MUDs Grow Up: Social Virtual Reality in the Real World

Pavel Curtis and David A. Nichols

Xerox PARC, 3333 Coyote Hill Rd., Palo Alto, CA 94304

Abstract

MUDs are multi-user, text-based, networked computing environments that are currently used mostly for gaming. Despite the obvious shortcomings of a text-only system, they are quite popular. The Social Virtual Reality project at Xerox PARC is extending MUD technology for use in non-recreational settings. Our goal is to keep the strength of MUDs—shared computing with a powerful real-world metaphor—while correcting their shortcomings by adding audio, video, and interactive windows. We are building two specific prototypes: Astro-VR, for use by the professional astronomy community, and Jupiter, for use by researchers within Xerox.

1: Introduction

MUDs, or "Multi-User Dungeons," are programs that accept network connections from multiple simultaneous users and provide access to a shared database of "rooms", "exits", and other objects. Users browse and manipulate the database from "inside" the rooms, seeing only those objects that are in the same room and moving between rooms mostly via the exits that connect them. MUDs are thus a kind of *virtual reality*, an electronically-represented "place" that users can visit.

MUDs are unlike the kind of virtual realities that one usually hears about in three important ways:

- MUDs do not employ fancy graphics or special position-sensing hardware to immerse the user in a sensually vivid virtual environment; rather, they rely entirely on plain, unformatted text to communicate with the users. For this reason, MUDs are frequently referred to as *text-based virtual realities*. [6]
- MUDs are extensible from within; MUD users can add new rooms and other objects to the database and give those objects unique virtual behavior, using an embedded programming language.
- MUDs generally have many users connected at the same time. All of those users are browsing and manipulating the same database and can encounter both the other users and their newly-created objects. MUD users can also communicate with each other directly, in real time, usually by typing messages that are seen by all other users in the same room.

```
>look
Corridor
The corridor from the west continues to the east
here, but the way is blocked by a purple-velvet rope
stretched across the hall.  There are doorways
leading to the north and south.
You see a sign hanging from the middle of the rope
here.
Gumby is here.
>:waves.
Munchkin waves.
>read sign
This point marks the end of the currently-occupied
portion of the house.  Guests proceed beyond this
point at their own risk.
        -- The residents
Gumby says, "What're you up to?"
>"Just exploring this place. Bye!
You say, "Just exploring this place. Bye!"
Gumby waves bye-bye.
>go east
You step disdainfully over the velvet rope and enter
the dusty darkness of the unused portion of the
house.
```

Figure 1: A typical MUD encounter

A sample transcript of a user's MUD session appears in Figure 1; the user's input is underlined.

MUDs have existed for about fifteen years, beginning with a multi-user "adventuring" program written in 1979 by students at the University of Essex, England [2], and becoming particularly prominent on the global Internet over the past five years or so. Throughout that time, they have been used almost exclusively for recreational purposes.

About two-thirds of the MUDs in existence today are specialized for playing a game rather like "Dungeons and Dragons," in which the players are assigned numerical measures of various physical and mental characteristics and then have fantasy adventures in a role-playing style. Nearly all other MUDs are used for leisure-time social activity, with the participants spending their connected periods talking with each other and building new areas or objects for general enjoyment. A recent list of Internet-accessible MUDs showed over 300 advertised, running at sites all over the world. The busiest of these frequently host 150 to 200 simultaneous users.

Clearly, these recreational MUDs are very popular systems. There are a number of reasons for this, we believe, but two that stand out for our purposes are the social quality of these systems (one uses them in the company of other people) and the richness of the metaphor they employ (much of what's possible to do is intuitively obvious, appealing to the user's lifetime of real-world experience).

We generalize from MUDs along these lines by referring to the notion of *social virtual realities*, which we define as "software systems that allow multiple people to interact and communicate in pseudo-spatial surroundings." It seems clear to us that social virtual realities, especially as embodied in the simple technology of MUDs, should be immediately useful in other, non-recreational applications.

This paper presents our plans for exploring the implementation, applications, and implications of social virtual realities in work-oriented contexts. In the next section, we describe the ways in which we have enhanced our own MUD server. We then describe the two major systems we are building as foci for our research.

2: Enhancing the usual MUD model

Current MUDs use plain, unformatted text as the sole communications medium, both between users and between the server and user. Text has the advantage of being universally available, allowing a wide range of users to participate. For recreational use, text has additional advantages similar to those of radio over television, as words can often paint a more compelling mental image than can, for example, a picture produced with MacDraw.

Unfortunately, text has significant drawbacks in a work environment. First, typing is much slower than speech for real-time communications. While most people can read faster than they can listen, they can speak much faster than they can type. In a world where telephones and face-to-face communications are readily available, real-time typing will not be a popular alternative.

Secondly, the aesthetic properties of text descriptions are largely wasted in a work setting. A worker's use of the computer is generally secondary to accomplishing some other task, and dramatic descriptions of the successful printing of a document, after victory over the evil forces of paper jams and toner spills, get old after a while.

In addition to text-based interaction, therefore, we are adding real-time audio and video, and window-based graphical user interfaces. While typical MUDs assume that users have only a dumb terminal, we assume that they have digital audio and a graphics display, with optional video input. This is an increasingly realistic assumption, as almost all computers sold today (PCs, Macs, Unix workstations) have graphics displays, and most either come with telephone-quality audio or can have it added cheaply (usually for about $100).

Just as workstations are improving, so is network technology. In particular, "multicast" routing capability [7] is becoming available, in which individual packets can be received by many hosts without duplicating the packets on any link and without the sender having to know who any of the recipients are; we use this to handle real-time audio and video traffic efficiently.

2.1: The LambdaMOO server and the client

As a base for our work, we are using a MUD server developed here at PARC, called "LambdaMOO." The facilities it provides are generally typical of those available on other MUD servers, though in a form that we find particularly convenient. [4]

The server has only three jobs: accepting and managing network connections from the users, maintaining the database of rooms, objects, and users in a robust and persistent way, and interpreting programs written in the server's embedded programming language, called "MOO." Users invoke MOO programs each time they enter a command.

MOO code has access to all of information about the objects stored in the database (including the code for all of the stored MOO programs) and produces its effects either by changing those objects or by reading or writing simple text strings from or to the various users' network connections. To make the programming model as simple as possible for naive users, each user command's associated program is run to completion before any other command is begun; thus, no issues of concurrency or locking arise.

As a consequence of this design, MOO programs do not run so quickly as to allow them to be used to handle any high-frequency events, such as character-by-character typing, mouse motions, or real-time network processing.

```
(VBox
  (HBox
    (Text LeftAlign (Name messageNum))
    (Fill)
    (Button (Name reply) “Reply”)
    (Button (Name delete) “Delete”))
  (Bar)
  (TextEdit (Name messageBody)))
```

Figure 2: Example window description

On the other hand, the interpreter is efficient enough for use in most human-speed command-response interfaces.

A purely text-based MUD can be reached by users running any normal Internet telnet program. All the characters typed by the user are sent to the server, and all the server’s output is displayed on the user’s terminal. The extended features we describe in the following sections require the user to run a more specialized client program. This program passes normal user commands and responses to and from the server as before, but also interprets special client commands from the server. These commands can cause the client to perform actions like creating windows, setting up audio/video connections, and fetching files from anonymous FTP servers.

While the MOO server is extensible by users creating and programming objects in the database, the client itself is not directly programmable. However, the special commands that the client receives from the MOO code in the server are fairly general in scope, so it can present a wide range of behaviors to the user.

2.2: Window-based user interfaces

The first addition to the usual MUD model is support for graphical user interfaces under the control of MOO programs. This allows MUD objects to interact with users via windows that appear on their screens. For example, “opening” a map object might display a diagram of the layout of the local rooms, allowing the user to move between them simply by clicking with the mouse. Window-based interfaces can also be used for various shared applications, such as text or graphics editors that appear on more than one user’s screen at once.

Our protocol for communicating between the server and client programs uses high-level descriptions of the interface elements, and borrows ideas from the FormsVBT package for Modula-3. [1][3] MOO code in the server sends to the client a description of the names and layout of a number of user-interface widgets. The client interprets this description and creates a corresponding window on the user’s screen. When the user manipulates the interface widgets, messages such as “this button was pushed” are returned to the MOO code. Low-level interactions, such as

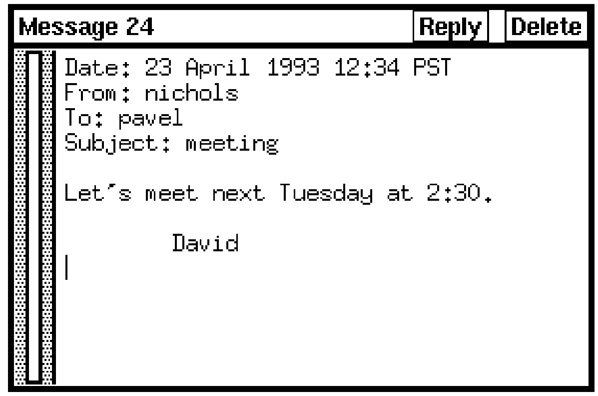

Figure 3: Example result window

scrolling text or echoing typed characters, are handled locally in the client and not reported to the server. MOO application code can respond to input events by changing the contents or appearance of widgets or other typical window-system operations.

This approach allows us to put as much of the window-system code as possible in the client, as opposed to the server, thereby reducing server load and making it easier to write new window-based MUD applications. It also reduces network bandwidth since the communication is in terms of high-level interface element events instead of low-level mouse and keyboard events. Finally, it allows for a great deal of interoperability, since the details of each user’s local window system are hidden from the MOO program controlling the windows; thus, for example, one user of a shared window application might be using a PC running Windows while another uses a Unix workstation running X.

2.3: Shared tool access

One of the principal advantages of MUDs is that all the users share the computing environment. If one user does something, others in the same room get some sort of indication of the action and can follow along. Our modifications provide this shared access for certain window-based applications. There are several approaches to building such a facility, and there are times when each is useful.

For applications written entirely in MOO, the code in the server simply drives more than one window. Each update is sent to all clients with a copy of the window, and events from any of them are processed as user input. The high-level interface to the windowing facilities keeps the number of interactions for the application to replicate to a minimum.

For terminal-based interactive tools, the server can act as a switch to mediate shared access to the tool. We have implemented this for Unix applications using XTerm,

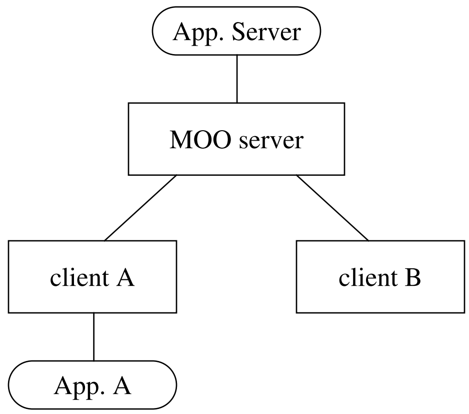

Figure 4: Shared access to applications

organized as shown in Figure 4. In this case, client A runs the application locally (App. A), and forwards all output to the MOO server, which repeats it to client B. Input from client B follows the reverse path, and is shown to the user at client A as well as being forwarded to the application.

We support shared access to network-accessible tools using a similar model also shown in Figure 4. Here, MOO code in the database directs the server to make a network connection directly to an application server, and the I/O is multiplexed among the various users as before. In some of these cases, we direct each user's client to open a separate XTerm window for the interaction; in other cases, it is more convenient for users to use MUD commands to manipulate the remote application.

We are not yet trying to provide shared access to X window system applications. However, we believe we may be able to do so by using one of the existing shared X servers, running on the same machine as the MOO server and under its control.

2.4: Audio

Audio in our MUDs works much as it does in real life; anything someone says is heard by all others present. We implement this by associating a multicast audio channel with each room. Users hear all the sounds from the room they're in and contribute to these sounds with their speech. When they move to a new room, their audio is automatically switched to the channel for the new room. Because the channel management is performed by MOO code inside the database, we can even arrange for the sounds from "nearby" virtual rooms to filter through at a lower volume, allowing users to notice when possibly-interesting interactions are happening "right outside their door."

We consider audio to be the most productive addition we could make to the usual MUD model. It substantially eases use of the communications aspects of the system for most users, especially those who are not good typists. Also, audio carries less ambiguously many of the non-language cues in conversation, such as tone, mood, emphasis, etc. Finally, studies by Chapanis have shown that audio is a substantially more productive aid for problem-solving communications than either text or video [4].

The audio is telephone-quality, 8 bits/sample at 8,000 samples/second, and is transmitted via multicast over the local network. We are currently using the audio devices built into our Sun SPARCstations, but any equivalent audio hardware (such as the current flock of PC sound boards) would do.

Unlike text and window events, audio and video must be delivered to the intended recipients within tight real-time bounds. Our central server-based architecture will not work for audio and video, because the overhead incurred by having the server receive and retransmit all the audio/video traffic would be too high. Instead, we have the clients send the traffic to each other directly using Internet multicast. Multicast datagrams are addressed to groups of workstations instead of individual ones. Because of this, several machines can receive the same copy of a packet, reducing the total load on the underlying network.

Even with multicast, the clients must know which multicast address to use to transmit and receive packets for a given room. Also, if the correspondence between other users and their network addresses is known, the client can provide speaker identification to the user by, say, highlighting the name of the current sender. This control information is low-bandwidth and not as time critical, and so can be easily managed by the central MOO server.

2.5: Video

While the addition of audio to MUDs provides a dramatic and obvious improvement in communication, the effect of adding real-time video is more subtle. It can enrich the perceived quality of computer-mediated conversation, as gestures, facial expressions, and physical surroundings are transmitted. Also, it makes it possible to monitor remote locations; for example, one can have a sense of whether or not someone else has been "around" recently, or if they are currently using the telephone by peeking into their office. Finally, video can allow users to participate in remote meetings where overhead slides or other visual props are used in presentations.

In our systems, we make it easy for users to view the output from any cameras associated either with other users in the same room or with the room itself. The latter case comes up, for example, when the MUD is used to attend meetings taking place in real rooms in the workplace.

At present, we are using Sun VideoPix boards and small attached cameras. We have software-only video compression algorithms that allow any workstation to display the resulting video, even if it is not equipped with a camera. The resulting video is 320×240-pixel, 7-bit grayscale at about 5 frames/second and takes about 128 Kbits/second of network bandwidth to transmit. As with audio, video is transmitted between clients using multicast, with the MOO server providing coordination of the video streams.

This video facility is not enough to support full video conferencing with audio/video lip-sync. However, it is sufficient to provide a sense of the activities of other users and for attending lectures via the MUD; in the latter case, one only needs to see the speaker's slides and hear the audio to get most of the benefit of a talk.

3: The Astro-VR system

In collaboration with David Van Buren, an astronomer at the NASA/JPL Infrared Processing and Analysis Center, we are building "Astro-VR," a social virtual reality intended for use by the international astronomy community. Our first serious users, from a major research project with investigators in both the United States and Italy, are beginning to use the system now.

The system is intended to provide a place for working astronomers to talk with one another, give short presentations, and otherwise collaborate on astronomical research. In most cases, this system will provide the only available means for active collaboration at a level beyond electronic mail and telephones.

Initially, Astro-VR will provide the following facilities of interest to our user community:

- real-time multi-user communication,
- a self-contained electronic mail and bulletin board system,
- shared, user-supplied *links* to online astronomical images,
- an editor/viewer for short presentations of text and images,
- collaborative access to standard programs used by astronomers, and
- window-based shared editors.

Astro-VR is built on top of LambdaMOO, so all of these facilities can be extended and customized by individual users, using the embedded programming language built into the server. We are not providing audio and video initially, since multicast capability is not yet widespread on the Internet, the home of Astro-VR's user community.

3.1: Real-time communication

The standard MUD means of textual self-expression and communication (e.g., speech, gestures, paging, whispering, self-description, etc.) are available to our users as well, because Astro-VR is, at heart, a MUD. While such communications channels are obviously not optimal, they are, at present, the best kind of conversational tools widely usable in today's global Internet. The wealth of experience already gained with recreational MUDs leads us to believe that this level of communications technology will prove useful and even sufficient until such time as something better (such as networked digitized voice) becomes feasible on a large scale.

For some situations, especially those involving planned collaborations between pairs of users, we expect that normal telephone calls will be used as an adjunct to the communication facilities we're providing.

3.2: Self-contained electronic mail

The international astronomical community, like most modern scientific communities, already makes a great deal of use of electronic mail. However, there remains a need for email forums both concerned with astronomical research and restricted to use by working astronomers. Bulletin boards and mailing lists such as the newsgroups of USENET have the problem (and virtue, perhaps) that access is unlimited; every user on USENET has the option of reading and posting to any newsgroup. From the perspective of a serious practitioner in some field, this communication channel is very "noisy." The `sci.space` newsgroup, for example, is posted to by people with all levels of astronomical background and expertise, from utter novices to working professionals. The general level of discourse is thus driven toward the middle ground, the knowledgeable hobbyists. Clearly, such a forum is inappropriate and ineffective as an outlet for serious discussion between experts.

The standard LambdaMOO database includes an electronic mail and bulletin board system completely contained in the virtual reality; email sent on a MUD does not, in general, leave it and email from outside the MUD cannot come in. In effect, the MUD provides a self-contained electronic mail community; the users can send email and maintain bulletin boards that are open only to that community, even though the participants are geographically scattered. We expect the users to find that a much higher level of bulletin board discourse is possible inside Astro-VR than outside.

3.3: Active links to images

Astronomers, perhaps to a larger extent than many other professionals, rely heavily on detailed photographs and other images to convey their ideas and discoveries to one another. At present, however, it is difficult for them to make their images easily available to one another. For example, it is difficult to discover if any (local or remote) colleague has online an image of any particular celestial object, let alone to get access to that image and display it on a given computer screen. In Astro-VR, it is easy for users to "register" the images in their collections, creating an object in the virtual reality to represent it. When another user ''opens'' such an object, the corresponding image is automatically fetched across the network and displayed in a window on the requestor's screen. These active *links* to images will be indexed in a number of ways on Astro-VR (e.g., by name, location, type of image, type of celestial object, etc.) for searching and browsing by other users.

3.4: Presentation editor and viewer

An important part of the scientific work practice is the periodic meeting of relatively small working groups, usually having 5 to 20 members. At such meetings, individual project members frequently give short, 10 to 20 minute presentations on recent efforts and/or results. To facilitate this kind of activity within Astro-VR, we are providing a set of *conference rooms*, which have extra commands for preparing and presenting such short talks.

A *talk* is structured as a sequence of *segments*, each containing a paragraph or two of text and one image, accessed using the active links discussed above. When each segment is presented, its text is printed out to all participants and the associated image is displayed on their screens. A distinguished user, the *presenter*, controls the "pace" of the talk by deciding when to move from one segment to the next.

Any discussions that take place during the presentation are recorded and stored with the talk. These discussion may be displayed during subsequent presentations of the talk. This allows project members who were not present at the initial presentation to catch up on what happened there and even to add their own annotations to the talk for other listeners. All talks are potentially archived for viewing by interested parties at any later date.

3.5: Collaborative access to standard tools

There are a number of computational tools frequently used by most working astronomers. For example, tools like the interactive plotting system mongo, the symbolic algebra system Mathematica, a variety of astronomical database systems, and even simple calculators all qualify as major tools in their day-to-day work. Through Astro-VR, we allow astronomers to use these familiar tools collaboratively. Using these tools is like crowding around the screen of a shared workstation, handing the keyboard and/or mouse back and forth and discussing the results produced. In the case of Astro-VR, though, the shared workstation is a virtual construct and the users can "crowd around" from the comfort of their individual offices around the world.

It is important to note that we are not, in this case, attempting to provide any new tools for our user community; there are many more knowledgeable people in astronomy doing that already. Rather, we see our task as providing the "glue" that allows collaborative use of those tools the astronomers are already using.

3.6: Window-based shared editors

Astronomers frequently talk about the orientations of and physical relations between celestial objects, usually communicating these ideas through sketches on paper or whiteboards. They also use sketches often for other purposes, such as conveying the general shape of some plotted data or function. These sketches are so important to their work practice that some astronomers even make these drawings as part of telephone calls, when the other party cannot see the image.

To support this style of interaction in Astro-VR, we are building a few especially important window-based MOO applications, the first of which is a simple shared drawing tool. This allows each user to see the strokes or figures laid down by the others and to "point" to parts of the drawing in ways that are visible to each other. Given the facilities provided in the window-system client program, such an application requires only a few hundred lines of MOO server code.

Another likely candidate for early implementation is a shared text editor with similar "pointing" capabilities, for communicating about relatively short pieces of plain text, such as paragraphs out of a research paper or the results of a collaborative database search.

4: The Jupiter system

To a large extent, Astro-VR represents the level of interface functionality that we can easily export to a large number of users, given the current state of the global Internet and the kinds of workstations typically available to our users. In an environment where we have much more control over the network and the workstations in use by the participants, though, we can explore the use of more advanced technologies. Then, as the Internet and users' workstations improve, we can use the lessons learned in our more controlled environment to improve the general lot.

Our other major effort, then, is an extended MUD that supports all the features mentioned in section 2. This system, called "Jupiter," is used by researchers both here at PARC and at other Xerox research and development centers. In addition, it supports users both within the laboratory buildings and at their homes.

To support this, we are arranging for most workstations at PARC to be equipped with microphones and, at the option of the user, a video camera. Our local network infrastructure already supports multicast routing, and we have high-speed ISDN telephone links to the homes of many PARC researchers.

4.1: Casual interaction

One of our goals in building Jupiter is to support casual interactions between its participants and to allow them to participate in the casual interactions taking place in the real environment of the laboratory. The use of audio is clearly important for this, because casual interactions must be essentially effortless, and speaking is easier than typing. We provide other kinds of support, as well.

Informal meeting areas are one example. In addition to the meeting rooms and offices, we have built areas that encourage casual interaction. The presence of a user in such a room is an implicit invitation for conversation from others. We keep casual-use objects there, such as newspapers (automatically generated from wire feeds) and games. It is important that users be able to use these objects together, as this encourages conversation.

We also provide facilities to encourage chance encounters. One example is the use of computer-generated sounds to notify users of the arrival or passage of other users. These electronic "footsteps" remind a user to check to see who else was in the area. Another example is mixing in a small amount of audio from adjacent rooms, if their doors are open. These would provide the snippets of speech that often draw newcomers into a conversation.

Finally, we are equipping the casual meeting areas of our laboratory building (such as coffee alcoves and lounges) with audio and video devices. These should allow Jupiter users to "happen upon" people in those areas and then to hold informal conversations with them.

4.2: Telecommuting

Many people currently spend some portion of their average week working from home, or *telecommuting*. In our experience, such people are effectively absent from the social milieu of the workplace, only "visible" in the form of occasional electronic mail messages, if that. We are interested in exploring the possibility that Jupiter can support a more effective form of telecommuting, in which even remote workers can be active social participants.

Jupiter's relatively modest network bandwidth requirements, coupled with recent advances in telephone connection quality, should allow workers to use Jupiter almost as effectively from home as from work.

ISDN telephone service is slowly becoming available from local phone companies throughout the world, and provides two 64-kilobit/second full-duplex channels per user at affordable prices. Jupiter fits audio, windows, and varying amounts of video over such a connection. Audio can be compressed to as little as 13kb/sec, using standardized algorithms such as GSM [8], without major losses in quality. Our window-system protocol takes much less than half a channel because the interactions are at a high level of abstraction. This leaves one channel to use for video.

Our experimental video protocols can send 5 frames/sec of grayscale video using 128kb/sec of bandwidth. Using only 64kb/sec, the frame-rate is lower but still tolerable, especially for largely static scenes such as workers in their offices. For a talk, we can send single frames of each slide, which easily fit into a single 64kb/sec channel.

The goal of telecommuting is to allow a worker to accomplish as much from home as they can by physically going to work. While Jupiter cannot be a full replacement for being there, it should provide a good substitute in many structured situations, such as meetings. Beyond that, its value as a replacement for travelling to work will depend largely upon the extent to which the goals of casual interaction are met.

4.3: Convolving the real and virtual worlds

There are a number of ways in which it appears to be useful to convolve the Jupiter user's perceptions of the real and virtual worlds. The most obvious example, alluded to earlier, is to have virtual rooms that correspond closely to certain physical places, such as common areas and conference rooms, allowing virtual visitors to encounter and communicate easily with people who are physically present. This is only one such possible connection, though.

It would not be difficult, for example, to tie a particular telephone line here at PARC to an virtual telephone handset in Jupiter. If a call came into that line, the virtual phone would "ring", giving indications to the users in the same location via text and audio. If one of them were then to "answer" the virtual phone, the audio from the call would be patched into the room's multicast audio channel, for all in the room to hear and respond to. A user could "pick up" the handset and carry it to other virtual locations, with the phone audio following the handset to other channels as appropriate. Of course, the phone call could as easily be made in the other direction, outbound from Jupiter, and the same sort of model could be used for fax transmissions.

Another possibility is a virtual bulletin board represented in the physical world by a large-scale flat-panel display in some appropriate place. Either interface could be used to add items visible to both the real and virtual incarnations.

With appropriate sensors in users' offices, we could reflect into the virtual world such important physical cues as closing the office door, or having a telephone off-hook. These could be presented to potential visitors in the virtual world to let them know that interruptions might not be appropriate at the current time.

One particularly intriguing direction in this area is a connection to the PARC building management computer system, allowing Jupiter users both to monitor conditions (by moving to the virtual analog of a physical location of interest) and to control them, using MOO programs. We have specific plans to investigate this direction in the near future.

4.4: Use for administrative support

A goal of the Jupiter system is that it tie together all of the workers here at PARC, not merely the research staff. There are a number of exciting scenarios possible if we can achieve this. We envision being able to visit the PARC library through Jupiter and to get the online assistance of the staff there in learning how to use their specialized interfaces to a variety of information sources. We would like to be able to confer with people in the accounting and purchasing departments concerning such matters as expense reports and purchase orders while sharing online views of those items. Our facilities staff, who maintain the physical plant of the building, could be queried regarding office furniture arrangements or telephone and network connections using a shared office map and furniture layout tool. In all of these cases, we imagine communicating easily with these people from our own offices using application-specific props and tools, preferably of the specialists' own design.

5: Project status

Much of Astro-VR is running and it is now accepting its first users. Jupiter is also operational, with essentially all of the technologies described above in daily use. In both cases, we are in the process of growing the installed base of applications preparatory to attempting to engage a larger community of serious, day-to-day users.

References

[1] Avrahami, Gideon, Kenneth P. Brooks, and Marc H. Brown, "A Two-View Approach to Constructing User Interfaces," in the SIGGRAPH '89 Conference Proceedings*, Computer Graphics* 23(3), pp 137-146, July 1989.

[2] Bartle, Richard, "Interactive Multi-User Computer Games," MUSE Ltd. Research Report, December 1990. Available via FTP from parcftp.xerox.com in pub/MOO/papers/mudreport.*.

[3] Brown, Marc H. and Jim Meehan, *FormsVBT Reference Manual*. Available as part of the Modula-3 distribution from gatekeeper.dec.com.

[4] Chapanis, Alhponse, "Interactive Human Communication," in *Computer-Supported Cooperative Work: A Book of Readings*, edited by Irene Greif, Morgan Kaufmann Publishers, 1988.

[5] Curtis, Pavel, "The LambdaMOO Programmer's Manual," available via FTP from parcftp.xerox.com in pub/MOO/ProgrammersManual.*.

[6] Curtis, Pavel, "Mudding: Social Phenomena in Text-Based Virtual Realities," in the Proceedings of the 1992 Conference on Directions and Implications of Advanced Computing, Berkeley, May 1992. Also available as Xerox PARC technical report CSL-92-4.

[7] Deering, Stephen E. and David R. Cheriton, "Multicast Routing in Datagram Internetworks and Extended LANs," ACM Transactions on Computer Systems, vol. 8, no. 2, May 1990.

[8] Rahnema, Moe, "Overview of the GSM System and Protocol Architecture," IEEE Communications, vol. 31, no. 4, p. 92, April 1993.