

Research Issues in Developing Networked Virtual Realities

Workshop Report for “Distributed System Aspects of Sharing a Virtual Reality”

edited by

Michael Capps

<capps@mit.edu>

MIT Laboratory for Computer Science
Cambridge, MA USA

David Stotts

<stotts@cs.unc.edu>

U. of North Carolina Dept. of Computer Science
Chapel Hill, NC USA

Abstract

This is a report on the proceedings of the “Distributed System Aspects of Sharing a Virtual Reality” workshop at the 6th WET-ICE Conference held June 18-20, 1997. The workshop’s discussions focused on the research issues in the field of networked virtual reality, as well as the current methods used to address those issues. In those areas where the state of the art was felt to be lacking, we additionally report on the most promising technologies that the near future holds.

1. Introduction

The focus in this workshop was on networked shared virtual environments (VE). Such systems have these common properties:

- comprising multiple processes
- code and/or data is distributed across nodes in a network
- used simultaneously by multiple people
- collaborative, i.e., users are aware of, and interact with, one another
- present an abstract representation of some space in which users operate (“the world”)

Note that this list does not require a networked shared VE to have 3D graphics, but most of the ones here did.

We should record that at the time of the workshop the state-of-common-practice in networked VEs is a multi-user Internet-based action game called Quake. Quake is the follow-on to the popular Doom, expanded to allow

multiple players to construct 3D worlds and interact in them. It is distributed much like the Web, in that the Quake “universe” is a loose federation of independent worlds. A user may visit any world he knows of. Graphics are pre-rendered and displayed from the local client. A server maintains the state for a world, allowing the various connected clients to exchange movement and action information. Quake is extensible via a programming language for developing new worlds, new objects, and new behaviors (in the sense a MOO is extensible). There is no sharing of objects or states among different worlds, and persistence is provided only for the life of one server execution.

In the discussion after each workshop presentation, we attempted to identify the open problems suggested by the research. The resulting large list was coalesced into three broad categories:

- issues of content
- delivery
- architecture

In the following sections we outline the major issues from each of these areas, summarize current attempts at solutions, and identify potential solutions that are on the horizon but undeveloped.

2. Content

Content denotes the objects populating the world, as well as the physical extent, limitations, and characteristics of the world space itself. Specification of the content is currently not easy, but is getting better. Specification of the behavior of the content, and the constraints imposed by

the space, is also currently difficult.

The open problems in VE content can be grouped as:

- specifying structure and behavior of world components
- leverage from semantics, characteristics, ontology of objects and worlds
- environments supporting authoring

2.1 Content specification

Before users can interact in a VE, someone must create the initial content and behavior for the component virtual worlds. With current methods this requires a combination of 3D modeling skills, programming skills, and graphic art design skills. Proprietary formats and modeling notations from early graphics research are still in use, but several standards have evolved and are in widespread use. These standards include graphics packages like OpenGL (which is a model format and must then be supported with an external programming language) and VRML. VRML allows some scripting to give rudimentary behavior to objects, but this is far short of the desired ideal.

IDL (Interface Description Language) has been used in DIS projects as a means of formally defining the information flows into and out of specific processes in a distributed VE. This approach is effective for humans composing VEs but is less than what is needed for good automated composition, as it does not adequately define what specific transformations occur in a process, it is, in a sense, more syntax than semantics.

An interesting solution on the horizon for content and behavior description is Java3D from Sun. Several participants noted that Sun is still developing an initial implementation and that a usable system is a year away. Java3D is essentially the modeling parts of VRML but with Java added for semantics and scripting.

2.2 Content semantics

Semantics denotes the behavior of objects in a VE. Uses for semantics definitions include assistance in navigating and VE as well as support for VE interoperation and composition. We distinguished three forms of semantics:

- informal semantics
- formal semantics
- analyzable semantics

Informal semantics are text attachments that human can (perhaps) understand, telling the behavior, meaning, or purpose of an item in a VE space. Until major problems are solved in natural language understanding, no program

or automated agent will be able to make use of such semantics to assist in navigating or using a VE.

