All users must be able to bring their own objects into the virtual community. In addition, clients share the same world after connecting to the community. Therefore, the community needs a function to register user’s objects and a degree of consistency in the shared space.

Lea *et al* proposed two concepts for space sharing: ‘simple shared script’ (SSS), a model, in which a selected client manages the entire space, and ‘application object’ (AO) model, in which a selected client (AO) registers new objects in the space [4]. However, these concepts are insufficient in a public community, because the consistency of the space cannot be assured and because public clients cannot register their own objects in the public community.

AGORA includes an ‘object-managing server’, which manages all the status pertaining to the virtual space, as shown in figure 3.

(1) A CB which has changed status in the space sends a message generalizing the event to the CS.

(2) The CS send this message to the object-managing server.

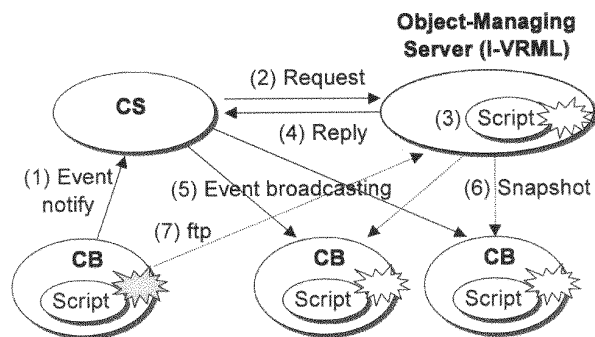


Figure 3. Space sharing by the object-managing server.

(3) The object-managing server initiates a process corresponding to the message that changes the status of the space.

(4) The CS is then notified of the result of this process.

(5) The CS broadcasts the message to all clients.

(6) The object-managing server presents a snapshot of the status to all clients.

Figure 4 illustrates the details of the ‘interactive VRML server’ (I-VRML server) which is a type of object-managing server. The I-VRML server manages shared space in the database in the form of VRML via external controls. When a new CB in the community requests the http controller of the I-VRML server to get a snapshot of the community, the I-VRML server translates the objects in the database into VRML form (the requested CB can ‘GET’ the snapshot through http protocol).

When a CB wants to present its own objects to the community, the CB can ‘PUT’ the objects into the I-VRML server directly through ftp protocol, after the CB has been certificated by the CS. The result of this process is broadcasted by the CS to all the clients in the community.

On the other hand, a self-directed script in the I-VRML server can update the status of the virtual world in the same manner, and send messages to all clients through the CS.

The following are some of the advantages of I-VRML server architecture.

—A newcomer CB can synchronize with the community via the snapshot.

—Illegal registrations from a CB can be checked by the CS.

—It only takes a short time to synchronize all of the CBs and the CS, because the CS can immediately broadcast a message from the I-VRML server if the same script is being processed on the CBs and the I-VRML server.

—Even a general VRML browser that is not a CB client can browse the snapshot of the community, because the I-VRML server can be accessed by http protocol.

In the following, two aspects of the I-VRML server are discussed.

—Unrestricted registration of objects from each CB to the shared space as shown in figure 3(7), leads to possible consistency conflicts because different objects registered from some CBs might occupy the same position simultaneously. To avoid such cases, a management rule for the shared space is necessary. We applied the simple

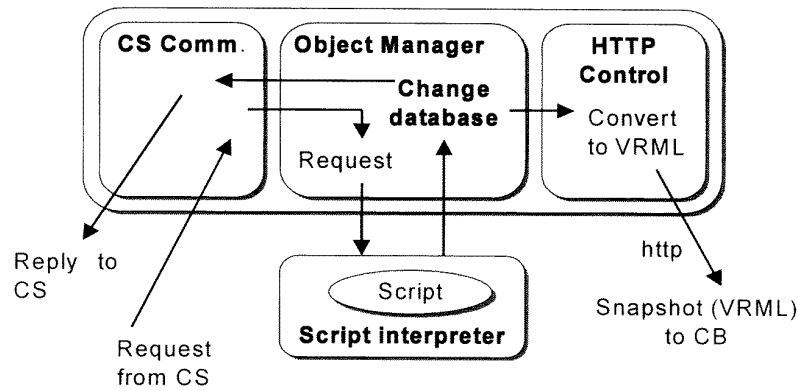


Figure 4. Structure of the I-VRML server.

Table 2. Effect of the pack-delivery method.

Case	Regions	Clients per region	Avatars in all the regions	Load average
A	1	100	10 (and 90 ghosts)	0.8 ~ 0.4
B	10	10	100	1.0 ~ 0.4
C	100	1	100	0.5 ~ 0.3

rule ‘first come, first served’. The I-VRML server checks the respective position in the shared space whenever a CB requests to register an object, and permits registration only when there is no object at the respective position. AGORA does not address complex conflicts between the objects, such as a case in which a CB requests to remove an object A (registered by this CB) under an object B that was registered by another CB.

—The http protocol for getting snapshots is not always advantageous for performance. On the other hand, it is difficult to receive large quantities of VRML data smoothly in any protocol. We selected the http protocol for snapshots because of its ease of use and popularity.

4. Performance

4.1. Pack-delivery technique

‘Pack delivery’ is a technique of message passing that uses a dead-reckoning method [12] and delivery by the CS to resolve latency problems. This technique can decrease the network load and synchronize all the CBs.

Messages from the CBs passing through the CS are called ‘notice vectors’. They indicate the coordinate position of an avatar, or the changes in an object’s attributes. A notice vector from CB i at time t describes $V_i(t)$ (shown in figure 5).

The CS sends $O(N)$ messages to all CBs via one message from a CB (in AGORA, a ‘multicast’ network for broadcasting is not used). The CS has to manage $O(N^2)$ (where N is the number of CBs in a region) messages to accept and broadcast continuously in all. To solve this problem, the broadcast interval times are fixed.

After each CB gathers its notice vectors $\{V_i(1), V_i(2), \dots, V_i(t_B)\}$ for $t_B(s)$ in a package and sends it

to the CS, the CS gathers and re-packs several packages from CBs for $t_S(s)$ in a new package $\{V_1(1), V_1(2), \dots, V_1(t_B), \dots, V_N(1), \dots, V_N(t_B)\}$ and broadcasts it to all the CBs.

In this manner, the CS broadcasts $O(N)$ messages for each $t_S(s)$.

However, a delay (near 0 s in the best case, $(t_B + t_S)(s)$ in the worst case, and $(t_B + t_S)/2(s)$ in the mean) occurs before the package reaches a CB (not including the delay transmission through the network). When the notice vectors include the positions of avatars, the rendering of each avatar can be compensated by the dead-reckoning method and the linear interpolation in each CB. The avatar can walk smoothly by the interpolation of $\{V_i(1), V_i(2), \dots, V_i(t_B)\}$. In practice, the interval times t_B, t_S , are set to 1 s.

An exception to this is when an event, which is important despite its rare occurrence, is sent immediately to the CS as a sole message.

Table 2 shows the effect of the pack-delivery method. In this experiment, the CBs and the CS process were run on the same workstation (Sparc-20/Model 612, 64 Mb, Solaris 2.4). Three experiments were conducted for 1 h each in which the number of regions, the number of CBs, and the number of avatars were changed. In these experiments, avatars were continuously walking, had chatted every 15 s, and entered and exited the scene every 5 min. These conditions mean that notice vectors of the position of an avatar are constantly sent to the CS, and short text messages are included every 15 s.

The following is an explanation of table 2. The region used in this explanation refers to a small area which is part of shared space. In experiment A, the shared space consists of only one region. In experiment C, shared space is divided into 100 regions. The number of clients, as used in this explanation, means the number of CBs that access

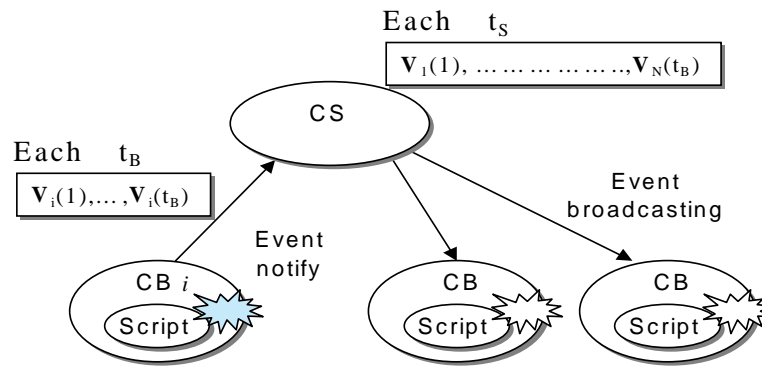


Figure 5. Pack-delivery method.

the region. In total, 100 clients access the shared space. The number of all avatars, as used in this explanation, refers to the number of all clients. However, as there can be only 10 avatars in one region, the other 90 clients access as ‘ghosts’ in experiment A. Ghost mode is a mode in which the CB sends no messages to the CS. The ghost is not visible, but can move everywhere and can eavesdrop.

Table 2 shows that the load average in experiment B is not so high and it is comparable to A or C, despite having the highest number of messages in these experiments.

4.2. Dual message queues

Each CB client must accept numerous messages from the CS each t_S (s). Therefore, the response of input devices such as a keyboard or mouse may slow down if they place too much demand on the CB terminal’s CPU.

In AGORA, the messages from the CS are separated into two queues: a sequential-type queue and a non-sequential-type queue (shown in figure 6).

The messages in the sequential-type queue must be processed in the order of arrival and involve the rendering of menus or managing of the database. On the other hand, the messages in the non-sequential-type queue may be processed in any order, or may be ignored when the queue overflows. An example of this type is the positions of avatars. The highest priority for processing is assigned to sequential-type messages and input/output control. Non-sequential-type messages have a lower priority.

4.3. Asynchronous rendering

The rendering of 3D computer graphics requires a large portion of the CB’s CPU power. The CB has to render VRML data in the correct order to the notice vectors from the CS in real time.

VRML, which is the standard form for CG, describes 3D objects and their attributes via a tree construction named a ‘scene graph’. The rendering is processed according to the scene graph. Because the rendering may exceed that of updating the scene graph, instances may occur in which rendering cannot keep up with the scene graph updates.

To solve this problem, the rendering of 3D scenes will be suspended when (a) the user is operating the keyboard or

Table 3. Experimental system.

Client (CB)	CPU: Pentium 90 MHz or more Memory: 32 Mb or more OS: Windows 95 Program size: 814 kb
(WWW browser)	Netscape Navigator or Internet Explorer
Server (CS)	CPU: Sparc-20 Model 712 Memory: 64 Mb OS: Solaris 2.4 Capacity: 300 avatars
3D space	Region: 27 Total amount of data: 23 Mb
Communication	LAN (TCP/IP) or Telephone-line (PPP, Modem: 14.4 kbps)

mouse, or (b) the user is changing their perspective while operating their own avatar. In these cases, the person using the CB may not need rigorous rendering.

In the first example, a user must be conferring with another user using a keyboard or mouse, reading some html documents, or executing some other application program. Precise rendering is not required in this case as the user is being subtracted from a 3D scene. The second example is a case when the user is watching a 3D scene controlling their own avatar. When this avatar is moving and its eyes are wandering around the scene, the 3D scene changes so widely and quickly that the user would not notice changes of small details anyway. While the simulated 3D scene may not be moving smoothly, if the CPU power of the client PC is not sufficient to render the whole scene precisely, priority was given to maintaining consistency.

5. Experimental result

We made a prototype CB, CS and I-VRML server, and have established an experimental network service called ‘CyberCity96’, which operated from July to December 1996. Table 3 shows the specifications of the prototype system. The CS and WWW server were run on the same

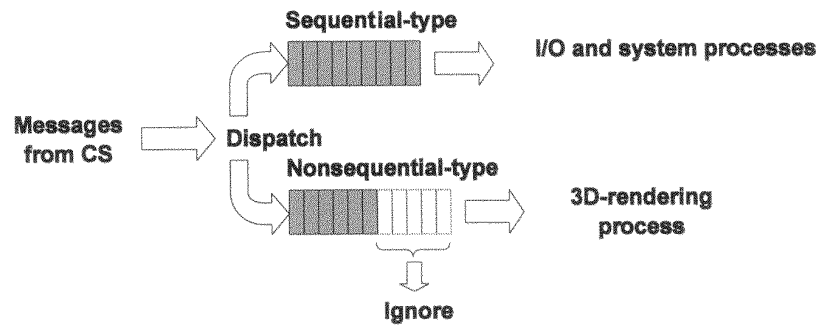


Figure 6. Dual message queues.

Table 4. Result of experimental service.

Total number of access	11 949 counts
Average access time	39.7 minutes
Maximum number of users per region	15 people
Number of guests	6655 people
Number of inhabitants	556 people

machine. The user capacity in all regions was set to 300 people.

In this experimental service, all of the 3D data described in the VRML had been prepared beforehand, and the CBs were not permitted to generate, dissolve, or register any 3D regions or objects. The results of this experimental service are shown in table 4.

All of the virtual spaces were opened to the public via the Internet, but users did not have to enter them. The actions of the users included the following.

(1) A type of pick-up program or personality program had been accessed, (2) some volunteers such as the guide and the publisher of a virtual newspaper appeared, (3) some inventors of new greetings and non-verbal actions appeared.

6. Conclusion

In this paper, we proposed an architecture that can share objects and space and can synchronize CBs in a virtual community. The advantages of this architecture include the following.

—The construction of regions is simple, making implementation easy and efficient.

—The I-VRML server can synchronize its speed to that of the event passing, and can open a snapshot to new CBs and other types of browsers.

—The pack-delivery method can prevent an increase of traffic by using a message delay. This delay can be compensated for by the dead-reckoning method.

The AGORA architecture demonstrated its practicality during this experimental service.

In the future, we will study security and distribution technologies for wide virtual communities. In addition, it will be necessary to improve the standardization of 'LivingWorlds' [13] as the standard for avatar control and

of 'UniversalAvatars' [14] for defining each avatar.

Acknowledgments

We are grateful to Kazutomo Fukuda, Makoto Urano, and Keiko Tateishi for their efforts concerning the CyberCity96 service. We also thank Yoshichika Kakiuchi, Makoto Wanishi and Hiroyuki Watanabe for designing the 3D spaces used in CyberCity96. We also extend our thanks to all those who visited CyberCity96.

References

- [1] Morningstar C and Farmer F R 1991 *The Lessons of Lucasfilm's Habitat in CYBERSPACE: First Steps* ed M Benedikt (Cambridge, MA: MIT Press)
- [2] Loeffler C E 1994 *Virtual polis Int. Symp. on Next Generation Multimedia Telecommunication Service (Tokyo)*
- [3] Harada H *et al* 1995 AGORA: a 3D communication space on world-wide web *2nd IEEE Workshop on Networked Realities (Boston)*
- [4] Lea R *et al* 1997 Community place: architecture and performance *Proc. VRML97 (Montley)* pp 41–50
- [5] Macedonia M R and Zyda M J 1997 A taxonomy for networked virtual environments *IEEE Multimedia* **4** 48–56
- [6] Braham R *et al* 1997 Sharing virtual worlds *IEEE SPECTRUM Special Report* 18–50
- [7] Pope A 1989 The SIMNET network and protocols *BBN report no 7102* (Cambridge: BBN Systems and Technologies)
- [8] Hagsand O 1996 DIVE-A platform for multi-user virtual environments *IEEE Multimedia* **3** 30–9
- [9] Macedonia M R *et al* 1994 NPSNET: A network software architecture for large scale virtual environments *Presence* **3** 265–87
- [10] Lea R *et al* 1997 Issues in the design of a scalable shared virtual environment for the internet *Proc. HICSS'97 (30th Ann. Hawaii Int. Conf. on System Sciences) SOF22 (Hawaii)*
- [11] Barrus J W *et al* 1996 Locales and beacons: efficient and precise support for large multi-user virtual environments *Proc. VRAIS '96* (Los Alamitos, CA: IEEE Computer Society Press) pp 204–213
- [12] Miller D 1989 Long-haul networking of simulators *Proc. 10th Interservice/Industry Training Systems Conf. (Arlington)* (ADPA) p 2
- [13] <http://www.vrml.org/WorkingGroups/living-worlds>
- [14] <http://www.chaco.com/avatar/avatar.html>