DVRMedia: A 3D Multiuser Distributed Virtual Environment

VICTOR M. LARIOS
Université de Technologie de Compiègne, Centre de Recherches de Royallieu BP 20529, 60205 Compiègne FRANCE, vlarios@hds.utc.fr, phone: 33 + 3.44.23.44.23 ext.4269, Fax: 33+3.44.23.44.77

Abstract
3D Virtual Environments support more natural and richer interfaces for representing information. These interfaces are directly related to Computer Supported Collaborative Work applications. In such interfaces, multiple users interact in real time at different geographical locations executing distributed applications over the network and dealing with problems such as concurrency, persistence, extensibility and communication performance. This paper presents DVRMedia, a 3D interface for multi-user virtual environments. The originality of this interface rely in its access via the Web, using a client-server distributed architecture. In addition, users can access databases and visualize their contents within the 3D environment. Our global objective with DVRMedia was to study the behavior of distributed virtual environments and propose new strategies to improve their performance. A solution for concurrency, persistence, and scaling problems in Virtual Environments has been implemented as well as a sample application. The use of a client-server-distributed architecture facilitates access via the Web but causes a lag in communications; this lag could be reduced using other computer paradigms, such as online algorithms.

Keywords: Distributed Virtual Reality, Virtual Environments, CSCW, Java and VRML interaction, 3D Web interfaces and Database Visualization.

1. Introduction

With the phenomenal growth of the World Wide Web, the big amount of information needs to be managed efficiently. In addition, research is conducted in virtual laboratories, in which scientists and engineers perform their work without regard to physical location. They interact with colleges, access instrumentation, share data computational resources, and access information at digital libraries. These Computer Supported Collaborative Work (CSCW) applications rely, in most cases, on 2D interfaces to browse information. Surveys show that over 50% of the designs and programming efforts in projects are devoted to the user interface portion [2].

51
Alternatively, Virtual Environments (VE) paradigms allow more natural representation and interaction with information behavior because, they are close to our real life interaction environment. A VE is a distributed application where multiple users are simultaneously present within a simulated 3D space, and where they can navigate interacting with 3D objects. Participants must be able to dynamically create, modify, and remove objects as well as enable others to do the same. In such environments, it is important that participants experience "acceptable" communication delays when they interact with other users.

With the introduction of the Virtual Modeling Language (VRML) [4] as a standard for the Web and the presence of more powerful personal computers, Web users may dispose of excellent tools for the visualization of geometrical 3D information. However, VRML does not yet provide support for multi-users VE's and on-line access to databases. However, by using VRML and Java together these problems can be solved.

DVRMedia has been realized within the TRANSDOC Project labeled by the French Industry Ministry. The aim of TRANSDOC project has been to provide intelligent vehicles to access information, and as a part of this project, this paper presents DVRMedia. The aim of this system is to solve two main problems: distribution in VEs and providing database access within a 3D interface. DVRMedia has an application that uses VEs as an interface for MPEG video sequences consulting, it is accessible from the Web and can be executed from heterogeneous platforms (PC, Mac, Sun, Silicon Graphics). This "virtual videotheque" is an example of an application that could be integrated to VEs on the Web. It may be used to provide a documentation center to organize and consult multimedia objects contents.

This paper is organized as follows: In section 2 the problems for a VE system are presented; Section 3 presents the architecture of DVRMedia; Section 4 introduces an overview of the application; Section 5 describes the system implementation and an application example; Section 6 discusses contributions and related works and finally, in Section 7 we conclude and discuss future works.

2. Motivations

2.1 Distribution

3D multi-user environments can support different CSCW applications [14]. From a distributed systems point of view, these concurrent applications address two interesting aspects in design: consistency and extensibility. The first one focuses on solving the mutual exclusion problem arising when concurrent users try to access the same object. The second aspect is related to the distributed architecture about which the 3D environment could be extended in a transparent way in the system. These two aspects may be adapted to hardware conditions like graphics performance and network bandwidth [11]. The success of 3D multi-user interfaces rests on their ability to provide an acceptable communication performance in the network where distributed strategies to implement consistency and extensibility in the system have an important weight.
2.2 Modeling the VE

A VE can be compared to a big database containing 3D object descriptions of three classifications: static, actor, and shared. Static objects contain the VE geometrical descriptions and they can not be changed by the users; for example, in a VE representing a city there are objects such as buildings, houses, streets, trees, etc., which users can not modify their internal parameters such as position, size or orientation. Actor objects are controlled by human beings or autonomous programs (agents), they are active processes in the system that can change dynamically their state (position, orientation, etc.). Shared objects have parameters defining their state, which can be modified by actors. As in a database, it is possible to insert, remove and modify object properties. In a multi-user environment, this database must be distributed ensuring data coherence and persistence.

A scene in VRML language is structured in a node hierarchy [3]. Within this concept, a world is a scene at the top of the hierarchy containing a set of sub-object nodes. Like in object oriented programming, an object could be composed of children objects. In DVRMedia, worlds are scenes containing three main branches at the same hierarchical level reserved for static, actors and shared objects respectively. The world server inserts actors and shared objects dynamically in the scene. For example, in an office world, static objects may conform the structure and furniture, shared objects are all the accessories such as a blackboard and finally, actors representing users. See fig. 1.

![Diagram of the object hierarchy in the system.]

Figure 1: The object hierarchy in the system.

3. SYSTEM ARCHITECTURE

3.1 Communications

This part of the system is composed of two services: the actor's management and the user communication tool. An actor is an active process in the system interacting in real time with the rest of the participants in the VE. Users are represented by an actor called "Avatar", displayed in 3D and carrying information like position, orientation, geometry description, animation scripts, and network address [3]. As an active process, the avatar is continuously
updating all the system about any change of his state in the VE (position, orientation, interaction with another object, etc.). Simultaneously, the user interface is updated with the present state of the other "actors" interacting in the system.

When a user interacts in a VE, the time needed to update all the system about the user's interactions is crucial. It has been shown that a delay in user action-result cycle of more than 250 ms will deter users from using the system [10]. The reason is because for delays greater than this, users do not have a good control of their avatars in the VE and also, it is not possible to have object interaction with precision and hence working with that interface becomes tedious. So this problem is concerned with maintaining efficiently communication and involves topics such as optimal protocols and distributed algorithms. Due to the inherent distributed nature of the system, the amount of information needed to update and keep the system running must be optimized using good distributed strategies and algorithms.

DVRMedia uses two communication protocols: The User Datagram Protocol (UDP) and the TCP/IP protocol. In the distributed system, UDP is employed to update in real time the state of actors and shared objects. In contrast, TCP/IP is used to guarantee the arrival of asynchronous informations that are important for the system.

The UDP protocol provides a mode of network communication whereby applications send packets of data called datagrams, to another one. A datagram is an independent, self-contained message sent over the network whose arrival, arrival time, and content are not guaranteed. Datagrams are useful when working in real time applications because the time delay between packets is close and in the case of lost packets, the introduced error is minimized when a newest ones arrives. To give an idea of the protocol importance, for N number of participants, there are N-1 updates whereas in the worst case there could be N²-N updates. Broadcasting updates to the whole system represents a complexity of O(N²) for network message passing and results in low system performance. With a protocol other than UDP and like TCP/IP, packets guaranteeing mechanisms can saturate easily the network.

TCP/IP protocol disposes of mechanisms to guarantee that the packets sent by an application will reach their recipients correctly. This enables the sender to resend a packet if corrupted data has been received or if the packet is lost. This is the reason why this protocol was not efficient for real time communications. Instead, it will be used to send asynchronous messages and information when arrival time is not important and it is important to ensure the content of the information sent. For example, when users achieve their login, we need to keep identification, password and initial parameters correctly as well as their geometrical description. Another use is in the chat zone where users can send text messages to groups in the VE. This user communication tool or chat run is a thread integrated to the user interface. It is possible to replace it and support more sophisticated tools like voice chat. In the case of voice chat, the protocol depends on the specific application.
3.2 Persistency

This is another important issue to consider since it is a distributed system, it may have more
than one server running in the same world. So, if a world server fails, there is always a backup
server preserving the world in its last state. In addition, running more than two world servers
at different sites in the network helps to distribute the user’s communication load. However,
managing a world with more than one server carries the problem of keeping coherence with
shared objects and can increase the time to update all users. An election algorithm between
servers is used to solve this problem [8]. This algorithm helps to keep coherence and to
remove concurrency problems when user tries to access shared objects at the same time.