Formal semantics are expressed in some notation that can be manipulated by algorithm. From this point of view, any programming language can be used for formal semantics, and in fact, this is what happens in practice. The meaning or behavior of an object is encoded as a Java/C/C++ code block and then the semantics are realized by execution. However, nearly the same problem of automated understanding exists for this extreme as for the previous extreme (semantics in plain text).

Analyzable semantics are formal semantics expressed in a notation less powerful than arbitrary code. The goal is to use a notation that can be operated on by some inference engine, thereby allowing automated agents to “understand” the behavior of objects and make decisions based on the encoded information. If this problem were solved then other problems such as composing VEs and having VEs interoperate could be attacked on a deeper level. We see no broad effective solutions to the analyzable semantics problem, though components of a future solution are being researched.

Effective definition and use of semantics depends on a common knowledge base... a common domain of discourse for the many participants in a shared VE. The term “ontology” was used to denote this shared semantic domain. An ontology is a specification of conceptualizations used to help programs and humans share knowledge. In practice it consists of definitions of representational vocabulary, including axiomatic theories. Several well developed results exist from the AI community for building and using ontologies. KL-ONE is a well-known knowledge representation language. Interchange languages like KIF (Stanford) and engines like Ontolingua have been developed to allow disparate programs to share machine-encoded knowledge via translations. The KSL Interactive Ontology Server (Stanford) contains dozens of existing pre-defined ontologies for common domains as well as very specialized ones (like math).

We see a great need to make use of such research in developing and using ontologies and supporting semantics processing tools in shared VEs. Some of this research is going on now and is reported on in the papers here, but much remains to be done.

Will formal analyzable semantics eventually provide practical support in development and use of shared VEs? We concluded that this is an open question.

2.3 Authoring support

A third main problem area is authoring environments. An authoring environment provides systematic support for developing complex worlds, using

the specification notations mentioned previously. Progress is being made in this area, with several participants noting that sophisticated tools are appearing for VRML development in particular. The area seems ripe for much more work though.

2.4 Middleware

Some of the systems discussed in the workshop employ a middleware layer to interconnect VE components that are heterogeneous in various ways (execution platform, data format, functionality, resolution requirements, receiver bandwidth, etc.). Specifically mentioned were CORBA, Polyolith, and HLA/RTI. Of these, Polyolith is an experimental system being used for rapid prototyping; CORBA is an international standard and likely to play a large role in future VE middleware; HLA/RTI is a follow-on to the DIS program, and is a *de facto* standard due to the size of the funding for it coming from the US DoD.

Use of standard notations like VRML or Java3D will help the issues of interoperability and composition by eliminating some of the heterogeneity that is apparent among current VE designs. When all systems express their models in a common format we will not need middleware like HLA/RTI to manage the disparities. There was concern expressed that in a new field suffering from heterogeneity middleware solutions always seem useful and worth exploring, but that useful standards always seem to develop faster than useful middleware. There was no consensus that middleware solutions should not be explored, but there was some sentiment that high levels of funding for one approach (HLA/RTI) might be choking off work that could be done into other alternatives.

3. Delivery

The second major topic of discussion was the problem of Delivery. With content, we broached the subjects of how to build worlds rich with objects and how to describe those components; for delivery, we consider how to share those objects between many users in a reliable and scaleable fashion.

Difficulties in delivery currently stem directly from three characteristics of available network technology:

- network bandwidth;
- network latency and delay;
- network reliability.

For each of these issues, we illustrate the methods currently used to ameliorate the effects of an imperfect network

3.1 Network bandwidth

Many workshop participants felt that network

bandwidth limitations pose a primary problem for building large-scale multi-user VEs. One solution is upgrading the network, which is a slow but constant effort; however, we do not see any time in the foreseeable future that the network will allow point-to-point updates between thousands of users. It is therefore necessary, for both the short and long term, to conserve bandwidth wherever possible when designing such shared VEs.

Techniques intended to reduce the size of communications about system occurrences have parallels in other distributed-system communities, and so we do not address these at length. Methods that are applicable and popular in shared VEs include differential updates, function calls (for local modification, as opposed to communicating the modifications), and object versioning. The latter technique allows a client to connect to a VE and decide whether it already has an up-to-date version of a world object stored locally.

