IEEE

MultiMedia

Spring 1996

50 YEARS OF SERVICE
IEEE COMPUTER SOCIETY
1948-1998

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

Departments

Circulation: *IEEE MultiMedia* (ISSN 1070-986X) is published quarterly by the IEEE Computer Society. IEEE Headquarters: 345 East 47th St., New York, NY 10017-2394. IEEE Computer Society Publications Office: 10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA 90720-1264; (714) 821-8380. IEEE Computer Society Headquarters: 1730 Massachusetts Ave., Washington, DC 20036-1903. Annual subscription: $22 in addition to any IEEE group or society dues. Members of other technical organizations: $31. Nonmember subscription rates are available on request. Single copies: members, $10; nonmembers, $20. This journal is also available in microfiche form.

Postmaster: Send address changes and undelivered copies to *IEEE MultiMedia,* IEEE Computer Society, 10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA 90720-1264. Second class postage is paid at New York, NY, and at additional mailing sites. Canadian GST #125634188. Printed in USA..

Interactive Multiuser VEs in the DIVE System

Olof Hagsand
Swedish Institute of Computer Science

Multiuser virtual environments (VEs) raise challenging research questions concerning how users interact with objects, applications, and other users, and how distributed VEs behave when the number of users increases. The Distributed Interactive Virtual Environment (DIVE) is a software platform for multiuser VEs that has served as a toolkit for many distributed VE applications. It emphasizes networking and human-computer interaction and supports autonomous behavior-driven objects, collision detection, and audio and 3D navigation.

A virtual environment (VE) is a real-time simulation of a real or imaginary world where users navigate and interact with 3D objects within it. In a fully interactive multiuser VE, several participants connected by a network may meet, collaborate, and work.

An appealing characteristic of VEs is the ability to offer intuitive modes of interaction, analogous to the ways in which humans communicate with each other or manipulate objects in the real world. VE applications can use 3D spatial properties to represent users and to model interaction, offer direct manipulation interfaces that mimic actions in the real world, and use immersive techniques that give participants the sense of being embedded in the synthetic environments. As such, distributed VEs can be seen as powerful human-computer interfaces. They thus provide new ways to communicate remotely and to access information over networks.

The work presented here centers on multiuser VEs. The discussion includes how to design and construct networked, multi-participant, wide-area, on-line VEs as well as how to create useful metaphors and techniques for communication between participants and applications within such environments.

DIVE is a multiuser, distributed VE system developed by myself and colleagues at the Swedish Institute of Computer Science (SICS). DIVE differs from similar approaches in its dynamic and flexible—some say anarchistic—capabilities and its focus on interaction and human-human communication. Starting as a lab tool in 1991, it has evolved into a well-developed system implemented on many Unix platforms and used to prototype VEs at several research sites around the world.

The figures used in the examples are snapshots from existing, real-time, functional applications of the DIVE system. The system, along with some examples in this article, is available over the World Wide Web at http://www.sics.se/dive/.

Multiuser VEs

An interactive multiuser VE is a distributed application where multiple users are simultaneously present within a simulated 3D space. We call such a shared environment a (virtual) world. All users perceive the same world by seeing the same scenes and hearing the same sounds, albeit from separate spatial locations. Each user can navigate through 3D space and see, meet, talk, and collaborate with other users in the environment. Worlds are populated by objects with or without graphical representations.

An example of a multiuser VE, Figure 1 illustrates a world seen from a user's viewpoint. The scene includes two other users or automated players (actors) with simple representations (embodiments). The actors stand in a room containing a couple of plants and a lamp illuminating the scene. A landscape is visible through a window.

The actors may move freely through the world. They may explore the building, move objects around, communicate with other participants (with gestures, audio, or text), and dynamically create and modify objects on the fly, for example by collaboratively modifying the 3D scene. Additionally, objects can have autonomous behaviors associated with them. For example, the lamp in the figure can be turned on or off by pointing to it.

The embodiments in Figure 1, called "blockies," are very simple representations of humans. The head of a blockie might correspond to a real user's head wearing a head-mounted display (HMD), while the eyes represent points in space from which a user views the virtual world, possibly in stereo. The movement of a data glove or other input device can be coupled to a virtual pointing device (such as a hand). In Figure 1, the user on the left has just turned on the light. The user's virtual pointing device appears as a white line.