3.3 Scaling

Scaling means how the VE should be extended [10]. In DVRMedia one or more servers can
control each world. A world represents only a fragment of the VE and a correct VE
partitioning can help to manage communications between users more efficiently. Another
problem, than communication is due to the fact that the user’s hardware can only handle a
limited number of objects in memory and CPU usage and large VEs can slow down the
interface performance. In addition, partitioning a VE helps designers to create each section of
the VE separately and dynamically assemble it later.

An analogy for the scaling problem is again related to a distributed partitioned database where
a set of distributed databases contains slices of the VE over the network. Some systems call
these slices "regions" where they are all interconnected to compose a continuous and vast VE.
These connections are done using special links to 3D objects representing the boundary
between regions. When user touches such objects, their connection changes to the
neighborhood region. This mechanism is known as a "portal"[13]. To summarize, portals are
links that allow users to change from region to region and in consequence, from world
servers. Figure 2 shows how persistence and scalability mechanisms are implemented.

![Figure 2: Portals and persistence mechanisms](image-url)
3.4 Overview of the user interface

Browsing and searching multimedia objects with traditional 2D interfaces could be a tedious operation when the amount of information is considerable (query result set returns a big chunk of related objects). Our original approach is the world "Videotheque" in DVRMedia, which is an application to browse 3D multimedia objects on the Web. This world is composed of bookshelves, 3D icons, database tools, and a set of virtual TVs. When a user access this world, a connection is established with a database via the database server. Then icons are

![Avatars](a)
![Label](b)
![Chat](c)
![Bookshelf](d)

Figure 3: an overview of the application. (a) The multi user interface, avatars and the Chat tool; (b) The "Videotheque" world with their components; (c) A virtual TV playing an MPEG contents and (c) The database interface for fast SQL queries.
created in function of the information returned by the database. The real object's memory size represents the icon's dimensions and the icon's color is related to object creation date; for example, the newer object is the closer its color is to red. As it gets older, its color changes gradually to blue or white. See figure 3(b).

3D Icons are shared items representing multimedia objects. They are placed in bookshelves, with each icon disposing of one touch sensor. Each bookshelf has a label indicating its general content. When a user touches an icon, all the information of the related object is shown in a special user interface window. This interface is also used for database queries, to show object information, and to execute shared actions like 'take' or 'leave' an object. Once a user has taken an icon, he can go to the virtual TV and play its contents, see figure 3(c).

3.5 Database access

The database access is done by a three-tier architecture supported by the Java's midleware JDBC API. The user must formulate a query in SQL language, then an intermediary interface provided by the JDBC API will take that query and reformulate it for an specific database manager. When the result set is obtained it is outputted in standard SQL and sent to the user application. The advantage of this architecture is the portability in applications. For our system, the database employed was a mSQL [5]; other databases may be used depending on the JDBC's loaded drivers. This database has been chosen because of its fast data set access and minimal host resource consumption. In the database tables, they are fields containing a complete description of multimedia objects and one reference to the related object in the network, see figure 3(d).

4. SYSTEM IMPLEMENTATION

Our system prototype has been developed in Java and VRML languages [4]. Java simplifies the applications development over the networks and VRML is used to model the interactive 3D environments. The integration of these two languages is done with the VRML's External Authoring Interface [3][4] defining a Java interface for external applets that communicates from an HTML Web browser. This integration results in independent client hardware platforms for the access and visualization of the 3D-user application.

From a Web page, users get connected via an applet. They must have already installed any VRML plug-in with support for the EAI interface in their Web navigator to enable Java interaction. When the client executes the Java applet, a network connection with a multi-user and a database server is established. The multi-user system manages the concurrent access and the participant's interaction in the VE. Some shared objects have references to more information stored in a database, for example a 3D icon representing a multimedia object. When the user decides to interact with such object in order to visualize its content, the database server establishes a connection with a database via the middleware JDBC returning an adapted content interpretation to be represented in the VE. Some times, these objects establish database access with a proximity sensor; for example when the user is close to an object containing a photograph, then more information including the image will be requested to a database and displayed in the VE, see Fig. 4.
A VE has been implemented allowing one or more people to visit our research center. This place is a 3D reconstruction of our real research center at the University of Technology of Compiègne, where users have access to one building. Inside the building, there are offices that could be visited, and the "videotheque" application can be accessed. The VE has been divided into four worlds: the buildings at the exterior of the research center, the interior of one building, one office, and the videotheque. Each different world represents at least one active server. A miniSQL database has been stocked with objects linked to MPEG video sequences distributed in other Web sites.

6. DISCUSSION AND CURRENT STATUS

Systems for multi-user VE's may be classified as: 1. Those focused on optimizing interaction between users, which are more oriented to CSCW goals [9]; and 2. When big distributed simulations with hundreds or thousands of users at different geographical sites are expected and where group research is focused on optimizing communication to ensure a good service quality. Our primary goal with VRMedia is to study the behavior of the distributed system and to propose new strategies where simulations with many users (as in the second category) are important. Because of this, all multi-user systems may consider these two aspects to be relevant to their distributed architecture.

Some of the most significant projects oriented to the CSCW using VE's can be characterized by their distributed approach used in system communications [11]. VEOS and MRToolkit are two systems using a peer to peer unicast strategy where each user's peer have N-1 users to update yielding a $O(N^2)$ complexity on each update. As the number of users increases each user's peer performs more work for updates, hence this strategy does not work efficiently for many users. NPSNET and Dive are systems using peer to peer systems employing multicast to update users, updates depend only on multicast groups allowing many users but in spite of that, these systems are only available for networks supporting peer to peer multicast.
messaging. Bricknet is a system using a client-server traditional approach where users send messages to a server who filters and propagates them to other servers containing the group of users to update. The last strategy scale infinitely in storage, processing and bandwidth depending on the kind of network but has inconsistent latency introduced for each update because of server's inter-communication. However, all these systems and their different approaches use specific platforms and network systems that are not accessible to the most part of users in the Web. DVRMedia uses the last model, which is more portable to the Web where distributed online algorithms, can optimize that extra-latency introduced by server’s inter-communication.

With the establishment of VRML as the standard in the Web, there has been an interest in providing access to multi-user systems to the Internet community considering low communication bandwidths via modem. The architecture for these systems has been inspired by Community Place and their Community Society [15]. Using the principles of a peer to peer multicast topology as in DIVE and adapting them to a client-server scheme, peers are substituted by servers called "auras" which manage a group of users. However, a big part of these systems accessed via the Web are implemented in most of cases with a centralized client-server architecture, and do not have an open structure to insert new functionality's to the interface.

On the topic of the use of VE’s as a tool to browse and visualize multimedia information in the Web, Flodar [6] is one system allowing visualization and analysis in real time of network traffic within a 3D interface. GOOVI-3D is another example of such a system [12] interacting with databases in networks and used for data warehousing. However, these systems do not have multi-user capabilities and their architecture doesn't consider many of the concurrency related problems that we are interested in.

7. CONCLUSIONS AND FUTURE WORK

Results obtained with DVRMedia show the advantage of using VE’s as an interface to represent and manipulate multimedia contents on the Web. Our main contributions are concentrated on the distributed architecture and strategies to improve the system communication performance using a client server approach. DVRMedia covers our expected goals in distribution but has some restrictions. Broadcasting to all the users with a simply UDP protocol is not enough to minimize the communication bandwidth and this issue affects the overall performance of the system. One of the interest of developing this system is to obtain a test platform to study different distributed algorithms that could be used in VE’s. Further works will be focused on optimizing the system in order to improve communications and quantify results. We are currently working on the implementation of online algorithms controlling servers to optimize client connections and service requests. These online algorithms use mobile agents to be aware of the network state and take a competitive decision for servers communications [1].

On the subject of the 3D interface, the development of this application with Java and VRML allows the participation of users via the Web from different hardware platforms. However, using the EAI interface to interact with both languages carries some limitations, interacting with a dynamic set of 3D-objects results in complex data structures. Other more direct
interface structures may simplify object representation and their manipulation dynamically. Further works may also be carried out to optimize this part between VRML and Java with another model of interaction such as the Java 3D API [7].

ACKNOWLEDGEMENTS

I would like to thank all people involved in this project, specially to professor Marc Bui for his helpful comments and direction on this work, to Pham Huy Huang for his important contribution in the system; and to the HEUDYSASIC at the UTC and LRIA at Paris 8 laboratories, for all the facilities that they disposed for this research project.

REFERENCES


AUTHOR

Victor Larios is a Ph.D. student in the Department of Computer Science at the University of Technology of Compiègne, France. Victor LARIOS is graduated from the Department of Electronics at the "Instituto Tecnologico y de Estudios Superiores de Occidente", Mexico, in 1996 and received the French diploma "DEA" (one year postgraduate study required for admission to a Ph.D. program) from the Control Systems Department at the University of Technology of Compiègne, France, in 1997.