Interest management has been a focus of research throughout the last decade and still holds promise for significant additional reduction of communications volume. Participants in a world give some expression of what information they are not interested in, and at some point in the system topology those unwanted messages are culled. The participant can certainly discard unwanted messages locally, but this reduces only the local processing load and does not affect bandwidth consumption. A centralized server is considered to be the most convenient point for message culling, but there is some argument about whether the centralized approach is scaleable to very large systems.

Probably the simplest method for interest management, and one that has been used the most often to date, is regional interest. In such a system, a participant is only aware of (and therefore receiving update messages concerning) other participants within a certain distance, or within a predefined grid area of the world. In fact, some systems completely encapsulate these partition areas as isolated communities, and do not allow direct synchronous interaction between participants in separate areas. A finer-grained approach of this inter-user visibility culling is to annotate a virtual world with precomputed hierarchical visibility; in this way, a server which has access to these annotations and knows the position of two participants can determine if there is a line of sight between them. Then, rather than bombarding a participant with updates about all others within a community, a participant alone in a closet would receive no updates at all. Current possibilities for spatial culling include an even finer approach, which entails precomputing visibility in a non-binary fashion, to allow clients with varying network and processing capacity to receive varying frequency of updates. A related promising approach defines participant awareness in various media, and defines for each world object its

attenuating effect on those media. A media such as visual light might be blocked by a building, yielding an area for message culling, but unaffected by transparent windows.

Message management need not only be spatially-based; interest can also be related to the type of message. For instance, a client with no capacity to modify the world would be uninterested in messages relating to object locking and the like. Functional groups can be developed for a similar manner of interest management, such as a participant only wishing delivery of updates regarding members of a certain group of other participants. Membership in such groups is decided by the client, and the server relays messages appropriately. Functional group culling has an especially good outlook for systems that experience large user populations in a space with no visual occlusion.

3.2 Network delay

Network delay causes a host of concerns for consistent, distributed delivery of contents and actions in a shared virtual world. For the purposes of this report, we define network latency to be the minimum physically-possible signal propagation time. Similarly, we state network delay to be the total time between the initiation of an action in the sender and the arrival of that action at the application level at the intended recipient. While latency is a considered a constant and an effect of the speed of light, delay is a complex mix of network congestion, signal path, and processing loads throughout the trip of a network packet. This delay engenders inconsistency between clients regarding the definition of the world, in multiple ways. First, an action already experienced by some participants may not yet have arrived at all points in the world; and second, the ordering of occurrence of actions may vary between clients. Third, and causally related to the first, is that cooperation on a discrete action cannot be accomplished between delayed clients.

3.2.1 Reducing delay

Techniques being investigated to overcome the stated concerns fall into two categories: those that attempt to reduce the source of delay, and those that attempt to ameliorate its effects. The primary source of delay is usually network overload, so techniques in the former category attempt to reduce number and size of communications. For instance, standard Internet communications are usually sent via TCP, a reliable transport protocol that uses a series of bi-directional messages to guarantee delivery. Many update messages in virtual worlds do not require guaranteed delivery; for example, in the case of positional updates for avatars only the most recent update is used, and a constant stream of these updates means that a packet loss only results in a

minor and temporary loss of fidelity. For this subclass of messages, using a lighter-weight protocol such as UDP is acceptable. IP Multicast can reduce network usage as well by delivering a single message to multiple users while ensuring that only a single copy of a message travels any shared network connections.

While some attendees of this workshop are actively engaging IP Multicast in networked VE systems, the group is in agreement that due to its being an unreliable protocol and its limited availability to clients, short-term projects would not do well to rely on IP Multicast. The protocol is growing in popularity, and IETF working groups are currently investigating reliable versions, so it is expected that multicasting techniques will be integrated into all next-generation shared VE systems. It is notable that several workshop participants are developing high-level protocols that combine reliable, unreliable, and multicasting network protocols and make intelligent choices among them based on each message's purpose and contents.