Figure 2 shows a graphically more elaborate example: a conferencing application employing real-world metaphors extensively. (See the sidebar on p. 39—this example is on the Web.) Three actors communicate in three dimensions with text and

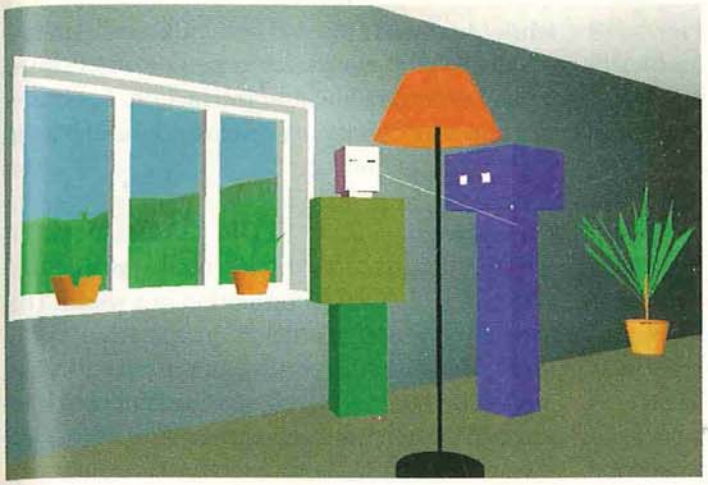

Figure 1. A world seen from a user's viewpoint. The scene includes two other actors embodied very simply, as "blockies."

audio in a traditional conferencing environment. The three documents on the table and the wall display are active, displaying text and graphics coupled to external tools in the outside (real) world.

Interaction with individual objects

An interactive multiuser VE, in which participants interact with objects constituting a 3D scene, must support the creation and modification of individual objects. Information is distributed on a per object basis, where individual objects are addressable and may be requested and retrieved over the network.

In a truly interactive environment, participants must be able to dynamically create, modify, and remove objects, as well as enable others to do the same. This means that objects are not owned by a creator—once introduced, an object may be accessed and modified by any participant.

If combined with autonomic behaviors, such a shared environment has enormous power. For example, an object created and released in a world can perform actions by itself and be picked up and used by other participants.

Why networking is important

Peers of such an advanced distributed application need to exchange large amounts of information, including object and world definitions, navigation commands, audio, and bitmaps. Over an internet, messages might have to be delivered among many participants over high-latency paths. Recent round-trip estimates measured in DIVE experiments range from 25 milliseconds within Sweden, 200 ms to American sites, and 800 ms to Japan.

In such an environment, it is crucial that participants experience "acceptable" delays. The ultimate endpoints of communication are the human senses and the brain's actual perception of sound and pictures. Typically, end-to-end latency as experienced by a human user has an upper bound for acceptability. For audio this limit is roughly on the order of 100 ms, while interactive manipulation requiring feedback has an even lower bound. Unfortunately, network and physical realities make these limits unreachable for global distributed systems. Other properties, such as object motion and presentation, might be less sensitive to latency, especially if described by behaviors evaluated locally at each peer.

System designers must therefore address protocol and performance issues and design the communication between peers to take advantage of the network's available bandwidth. In short, the amount of traffic between peers must be reduced and end-to-end latencies minimized.

Summarizing, two distinguishing features make interactive multiuser VEs possible: interaction with individual objects and scalable networking. We will later return to how these issues were addressed in DIVE.

For a discussion of networked multiuser VE systems, see the sidebar on the next page.

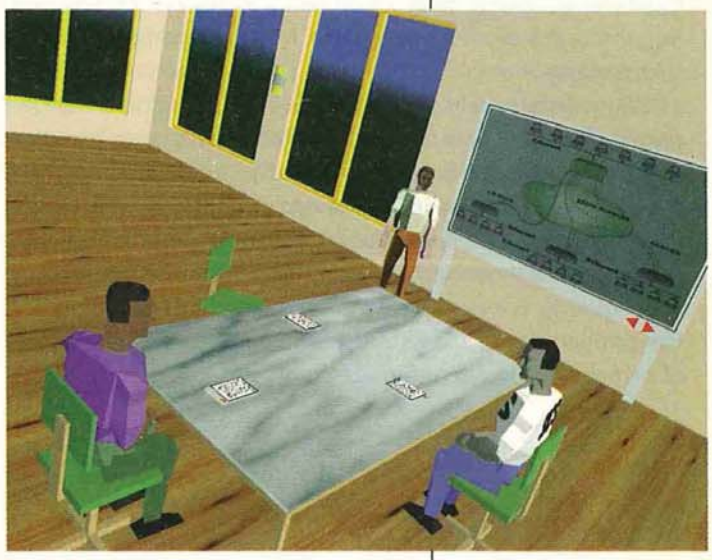

Figure 2. A conferencing application with three actors who can communicate using text and audio. The documents on the table and wall display are active, displaying text and graphics coupled to external tools. (See the sidebar on p. 39 to find three forms of this example on the Web.)

Networked Multiuser VE Systems

Several other multiuser VEs take distribution and multiuser aspects into consideration. The VUE system[2] is a distributed client-server architecture where processes communicate via asynchronous message passing. In the Minimal Reality (MR) Toolkit,[3] a set of master processes is connected pairwise to other master processes. Messages may be sent unreliably between one process and the other peers.

Several systems are based on an object-request broker approach. Their interfaces let objects be accessed remotely (by asynchronous message passing) through a client-server system. Massive[4] and BrickNet[5] are examples of such systems.

Many systems rely on the DIS protocol,[6] originally designed for multiparty training in combat situations. With this approach, multiple immobile objects form a static background, such as landscapes and buildings, while the movements of a smaller set of dynamic objects, such as vehicles, are distributed to all connected peers. One such system is NPSNet (Naval Postgraduate School Net),[7] in which large-scale distribution issues were taken very seriously. For example, NPSNet is currently addressing how to increase the number of participants and information space to the range of thousands of participants over an internet.

The DIVE software model

To understand the fundamentals of the DIVE environment, let's briefly consider the platform's software model.

Distributed entities

DIVE entities form the basic units of distribution that can be addressed, requested, and distributed. Figure 3 shows a class hierarchy (used only for modeling—DIVE is implemented in plain C) where the entity class is the top-level abstraction. An entity has a globally unique identifier used for addressing, a name, a behavior description, and a set of properties usable for application-specific data. The entity and view classes are abstract classes; that is, no instantiations are allowed.

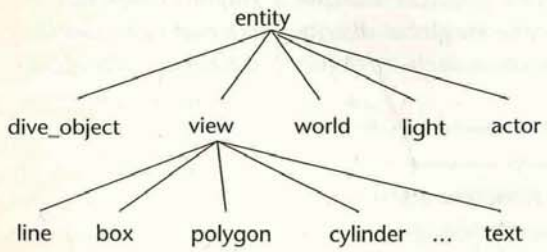

Figure 3. Class hierarchy of DIVE entities.

Entities are structured hierarchically in a tree: a world is a root, while dive_objects are nodes, views are leaves with 3D graphical representations, and lights are leaves with a light model definition.

Figure 4 shows an example of an entity hierarchy of the active lamp object shown in Figure 1. *lamp* and *active* are examples of dive_objects, while views and a light (a light bulb) are depicted graphically. The *active* object has an associated behavior triggered by user interaction. For example, selecting or stepping up close to the light activates the light bulb.

DIVE objects

DIVE objects carry the essential logical, interaction, and dynamic information in a world. This includes geometrical orientation, material descriptions, and variables controlling interaction and rendering. When DIVE objects are composed hierarchically, their own geometrical transformation is composed with the rotation and translation of the object at the next level in the hierarchy.

The following DIVE file format definition specifies the *lamp* object with a default black material and a displacement of 10 meters in the *z*-axis direction:

```
object {
  name "lamp"
  translation v 0 0 10
  material "black"
}
```

Graphical representations: Views

Lines, spheres, cylinders, boxes, grids, and polygons are examples of subclasses of the view class. Views are passive graphical 3D representations that may be dynamically created and modified. When a user interacts with a view, typically by pointing to it, the DIVE object closest to it in the hierarchy handles the actual interaction. For example, the *active* object in Figure 4 defines the behavior of the views and lights below it.

The lamp pole in Figure 4 is an example of a cylinder:

```
view {
  CYLINDER
    0.02
    0.02
    1.3
}
```

where 0.02 denotes the two radii and 1.3 the height of the cylinder.

Multicast domains: Worlds

A world represents a separate virtual space disjoint from other worlds, with its own set of objects, actors, and views. The world information is common to all entities within the space, such as background light, fog, and spatial boundaries.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.