Automatically Generated 3D Virtual Environments for Multimedia Communication

Dorée Duncan Seligmann John T. Edmark

Bell Laboratories
Lucent Technologies
Room 4F-605
101 Crawfords Corner Road
Holmdel, NJ 07733
U.S.A.
908-949-4290
doree@bell-labs.com

ABSTRACT

We describe how we use 3D graphics in order to enhance multimedia communication. We have developed a system which automatically generates a virtual environment consisting of a dynamic visualization of, and control mechanisms for, multimedia interaction. Our system automatically generates customized displays of the environment for each user, presenting to the user visual cues that are created explicitly to convey information about herself, others, services, and devices. We have developed a suite of self-modifying intelligent objects, constrained viewers, context-sensitive annotation objects, and intelligent animated objects. The approach described is suitable for a wide variety of applications, including conferencing, long-term collaboration, chat groups, Web browsing, live performance, and on-line help; the architecture accommodates users with hybrid configurations of platforms, services, and peripherals.

INTRODUCTION

We have developed a system which automatically generates a virtual environment consisting of both a dynamic visualization of, and control mechanisms for, multimedia interaction. Our system automatically generates customized displays of the environment for each user. In addition, every user is presented with cues that are created explicitly to convey information about themselves, others, and the activities in virtual and real places. Our system automatically generates and positions 3D representations of both real and virtual places, the objects and people in them, as well as conceptual relationships, thus creating the environment itself. The objects appearing in the visualization are not simply representational, they are dynamic objects; some are managed media-specific objects (e.g. the displays of a shared application program); some show connectivity; others are the interface objects that afford users control over the system. The graphical objects include methods, constraints, and information that are used to automate their display, geometries, and rendering parameters; these objects also automatically reconfigure to reflect changes in the environment. The viewers are also constrained by the information they are designed to show, and dynamically modify themselves as situations change.

We also have developed libraries of 1) intelligent objects that are self-modifying, reacting to changes in the environment, user actions, and viewing parameters; 2) viewers that are constrained by the information they are designed to show; 3) annotation objects that convey concepts about objects in the environment; and 4) intelligent animated objects. These objects convey the events and state of the communication system.

THE PROBLEM

Heretofore, complex multimedia systems have been unwieldy and difficult to comprehend and, as a result, of questionable value. Natural conversation and interaction have been difficult to accomplish in multimedia systems, and the systems themselves are typically difficult to use. Users' experiences with such systems contrast sharply with that provided by the telephone which has become a transparent instrument by which we can engage in natural conversation.

For several years, we have been exploring ways in which we can render complex multimedia systems as intuitive as the phone has become. Using 3D graphic interfaces, we have developed methods for providing users the same type of information conveyed during the course of a simple telephone call. Our use of graphics enables us to view all communication and interaction occurring within the context of a place. Our model allows for both persistence and a concept of groups. The virtual environment exists even when no one is using it; the places within it have a history and will persist over time, thereby providing electronic rendezvous locales. Places are occupied by people and objects. The people who enter and leave a given place become members of the same group. Thus, in our environment, all the people who have visited a specific Web page are members of a group associated with that page.

TYPE OF INFORMATION CONVEYED

Our virtual environment consists of objects designed to convey a rich representation of the services and their current state information to each user, thus enhancing ease-of-use by providing users with information which will liberate them from continuously having to check on their system[SE95]. For example, people engaged in multimedia conferencing using existing technology often interrupt their conversation to verify the status of the video connections, (asking "can you see me?") or to find out what someone is doing (asking "are you typing?"). Yet by using video and audio cues our system automatically provides each user with the type of information she needs thus, enabling more natural interactions. This is only possible through the infrastructure we have built that consists of sets of protocols, and cooperating media services[AH95].

[Note: All the examples in this paper are taken from our meeting room service (MR). MR is a network-based service for multimedia conferencing. MR manages several other media services within the context of a virtual meeting room: a service that enables application program sharing; a 3D sound service that provides spatial audio for 3D audio cues and 3D realtime conversation [GA91, SH91]; a service that manages both talking head video, shared video, and surveillance video; a service that manages phone calls using regular phones and ISDN phones with RS-232 communication ports; and a service for messaging.]

COMMUNICATION OBJECTS

Our visualization of the virtual environment depicts the real world augmented with virtual places (the visualization of the real world is, of course, incomplete). We provide our system with a simple description of real places: the floor plan and devices used in multimedia communication (such as telephones and computers). *Structure objects* define the contours of a place: its floor and walls and are configured automatically: for real places, the system follows the floor plan; for virtual places the system selects a location. Our system creates support objects for the devices and people.

People objects are color-coded, and the representation of each person reflects his current context in the virtual environment; ghosting is used to indicate temporary absence -- (a person engaged in a phone call is ghosted in the real world). Currently, we use a set of full-body

photographs of each person that are image processed. Originally, we experimented with 3D texture-mapped 3D human models and have since opted for stylized 2D images. We found that the more realistic the representation, the more disappointing it was in its shortcomings. A photograph was disappointing because it was not live video; live video was disappointing because it was only from one vantage point.

All the objects in the environment are derived from a common base object definition. There are several versions for each object: one for the server and one for each of the clients (used for rendering and controlling media services). They all have identifiers, placement methods, and reconfiguration methods.

Identifiers
Every object is assigned a unique identifier that is global to the entire system. These names allow the server to send messages to update objects even though their geometric representations may be different. For example, a person has the same identity across media services. At the same time, any object may have multiple counterparts in different virtual contexts. The identifier is augmented to reflect membership and role in a virtual context.

Using a consistent naming scheme across all media services enables our system to identify parts of the same whole. This is important because each media service represents only one aspect of each object: hence, a phone service handles Karen's voice, the video service handles Karen's talking head; an application sharing service handles Karen's keyclicks into a shared program, and so on. This way, our system is able to represent Karen as a whole entity, using information about each part reported by each media service.

Some objects are associated with locations, people, and things. For example, the phone that John is sitting next to, in the real world, is associated with John, reflected in the augmented identifier. This affords us the flexibility to disassociate and associate objects with people as circumstances change and people move. Object associations are important because they allow users to dynamically change configurations of hardware, transport mechanisms, services, in addition to their actual location. Thus, wherever Karen travels to in the virtual spaces, so do the objects considered to be in her immediate environment.

Placement Methods
Each object is embedded with object-specific knowledge that specifies how its translation, rotation, and scale are calculated relative to other objects. When a new object is created, the server assigns it a location by calling the object's *placeIt* method with objects—*not* explicit coordinates—as parameters. For example, a table is placed relative to the floor and walls, a cable is placed between two objects, and a pointer is positioned relative to a person and an object of interest. Objects are moved only by calling their own placeIt methods. Thus, all object-specific knowledge is provided by the object itself, enabling new libraries of objects to be introduced at any time. This allows for extensibility: a new service can be linked in along with several new objects.

Automatically Triggered Reconfiguration Methods
When an object is positioned, it becomes "attached" to those objects that were passed as parameters to its placeIt routine. When those objects move, the reconfiguration methods are triggered. So, for example, when either of two objects connected by a cable moves, the length and path of that cable is reconfigured accordingly.

MEDIA-SPECIFIC FEEDBACK

Here we will describe how we are using 3D graphics to show the state of (i.e. answer questions about) a particular media service. The examples below are based on the scenario in Figure 1. The application sharing service is associated with the virtual room shown and a Netscape browser is being shared.

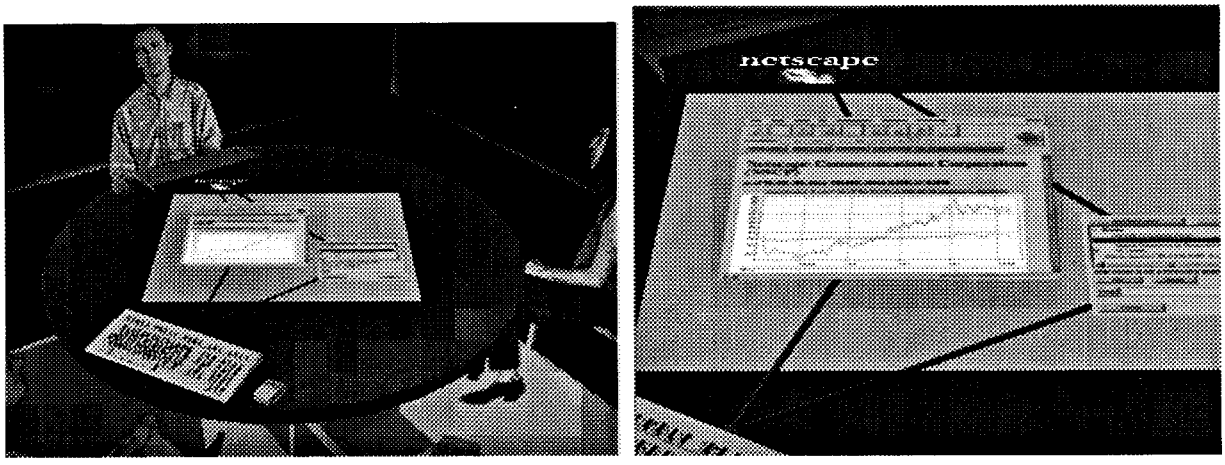

Figure 1 Three people in a virtual meeting room with the application sharing service. The absence of a keyboard and mouse indicates that the "yellow" person cannot share applications. The hand holding the word "Netscape" identifies the shared program. The red input cables connecting the keyboards to the windows indicate both persons can provide input to the program. The last keyboard to provide input is highlighted.

WHAT SERVICES ARE ACTIVE? A new object was created to indicate that the application program sharing service has been associated with the room. The blue area on the table represents the shared display. It contains the shared application windows (reflecting the exact configuration of the real windows on that user's monitor).

WHO CAN USE THIS SERVICE? Note that not all participants may have access to a given service. When the application service was associated with the room, objects representing the devices used to interact with this service were generated and placed on the table. In this case, keyboards and mice were generated and placed in front of each of the participants that are using the shared application service.

WHAT OBJECTS ARE BEING SHARED? Application icons were generated to indicate the applications brought into the room. In this case, the hand holding the word "Netscape" indicates that it is the only application program currently being shared.

WHO MAY CONTROL THIS OBJECT? Shared objects can be managed so that only specific people can control them. *Input cables* were generated, connecting keyboards to the windows, to show which users may type into the program.

WHAT IS HAPPENING? Visual feedback (coupled with 3D audio cues) is triggered by media-specific events. Certain device objects have methods for media-specific events. For instance, as the user types, the appropriate keys on the modeled keyboard move up and down (coupled with audio sampled keyclicks in the 3D space), and letters travel through the cable to the window.

WHO OR WHAT HAS INPUT FOCUS? Window managers use highlighting to indicate the current input focus of the local keyboard. Here, the last device used is highlighted, making it clear who was the last person to provide input in the shared object.

WHAT ARE YOU REFERRING TO? Interaction objects are generated to represent service-specific reference objects. In this case, the application sharing service allows us to place pointers on shared windows. Therefore, a pointer object is generated, color-coded, and positioned to show who is pointing.

CONTEXT-SENSITIVE OBJECTS

Some objects reconfigure themselves based on changes in the context. Objects in the server have sensors to monitor certain aspects of the environment. For example, a conference table's size is based on both the number of people in the room and the objects placed on it. The reconfiguration method is triggered whenever there is a change (when people are added to or removed from the room, or when objects are placed on or removed from the table). People objects track changes in the tables at which they are seated, and relocate themselves as the table shrinks or grows. In this case, an object can be endowed with behaviors that are analogous to real world behaviors. For example, the people at a table will move aside to make space for a newcomer.

VIEW-SENSITIVE OBJECTS

Whereas Friedell's system[FR83] automatically generated objects based on the constraints of the view and information to be shown, we are instead building objects adaptive to changes in the view parameters. These objects have methods using object-specific knowledge to change attributes. The person object consists of a set of images of a person from different angles. It selects what image to use based on its relationship to the current camera position using a sensor mechanism that is triggered each time the relevant camera parameters change. Figure 2 shows people using both the profile and frontal views. Also in Figure 2, the shared display, which contains shared application windows, is another example of an adaptive object. It is constrained to remain legible and is always oriented in a upright position relative to the camera. This is an improvement over the real world in which a document on a table can only face one person.

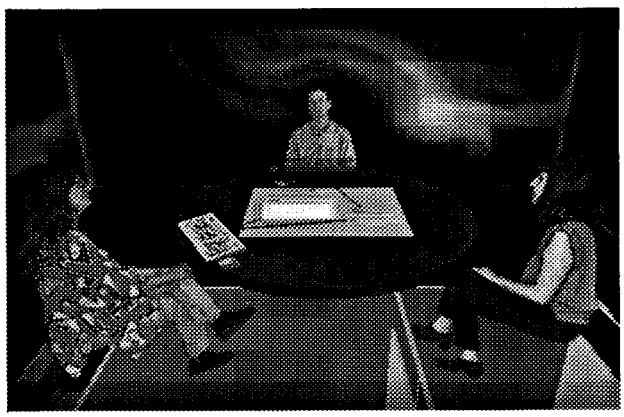

Figure 2 View-sensitive objects: people and shared display. Each people object selects an alternative representation as the camera settings change. The shared display (the blue area on the table) reorients itself to remain in an upright position relative to the camera. (Compare its orientation in this view to that in Figure 1.)

CUSTOMIZED VIEWERS

A general viewer enables the user to navigate in the 3D environment. However, a single view may not be able to show everything a user needs to know at the same time. Therefore, we have designed a suite of context-sensitive viewers. Each is bound to communicate a specific set of concepts and dynamically modifies itself as the situation changes[SE93]. Each viewer is assigned a goal to achieve, such as to show the location of an object in a certain context, or to show a set of objects (and ensure that they are visible and recognizable).

WHO IS THERE? It is very useful to know who else is with you in a virtual context (from the people on multiparty line, to the people in a chat room). Figure 3 shows the *Panoramic Viewer* as it changes state when people enter the virtual meeting room. The viewer updates its location, orientation and perspective (viewing angle) based on the number, location and size of the

Figure 3 The Panoramic Viewer: as people join the conference. The camera's location corresponds to the user's head location in the virtual room (the user is "seated" at the table). Initially, the user is alone, and the empty table is shown. When people enter the room (and are placed at the slightly larger table) the viewer modifies the view parameters.

participants in the room from the user's vantage point. The panoramic viewer also provides a visual representation of the 3D sound space (in which each participant's voice is convolved to emanate from his or her respective location).

WHAT IS BEING SHARED? During remote collaboration it is often unclear what objects are accessible to the participants in the virtual context. Figure 4 shows the *Shared Surface Viewer,* which locates and orients itself to depict all the objects on the shared work surface, in this case, the table.

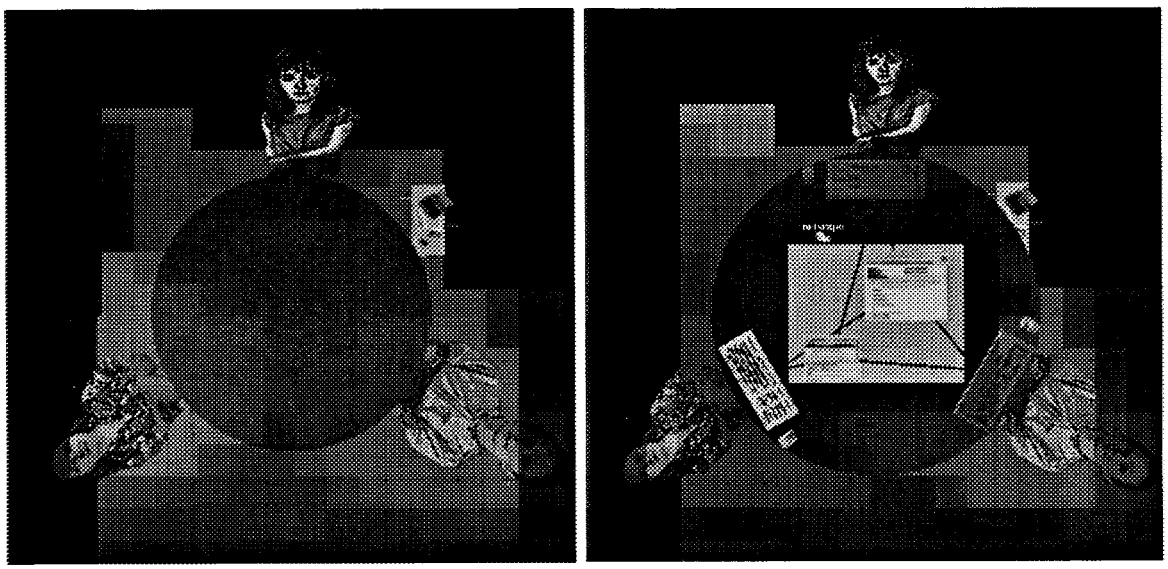

Figure 4 The Shared Surface Viewer: before and during application sharing. When the application sharing service was brought into the room, objects were placed on the table, causing it to grow. The viewer automatically changed the camera setting so that the participants and shared objects remain visible.

WHAT DO I HAVE? The *Local Viewer* shows all the devices and services available to the user. This viewer locates and orients itself to show the user's complete local area, consisting of both the user's real-world environment (e.g. office, home, etc.) and the currently accessible service devices (such as video cameras, computers, phones, etc.).

AUGMENTING WITH ANNOTATIONS

The user can request additional information, thus causing the client to augment the 3D space with annotation objects designed to show various types of information. These objects are local, and only appear in the user's viewer, without disrupting the other users. Thus, the user is able to quickly access information about the environment without interrupting interaction.

HOW ARE THINGS CONNECTED? Connection cables show how the various devices are connected. Figure 5 shows the color-coded cables that are automatically created using the information cached about how each video source and destination is bridged. In this example, a user can quickly determine who is looking at whose talking head.

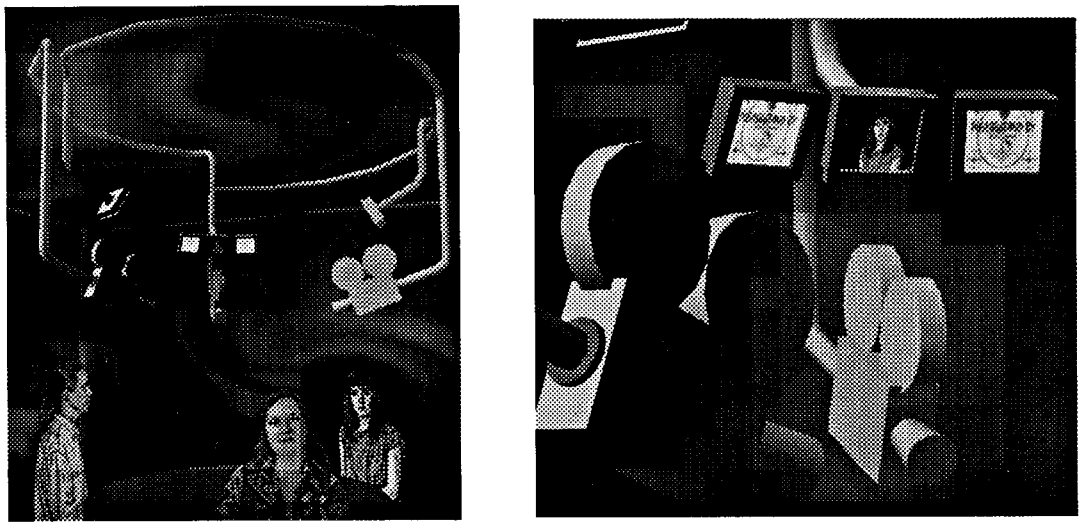

Figure 5 View augmented with connection cables to show how video devices are bridged. Video devices are color-coded to match their owners. The "red" person has three monitors, of which only one is in use. It is connected to the "yellow" person's camera. This camera's location and angle was used to create the image of the virtual environment which is texture-mapped onto the monitor, thus "showing" what is being displayed.

WHAT DOES THIS OBJECT CORRESPOND TO? Association cables show what objects are associated with each other. Figure 6 shows the color-coded "dotted" cables that indicate the real-world counterparts for every video device in a particular virtual place. In this example, a user can quickly determine where a video device in a virtual place is actually located in the real world.

WHAT IS THIS? Textual labels are placed on objects to identify them. These label objects are constrained to remain legible; they change size and orientation based on the viewing parameters.

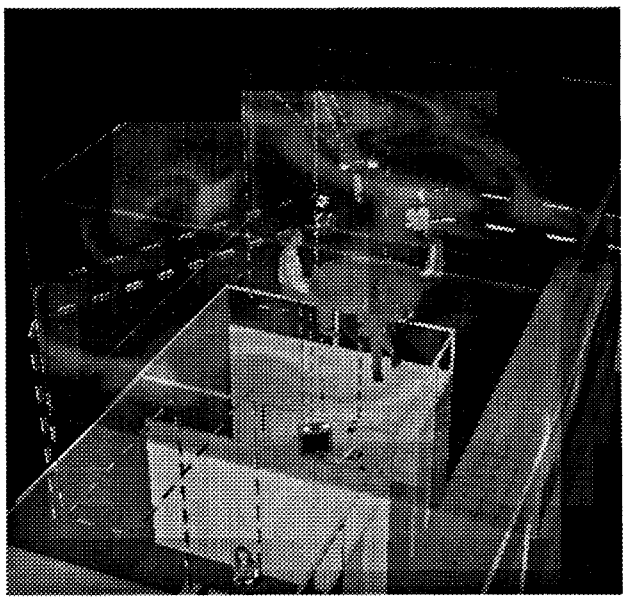

Figure 6 View augmented with association cables to show the relationship between virtual and real objects. The "dotted" cables connect devices in the virtual room to their real-world counterparts in the offices below. (They are color-coded to match their owners.)

CONTENT

Although the media services often provide content separately, it may still be useful or desirable to have some representation of that content in the virtual environment. For example, a user might want to verify what is being displayed on a remote video monitor. Figure 7 shows a grabbed frame of video in a video monitor object. The frame is retrieved by the video service. Alternatively, the video service can use local hardware to display live video directly into texture-memory. Figure 1 shows live windows texture-mapped into the 3D

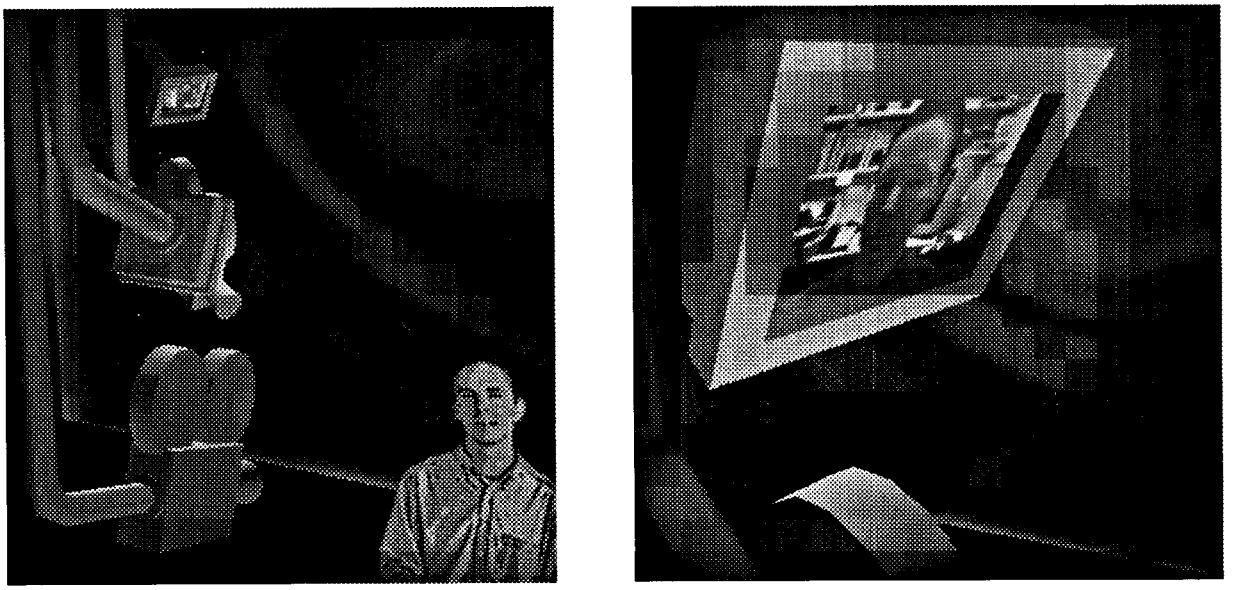

Figure 7 Content Objects: live video. The user can view the actual video streams being sent to the various devices.

scene[DY93]. The content of the window is retrieved using the shared application service.

ANIMATIONS

Certain objects are animated. We have developed a generalized method for showing data flow. As described earlier, cable objects are given two locations and constraints, and instantiate themselves. *Envelope* objects travel through these cables at different rates. Typically, an envelope is invisible and any object can be placed "inside" of it (with or without clipping). In the spirit of [BI93] in which filters are applied to show different aspects of objects in a scene, a user can request that the data streams be represented in different ways. Figure 8 shows letters traveling through the cable connecting a keyboard and window, illustrating the input stream of keyboard events to a window. The letter objects are constrained to remain upright and facing the camera. Some objects illustrate actions using frame-based animation. These objects contain methods for different activities which, when called, activate engines that cycle through the appropriate animation. Figure 9 shows the Dogzaic messenger running down the hallway to deliver a message. In this case, an invisible cable is created (the cable generates its own path using knowledge about

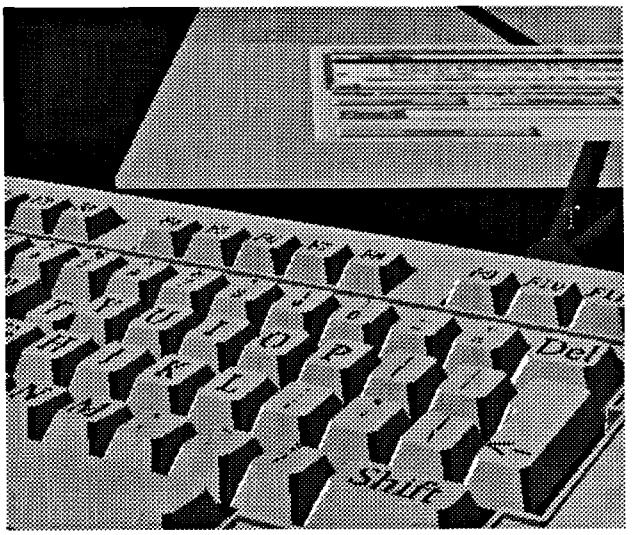

Figure 8 Animated objects show activity. The user has typed a "T" into a shared program. The keyboard is highlighted, the "T" key on the modeled keyboard moves and a 3D letter "T" travels through the input cable to the appropriate window.

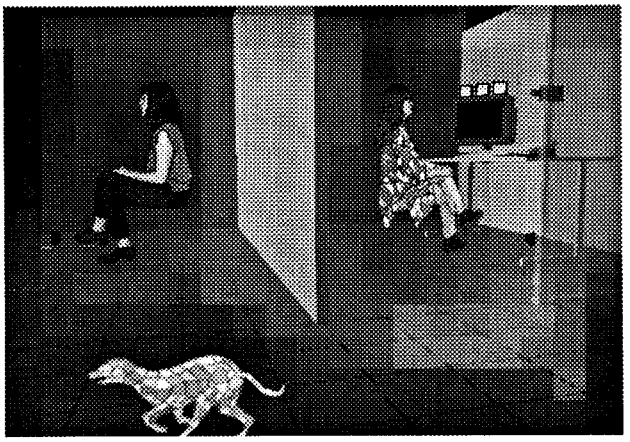

Figure 9 Animated objects: Dogzaic. Dogzaic travels through the hallways to deliver messages.

hallways) and the Dogzaic is sent through it. In both cases, the call made to initiate the animation is simple, consisting of object names and the method "send."

IMPLEMENTATION

All code is written in C++. We have created a library of visual objects, and libraries for each media service. 3D graphics are implemented using SGI's OpenInventor[WE94] and OpenGL[NE93]. Constraints are implemented using our library of sensors and engines in conjunction with the "connectFrom" field value provided by OpenInventor. We have created a suite of OpenInventor objects to drive the animations and sound effects. Some objects are multimedia, including frame-based texture-mapped animated objects with 3D sound cues. (All textures were createdby Cati Laporte using Adobe Photoshop on an Apple Macintosh.) Audio service ranges from ordinary telephones, ISDN phones with RS-232 communication ports, to 3D sound using Crystal River Engineering Inc. Convolvotron hardware, and most recently a CRE Acoustetron 2x3. Our system operates on SGI machines ranging from Indy's to a 4-processor Onyx Reality Engine. We run the server and other media servers on the Onyx. The client can run on any of our machines, including the Onyx. We have conducted experiments between two cities using a ppp connection over ISDN. The figures were generated on the Onyx Reality Engine.

CONCLUSIONS

Our task was to use 3D graphics in order to enhance multimedia communication. We conclude that graphics can be used to provide visual cues that convey information to help users interact more naturally and with greater confidence using complex multimedia systems. To this end, we have adopted knowledge-based 3D graphics techniques to build a system to automatically generate visualizations of, and control mechanisms for, complex and dynamic virtual environments. We have shown how we use a combination of intelligent objects, constrained viewers, annotation objects, and animations to convey the events and state of the system. Every object serves a specified purpose, and each view is customized for a particular user. In addition, our architecture allows for shared dynamic virtual environments without high bandwidth requirements. Although, we have shown examples for just one application, multimedia conferencing and collaboration, our approach is applicable for a wide variety of multimedia applications.

FUTURE WORK

The most time-consuming aspect of developing our system has been the authoring of the objects. We need to augment standard 3D modeling tools so that we can seamlessly augment models with the attributes described in our paper. (For example, we should be able to create animation engines that are simply attached to objects.) We would like to develop more complex objects that are able to convey some of the more subtle and complex of human communication, such as synthesized backchannel responses[CA94]. We would like to put our system on the Web. We hope that efforts, such as VRML, will eventually enable dynamic environments like the one shown here. We would also like to extend our model to enable users to create personalized sub-worlds, allowing users to customized local versions of the environments (e.g. building "mansions" of frequently visited virtual rooms). We are also interested in developing a distributed version which ultimately could be used for MUDDs.

REFERENCES

[AH95] Ahuja, S.R., Ensor J.R., Seligmann, D.D., "Archways: Making Remote Multimedia Conversations Persistent and Natural" In *Proc. of Technology Summit Telecom '95,* Oct 2-7, 1995.

[BI93] Bier, E.A., Stone, M.C., Pier, K., Buxton, W., DeRose, T. "Toolglass and Magic Lenses: The See-Through Interface." In *Proc. of ACM SIGGRAPH '93*. Aug. 1-6, 1993. p. 73-80.

[CA94] Cassell, J., Pelachaud, C., Badler, N., Steedman, M., et al "Animated Conversation: Rule-based Generation of Facial Expression, Gesture & Spoken Intonation for Multiple Conversational Agents." In *Proc. ACM SIGGRAPH '94*, Orlando, FL, July 24–29, 1994.

[DY93] Dykstra, P., "X11 in Virtual Environments." In *Proceedings of the IEEE 1993 Symposium on Reserach Frontiers in Virtual Reality*, San Jose, California, October 25-26, 1993.

[FR83] Friedell, M. "Automatic Graphics Environment Synthesis." Ph.D. Thesis, Department of Computer Engineering and Science, Case Western Reserve University, 1983.

[GA91] Gaver, W. W., Smith, R. B., and O'Shea, T. "Effective Sounds in Complex Systems: The ARKola Simulation." In *Proceedings of ACM SIGCHI '91 Human Factors in Computing Systems*, New Orleans, Louisiana, April 27–May 2, 1991.

[NE93] Neider, Jackie. "OpenGL Programming Guide: the official guide to learning OpenGL", OpenGL Architecture Review Board; J.Neider, T. Davis, M. Woo. Addison-Wesley, New York, 1993.

[SE93] Seligmann, D.D. "Interactive Intent-Based Illustrations: A Visual Language for 3D Worlds." PhD Thesis, Dept. of Computer Science, Columbia, University. 1993.

[SH91] Shimizu, Y. "Research on the Use of Stereophonics in Teleconferences." In *Business Japan*, March, 1991.

[WE94] Wernecke, Josie, "The Inventor Mentor." Addison-Wesley Publishing Company, New York, 1994.

[SE95] Seligmann, D.D., Mercuri, R.T., Edmark, J.T. "Providing Assurances in a Multimedia Interactive Environment." In *Proceedings of ACM SIGCHI '95 Human Factors in Computing Systems*, Denver, Colorado, May 7-11, 1995