All of the techniques mentioned as useful for reducing bandwidth consumption are of course equally applicable to reducing delay, since network congestion is a prime contributor to delay.

Another method to reduce aggregate delay in a virtual environment is to allow balancing. Migration of clients to nearer servers (near in this instance meaning "having less delay"), or migrating the servers themselves can significantly balance delay between clients. This does require some network monitoring; the position of the workshop participants is disappointment at the unwillingness of Internet service providers to provide network monitoring information as a public service.

3.2.2 Ameliorating effects of delay

The preceding techniques are intended to reduce delay, but no techniques can possibly eliminate the issue. A network latency of approximately 10ms is generated by crossing a time zone, so intercontinental traffic will always experience delay. Two promising methods discussed in the workshop were time warping and prediction.

Temporal warping uses knowledge of network delay between interacting participants to modify the rendering of objects. For instance, if user A throws a ball to user B, and the time of flight should be twice the network delay between them, then a standard wallclock system might show the ball appearing in mid-flight. With temporal warping, the ball would start at the proper point, but fly twice as quickly to the destination. There is a critical velocity problem, in that as these users near each other in the world, the acceleration factor is increased.

The other technique is prediction, which in fact works well in concert with the warping method aforementioned.

Certain predictions can be made about a participant's behavior, whether it is simple dead-reckoning based on their velocity in the world, or a complex heuristic based on recent actions. For example, if a participant winds up to throw a ball, the catcher's computer can predict the throw itself with some accuracy. Avatar motion prediction can also help with model fetching; rather than fetching pieces of the world as they are needed for rendering, motion prediction can be used to precache those parts of the world.

The issue of out-of-order packet arrival has been well-explored in standard CSCW systems, though no technique has yet received general acclamation. Networked VEs suffer from similar difficulties, and current approaches include use of time stamping, waiting for missing packets, ordering methods, causal ordering, and relaxed causal consistency.

Variable delay compounds the network delay problem, since prediction is much less effective with uncertain delivery times. Additionally, standard wallclock generating protocols such as Networked Time Protocol (NTP) are ineffective in such a setting, or even when delay is not equal in each direction.

3.3 Network reliability

Network reliability is the rate at which messages are lost, or misdirected to such degree that they are for all practical purposes lost. This has been discussed previously in the trade-off between reliable and unreliable protocols; UDP and IP Multicast would be acceptable for all applications with an ideal network where there was no packet loss.

In the face of an unreliable network, VE systems must be able to withstand the difficulties associated with lost packets. This is in fact quite similar to out-of-order packet handling, in that decisions must be made both about how long to wait for a missing packet, and what can be done when a packet is given up for lost. Again, the data consistency problem arises. Promising research to address this is being done on sliding consistency restraints, such that not all objects and behavior are treated alike, and causal relationships to determine which actions can be taken out of order (or lost) without ill effect. Though the problems of database consistency are not new, and are an active subject of research by the database community, workshop participants feel there may be additional complexities in consistency in a virtual simulation of an entire world.

Unreliability can also include network link failures, and a large-scale system should be able to survive such losses without a subsequent loss of presence for participants. For instance, if a single participant in a military simulation is lost to a machine or link failure, the system should be

dynamically reconfigurable such that an available host can begin to simulate that participant. This area was generally agreed to be an important area of exploration for dependable VE systems.

4. Architecture

The problems identified in this area are closely related to engineering high-quality software in general. A "good" is defined to mean

- good network topology
- composable (statically and dynamically)
- interoperable
- rapidly evolvable

These issues can be characterized as reuse of existing VEs to construct new ones. This problem is central to efficient and effective development of software in all application domains and so is not specific to networked VEs.

Very little discussion of these issues appears in the papers from the workshop, yet the problems cropped up consistently in the discussions so we have included the area in the summary. We believe this indicates an important problem area in which possible solutions really are not there yet.

4.1 Network topology

A shared VE that exhibits good network topology will scale to allow a very large number of concurrent users (thousands). It must allow models that cannot be stored on any one central server (terabytes). It must, as previously discussed, degrade gracefully in the face of hardware failure. It must provide adequate performance to users with widely differing communications capabilities.

There was a remarkable consensus among attendees that this cannot be achieved today except by a hybrid architecture of known techniques. "Central" servers are necessary to provide persistence to worlds. Clients are in profusion to allow users to probe other's worlds. Peer-to-peer communications are needed for aperiodic communications as well as for efficiently delivering streaming data like audio and video. Multicast must be used otherwise to cut network traffic to manageable levels.

4.2 Interoperability

The content and delivery problems (especially the semantics of content) must be solved before many of the architecture issues can be settled. For example, consider interoperability. We would call two VEs interoperable if we can leave one system and enter another without appearing to leave the unified "space." We should be able

to pick up objects controlled by one VE process and take them into another space, effectively transferring ownership and control of the item to another VE process.

A severe semantic problem keeps this from being reality currently. If I pick up a basketball in one VE and carry it into another, what tells the second system that this orange sphere should fall to the ground if dropped? That it should bounce rather than shatter when it hits the ground? What if the new system has no notion of gravity... what will the "basketball" sphere do then, and which of the two VE systems contains all this semantic information? Perhaps the basketball itself contains much of the "bounce don't shatter" behavioral definition, but even so such semantics can be environment dependent. Perhaps the basketball will "shatter don't bounce" if taken into an arctic environment and dropped.

Some progress has been made in interoperability in the field of MOOs and MUDs. The DARPA CAETI program specifically worked on cooperative data sharing and interoperation based on MOOs. One paper shows the interconnection language used in this effort.

4.3 Composability

A concrete scenario will best illustrate the notion of a flexibly composable VE software architecture. Suppose you have two existing VEs: a group meeting system, and a single-user ship walkthrough system. The group meeting VE allows several geographically-distributed participants to put on headmount gear and appear to be sitting around a table communicating with each other; perhaps there is some floor control to give structure to the meeting. The ship walkthrough VE allows a single user to explore the design of a new ship. Why can't we construct a group ship walkthrough by some combination of these two systems? Several users could then explore the ship design together and communicate among themselves their reactions. The functionality needed in the third system is contained in the union of the first two. However, we do not have the software development techniques yet to perform the combination in any automated (or semi-automated) way.

We identified a distinction between static composition and dynamic composition. Producing the combined system from the previous example would require methods for static composition (at software build time) as well as dynamic methods (inter-system data transfer at runtime). Dynamic composition is closely related to interoperability, though the workshop did identify some subtle differences.

One participant outlined a generic object framework for VE components that specifically allowed hierarchical

composition of worlds within worlds. This approach would facilitate system composition by providing a standard format (as opposed to being a middleware solution).

Related to shared VE composability is specification and analysis of collaboration protocols – the rules governing the interactions of the many users. In most current systems the collaboration protocol is buried in the code, spread across many modules, making it static during execution, and hard to modify during follow-on development. Research efforts are underway to extract these rules from an implementation, specifying them abstractly as data to be used by the processes in a network VE. One approach is to use a MUD/MOO for serving the protocol rules. This seems promising but leaves the issue of analyzing the protocols open, as protocols in a MOO are specified in a programming language. Other approaches use analyzable notation such as Petri nets for the protocol definition.

4.4 Rapid Evolution

The networked VE field suffers currently from an inability to rapidly take results from one research program and adopt them for further experimentation in another project. While this may be a common problem in all science, we lack even the most rudimentary standards with which to share results and accelerate new developments. VRML is providing some early relief in the area of content. Perhaps Java3D will do the same for formal (but not analyzable) semantics of contents. Several of the papers present specialized network protocols (VRTP, ISTP, DWTP) that make more efficient use of network resources and provide common system-level abstractions around which future projects may be organized.

Several other projects were mentioned as providing some early common abstractions around which new VEs can be organized. These include Living Worlds, Universal Avatars, and Open Community.

5. Bibliography

Those papers included in this section of the WET-ICE proceedings are of course directly relevant to the report included herein. Though our available space for publication was limited, we were very fortunate to have high-quality presentations from other workshop attendees. The following is a list of references to recent related work by those authors, to complete the collection in this section of the proceedings: