Virtual Environments: An Agent-Based Approach

Jeffrey Coble **Diane J. Cook**

The University of Texas at Arlington
Department of Computer Science and Engineering
500 W. 1st St. Woolf Hall, Rm. 412
Arlington, TX 76019
{coble,cook}@cse.uta.edu

ABSTRACT

This paper describes research focused on the design of an agent-based virtual environment architecture that allows developers to create applications that can ease the burden of interfacing to complex systems. We are concerned with designing an architecture that supports applications that can integrate multiple, heterogeneous systems into one consistent user interface and to function as a collaborative tool so that concurrent users can share information, expertise, and resources. At the heart of our architecture is a Virtual Environment CORBA Facility. This facility provides an improved method of distribution and most importantly, explicitly supports the integration of information from external systems and provides a method for interacting with those systems by using the virtual environment as a user interface. What we present in this paper is a method from representing and distributing a virtual environment so that it can support advanced agent functionality. The nature of that functionality is ultimately left to the discretion of virtual environment developers. We have developed a granular, object-oriented representation for virtual environments that allows developers to incorporate agent technology into a web-based distributed virtual environment.

Introduction

This paper describes research focused on the design of a virtual environment architecture that allows developers to create applications that can ease the burden of interfacing to complex systems. We are concerned with designing an architecture that supports applications that can integrate multiple, heterogeneous systems into one consistent user interface and to function as a collaborative tool so that concurrent users can share information, expertise, and resources.

Based on our evaluation [3] of current commercial virtual environment architectures and our subsequent attempts to use them to develop advanced applications, we have concluded that a virtual environment architecture must allow developers to create applications that are network-based, distributed, platform-independent, multi-user, and secure. The architecture must allow applications to seamlessly integrate a variety of multimedia types and should be able to function as a repository for information. The ability to modify and manage the state of objects within the virtual environment is crucial. A virtual environment must allow the users and external systems to change the state of objects, share that state with other users, and the state must persist. The architecture must be flexible enough to allow new programmatic capabilities to be created without the need to re-distribute updated versions of the infrastructure software.

Our previous work [3, 4] addressed many of these issues with a web-based architecture that was designed to allow developers to create persistent virtual environment applications. Our previous work focused on such issues as creating the structure needed to represent virtual environments so that they could support persistent state and could be augmented with other application specific services. What we present in this paper is an advance to our architecture that consists of the definition of a CORBA (Common Object Request Broker Architecture) Facility designed to include the aforementioned attributes but provides an improved method of distribution and most importantly explicitly supports the integration of information from external systems and provides a method for interacting with those systems by using the virtual environment as a user interface.

Why Virtual Environments?

The recent standardization of VRML (Virtual Reality Modeling Language), combined with the rapidly increasing power of the personal computer, is allowing the technology of modeling and simulation to move from the domain of engineering applications, requiring expensive computer systems, to everyday desktop use. VRML is designed for web-based distribution and has made possible the development of distributed virtual environments. The term "distributed virtual environment" may invoke comparisons to the many years of substantial research and development in large scale distributed simulation systems. For the

This work was supported by NIMA under grant NMA202-97-1-1022

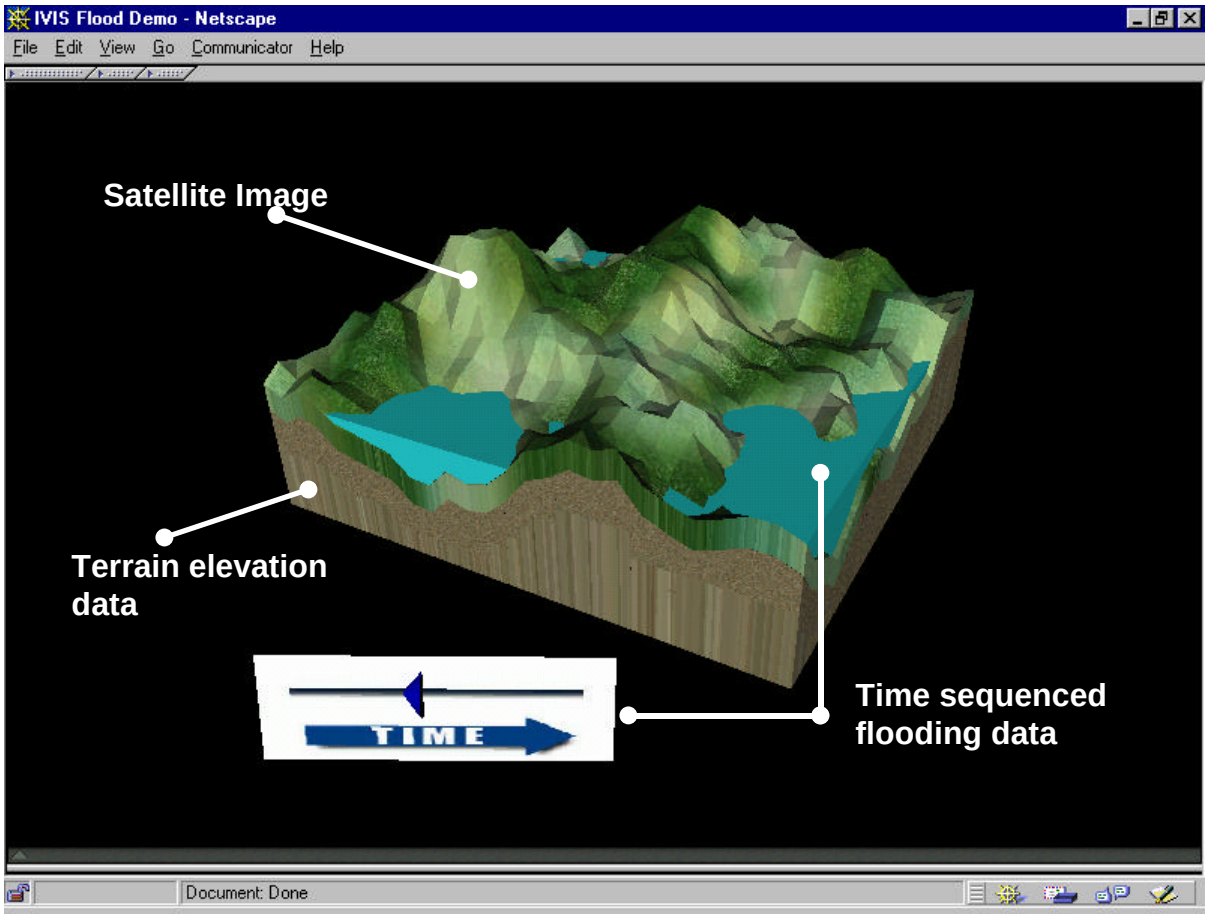


Figure 1: Screen capture from a Virtual Reality Modeling Language representation of the time sequenced progression of flooding over a land area. Our virtual environment CORBA Facility will allow this application to bring the satellite image, the terrain elevation data, and the elapsed flooding data from separate data sources.

purposes of this paper, our discussion will be restricted to the recent work involving *web-based* distributed virtual environments, designed to run on standard desktop computers without concern for the factors surrounding reality-based simulations.

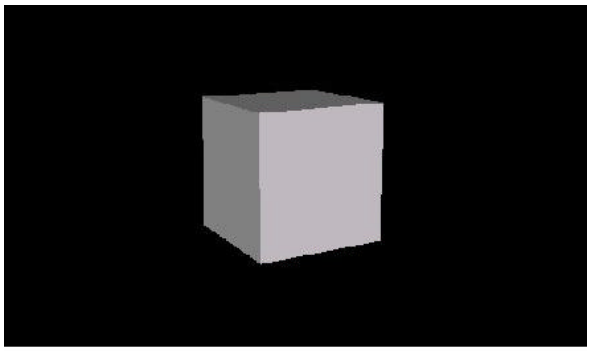
Humans use multiple senses to process information and to perceive their environment. Just like any data conduit, each sense has a limited capacity to provide information with respect to time. Furthermore, some types of information cannot be adequately conveyed in some modes of communication (e.g., try describing the taste of a banana). By combining the power of multiple senses, humans are able to asynchronously perceive and synthesize more information of varied types in a shorter timeframe. Virtual environments aim to provide a much more intuitive interface to complexly organized information by capitalizing on these human perceptual processes and intuitive human experiences.

Current commercial distributed virtual environments are web-based, computer-generated, three dimensional, interactive visualizations, which unlike simulations, need only suggest a real or imagined space [1, 10]. They provide a very compelling, multi-user environment in which information, resources, conversations, and applications can be accessed by distributed participants.

Many of these virtual environment architectures employ a VRML representation of their content and provide a forum for multiple distributed users to congregate, communicate, and access the same type of information generally associated with web pages. Our research expands upon these valuable capabilities by providing an architecture that can support the use of multi-user distributed virtual environments as advanced interfaces to distributed heterogenous computer systems and databases.

Although our architecture is designed to provide many features that we feel are necessary to support advanced

applications, the focus of this paper will be on the mechanisms by which our architecture allows developers to create virtual environments that can serve as a user interface to multiple, heterogenous computer systems. By supporting a mechanism for this type of integration, we allow developers to create virtual environment applications that can allow users to interact with multiple computer systems through one consistent interface. This has clear value in its ability to simplify processes that require the use of these systems via their own disparate user interface mechanisms. By combining the features of our architecture that support collaboration and persistence into a CORBA Facility, our architecture can be used to create an interface to a collaborative environment in which users can congregate to cooperatively solve problems and share expertise about complex, dynamic multi-modal information displays that are representative of volatile data, residing in distributed, heterogenous real-time systems [7, 8].



```
Transform {
  children
    Shape {
      appearance Appearance {
        material Material {
        }
        texture NULL
        textureTransform NULL
      }
      geometry Box {
      }
    }
    translation -0.123259 0.0819294
-4.5
}
```

Figure 2. Illustration of the VRML cube created by the accompanying VRML syntax.

Application

We are currently applying our virtual environment research to improving the productivity of intelligence analysts at the National Imagery and Mapping Agency (NIMA). In order to provide recommendations to policy makers, NIMA analysts must analyze a wealth of complex information, such as multi-spectral satellite imagery, which resides in mutliple, distributed systems. In many cases it is also necessary to examine both current and historical data. The analysts are faced with tremendous amounts of information, coming from many different sources in many different forms. It is often the case that analysts are attempting to identify a trend or event based on temporally ordered sequences of information. For example, a month long view of satellite imagery, RF (Radio Frequency), and SAR (Synthetic Aperture Radar) may indicate that a new military base is being constructed in a strategic region of Iraq.

Because of the complexity associated with analyzing any particular type of information, analysts often become experts in a particular medium. For example, one analyst might be an expert at analyzing satellite photos while another is an expert at interpreting the RF fequency spread. The decomposition of expertise may be even more fine grained than this example. Given this situation, it is often necessary for multiple, distributed analysts to collaborate in order to pool their expertise to ascertain what is happening based on the wealth of intelligence information. Analysts are often working on time critical problems, therefore the more proficient they are at accomplishing their task, the more accurate and useful their interpretations will be.

Figure 1 illustrates an example that we are developing in this domain. This is a virtual environment designed to allow the analysts to analyze the flooding at a particular land area over time. Using our architecture, each visual level of information in the example can be generated from a distinct database or computer system. Furthermore, the

change in the data that resides in these distributed repositories can be reflected in real-time in the visual representation of the virtual environment. The users would be able to move the time slide bar and generate a query to the external database that contains the time sequenced flooding data. The results of the query are visually reflected by the rise or fall of the water level in the virtual environment.

VRML has many inherent strengths that make it an excellent choice for 3D interactive visualizations [5, 6]. It runs within a web browser, supports JavaScript, provides a sophisticated API accessible from external JAVA applets/applications, and is hierarchical by design. The current VRML standard already supports virtually any type of graphical multimedia and can seamlessly integrate sound, color, spatial relationships, temporal relationships,

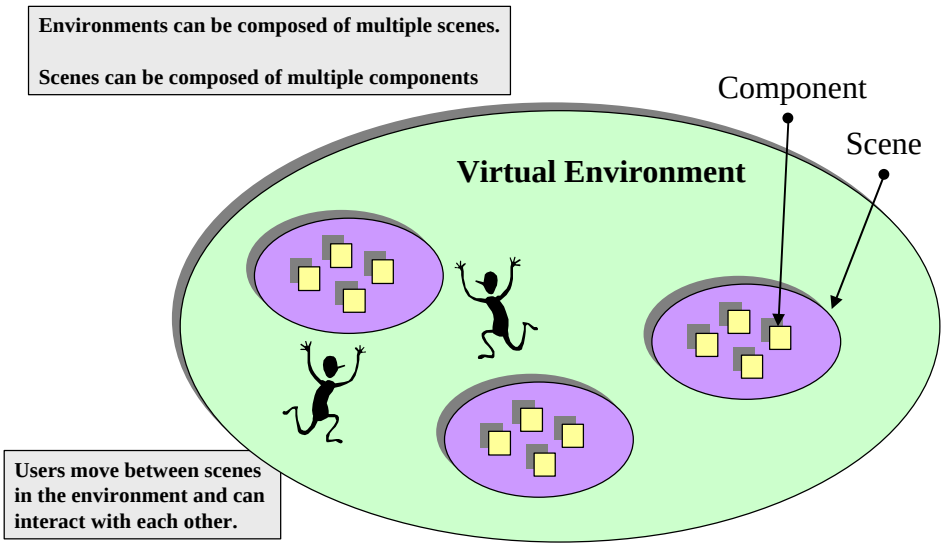


Figure 3. Virtual Environment Hierarchy.

What we have just described is useful in itself. However, since we are representing this information in the context of a multi-user distributed virtual environment, our architecture is designed to expand on the innate collaborative capabilities. Our architecture will allow multiple users to coexist in the virtual environment and interact with the information display. The method by which the components of the virtual environment are represented allows the information to reflect changes in the underlying data and for those changes to be shared, in real time, with the distributed participants. As we will explain in later sections, the information that a user sees is represented by a programmatic structure that is far more robust than that of a modeling language. Our Virtual Environment CORBA Facility provides the means by which the state of the visualization can be changed, shared, and stored to reflect real-time data from external systems or user interaction.

Virtual Reality Modeling Language

Since most web-based virtual environment architectures, including ours, use VRML to represent information to the user, it is relevant to examine its strengths and weaknesses.

and existing 2D widgets (charts and graphs).

The web-based nature of VRML naturally lends itself to Internet-based collaboration methods. These capabilities advance our effort to 'fuse' multiple, heterogeneous data sources and systems into one consistent interface.

Although VRML is expressive, by itself it lacks the power and the structure to support advanced virtual environment applications. VRML models are typically constructed offline and accessed via a web server. A VRML visualization is represented statically as a text file, which contains a description of the 3D visualization (Figure 2). This description is interpreted at the local web client, with a VRML rendering plug-in, and displayed to the user. Although this structure has produced the ability to develop '3D-web page' type content, it is insufficient to create a robust, dynamic, distributed virtual environment application [3, 4]. In short, VRML is a web-based modeling language, not a programming language and therefore lacks any notion of state or the ability to sufficiently augment the inherent functionality.

The Multi-Agent Architecture for Virtual Environments

To address the aforementioned issues, we introduced a Multi-agent Architecture for Virtual Environments (MAVE) in [4]. To understand the motivation behind MAVE, it is important to discuss the hierarchical nature of a virtual environment. Figure 3 illustrates this commonly accepted hierarchy. A virtual environment is composed of scenes and each scene is composed of objects. The Environment Scene (VES) (figure 4). The VEC and VES represent an object-oriented physical granularity that mimics the logical decomposition of the respective elements of the virtual environment. We define a VEC as the lowest level element in a virtual environment that can stand alone as a useful entity. This is intentionally a flexible definition but one that, in practice, is easy to follow. Since each VEC can correspond to an instance in an object-oriented database, it makes sense to maintain a

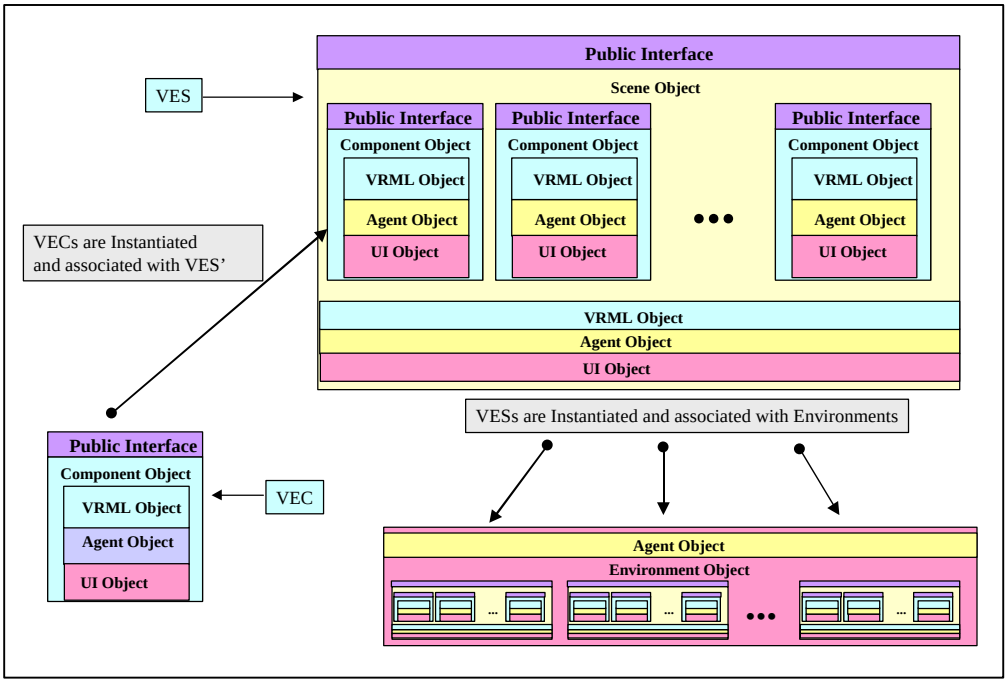


Figure 4. Tier 1 architecture. Illustration of the object oriented structure of the virtual environment.

acceptance of this hierarchy as the defacto standard is partially derived from the hierarchical nature of VRML but also from the need to decompose the visualization into manageable pieces. Attempting to render the entire environment as one entity would overload the processing power of most personal computers. This hierarchy lends itself to our architecture and the type of componentization that we are attempting to achieve.

The cornerstone of MAVE is a novel 'end-to-end' object oriented design. Our research focuses on architectures at two different tiers. Figure 4 depicts an object-level architecture that is crucial to creating the powerful software needed to provide the functionality of advanced virtual environments. Figure 5 depicts the Virtual Environment CORBA Facility that is designed to provide the infrastructure necessary to support the integration of heterogeneous systems, state maintenance, scalability, collaboration, and a distribution mechanism.

The first tier of the architecture includes the notion of a Virtual Environment Component (VEC) and a Virtual level of granularity that has some significant cognitive value. A VES has all of the same elements of a VEC with the added association of a series of VECs.

Use Cycle

Before we discuss the details of our architecture, it is necessary to describe the use cycle from development to distribution. The object-oriented structure of our architecture allows developers to either re-use a set of VECs and VESs, to create their own virtual environment, or any combination of new and old. The VRML objects embedded in the VECs and VESs are a programmatic representation of the VRML text that would normally be transmitted to a user who is viewing a standard VRML visualization. We are using JAVA objects for this purpose since they are readily usable in the context of a web browser. At a minimum, a VES can simply include a reference to a series of VECs. Once a series of VESs are

created, they are associated with an environment object. This complete structure constitutes a virtual environment.

What we have just described constitutes the development of a collection of objects that are used to create the user interface of the virtual environment. We now must consider the need to interface with external systems and create the elements to accomplish this. As illustrated in figure 5, our CORBA Facility includes a set of Object Managers. The purpose of these Object Managers is to link the state of active VECs/VESs to external systems via the CORBA interface. This will be discussed in greater detail in the following sections.

In the context of our example for NIMA, we designed and constructed these elements of the virtual environment so that they provide the appropriate functionality for a specific task. We designed the VECs and VESs so that they provide the desired representation of the information and the necessary human interface functionality and subsequent linkage to external data sources and computer systems.

The collection of objects that comprise the virtual environment is stored in an object-oriented database, which is an element of the Virtual Environment Facility. When a user requests access to the virtual environment, this object structure is sent to the client machine. Once the object structure is resident on the client, the VRML objects communicate with the rendering plug-in via the External Authoring Interface (EAI), which is part of the VRML standard. The EAI allows external applications to create or modify the VRML visualization. The VRML objects create the visualization by using the EAI. It is up to the developers of the VECs and VESs to determine how they will organize the visualization. That is an application implementation issue rather than a concern of the architecture. The server side architecture will be discussed in detail in the following sections, but it is worth noting now that this architecture explicitly supports multi-user collaboration capabilities but leaves the implementation up to the developer.

Object-Oriented Design

The hierarchical structure of the virtual environment and the natural hierarchical structure of VRML heavily influence our choice of an object-oriented design. Applying object oriented properties to our architecture increases the capabilities while maintaining maximum flexibility. For instance, exploiting inheritance can easily provide a mechanism for integrating a multi-level security model that allows a component to default to the security model of its parent scene or to overload the security methods to implement its own model. This is true at the scene level as well. Scenes can have a security model of their own or default to that of the environment. For example, any employee can enter their company's virtual environment, any employee of the Engineering department can enter Building 3 (VES), but only the manager can change the contents of the whiteboard (VEC) on the wall.

VRML Object. At the core of each VEC and VES is a VRML object. This object maintains a programmatic representation of the VRML visualization. Bearing in mind that the visualization can change dynamically from user interaction or a real-time data feed, the need for this representation is clear. The VEC's internal programmatic representation of the VRML communicates with the rendering software to display the visualization to the user. In response to a stimulus that would change the VRML visualization, the programmatic structure is changed first and then the visualization is updated. There is always a one-to-one mapping from the programmatic representation to the visualization.

If a VEC has been designed to be persistent, it will communicate with the Session Manager Object in the CORBA Facility (Tier 2) which, in turn, will update the state in the Object Database and propagate the new state out to the concurrent users of the virtual environment. When a new user enters the virtual environment, they will download the *new* virtual environment rather than the original version. This is a significant improvement over the current state of the art in virtual environments since most have no mechanism of maintaining state. In other systems, users only receive a copy of the original virtual environment every time they enter. The hierarchical representation of the virtual environment provides the ability to maintain persistence at a fine-grained level. Not only is this convenient, but it is also efficient. Changes to the state of a small object do not require large amounts of information to be transmitted as it would if the environment was one physical entity.

The VRML object is the only required element of the VEC/VES. The others are either dependent upon the functionality of the VEC/VES or required only for advance virtual environment functionality.

User Interface Object. The User-Interface object would be used when more traditional methods of input or output are desired, such as dialog boxes or buttons.

Agent Object. The Agent object will not be discussed in detail in this paper. It is our belief that inserting agent objects into each level of the hierarchy is crucial for the effective use of intelligent agents. By doing so, we are creating a relationship between the VRML, Agent, and User Interface Objects. This would be very important for providing a mechanism for Intelligent Agents to obtain component level information with regard to the visual content and the user model at each level. This structure will allow intelligent agents to provide better services to users. The public interface at each level is a provision for sharing VEC/VES services with other components. This would normally be an IDL (Interface Definition Language) that is consistent across the virtual environment.

Component Object. Depending on the application, the Component Object may be complex. This object will be

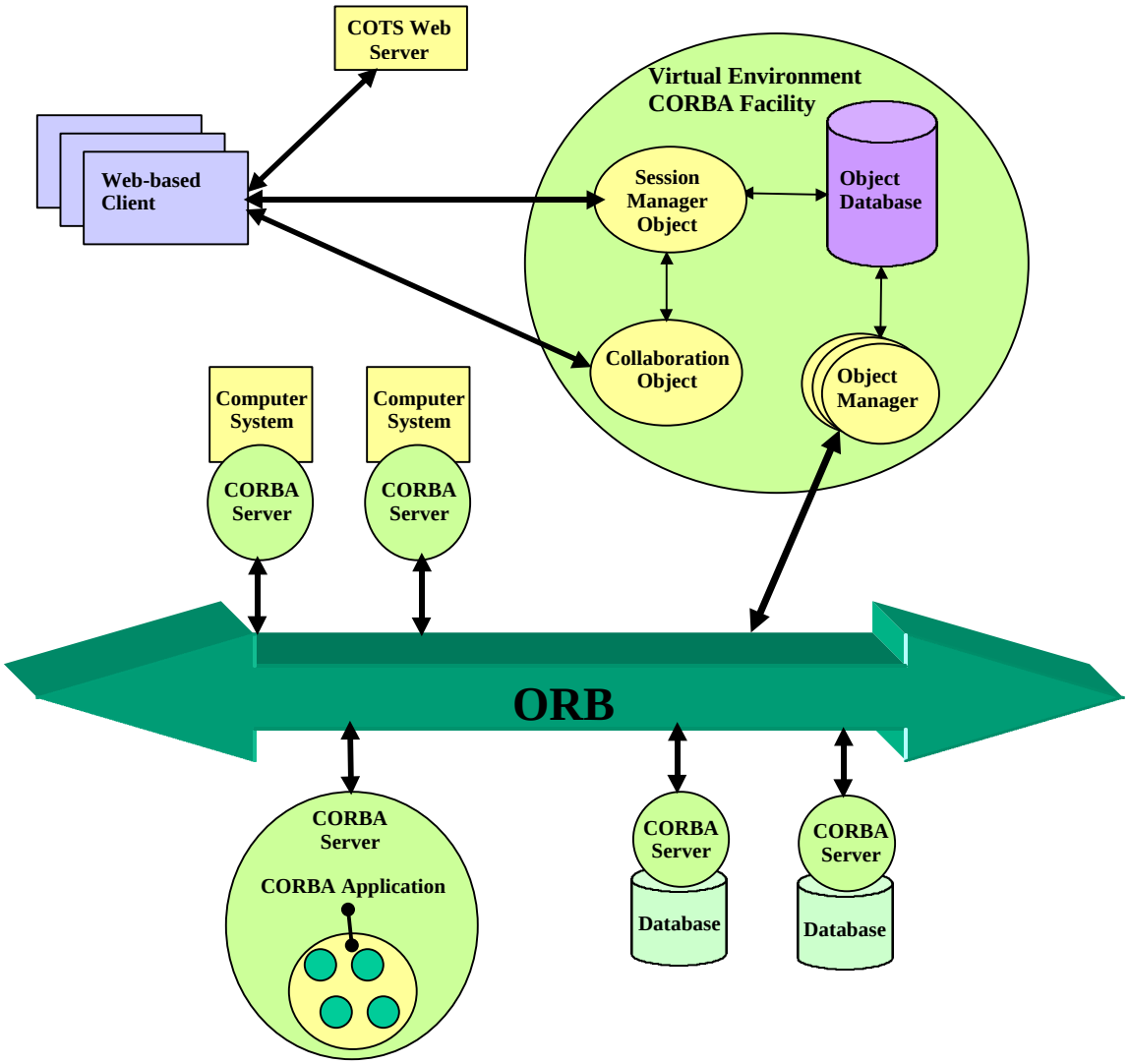


Figure 5. Virtual Environment CORBA Facility. Designed to provide integration of distributed systems into a consistent interface, persistent components, and concurrent access by multiple users.

the top level object in the VEC and will have the responsibility for instantiating the other objects with their initial state when the first user activates the virtual environment. When a user's request for the virtual environment is received by the Session Manager Object, a check is made to determine whether or not the virtual environment is already active. If it is not, the Session Manager Object invokes the Environment Object (top level), which in turn instantiates the VESs, each of which in turn instantiate all of their associated VECs. At this point, each Component Object, which is the top level object in a VEC would instantiate all of its elements (VRML, Agent, UI) and also must instantiate an Object Manager if the VEC is designed to interface to external systems. The implications of this are summarized as:

- Developers must create the components illustrated in figure 4 and
- Developers must create any corresponding Object Managers, illustrated in figure 5.
- Developers must create their object hierarchy so that all of the appropriate objects are instantiated to guarantee the linkage from external systems all the way down to the end user.

These criteria help us to illustrate the value of defining much of the functionality of this architecture as a CORBA Facility. The following section will discuss this in detail.

Corba Facility For Virtual Environments

Figure 5 outlines what we refer to as the Tier 2 architecture. The CORBA Facility in this figure is the server-level element of our architecture while the remaining components illustrate the interaction among the various pieces of an application.

Corba Overview

CORBA technology is gaining wide acceptance as a standard for developing distributed computer applications and for providing remote access to object repositories [2]. The CORBA standard includes many important features that make it a clear choice for our architecture. Some of the primary features are:

- Designed to provide TCP/IP access to CORBA resources and therefore supports access via the Internet.
- Language independence allows clients running on virtually any platform to access servers in another language by using native operations. The CORBA IDL (Interface Definition Language) specification has been mapped to many common languages, including JAVA, C++, and C, which allows transparent access between applications written in these languages.
- Supports reconfigurable applications by allowing developers to create modular components that can be added or removed without disrupting the overall application.
- Provides a consistent method of access to any CORBA server.
- Each vendor must support communication with other vendor products via the IIOP (Internet Inter-ORB Protocol).

Each vendor's implementation of the CORBA standard is called an ORB (Object Request Broker). The ORB provides the basic functionality defined in the standard. Furthermore, each vendor has implemented some set of the CORBA Services, which are utilities useful for developing distributed applications. CORBA Facilities are designed to provide higher level services. Some facilities under development are geo-spatial data processing and system management.

Facility Components

The purpose of our Virtual Environment Facility is to provide a framework for developing virtual environment applications that can take advantage of the power of CORBA while allowing developers to be abstracted away from *some* of the implementation issues inherent to CORBA. Our Virtual Environment CORBA Facility includes several components. They are:

Web Server. This component is not actually part of the facility. The initial connection to the virtual environment is established by accessing a web page with a JAVA enabled browser. A JAVA applet is sent to the client's browser, which opens a CORBA connection to the Session Manager Object within the Virtual Environment Facility. Developers will only need to be provided with the IDL (Interface Definition Language) specifications for the Session Manager Object in order to establish this connection.

Object Database. An object-oriented database is a natural choice for our architecture due to the efficiency issues related to storing objects in a standard relational database. An active virtual environment is stored in the Object Database. Changes in state that occur from interaction from users or external systems are stored in the database.

Session Manager Object. This is the primary object within the Virtual Environment Facility. It has the basic responsibility for instantiating virtual environments on first access, tracking users within the virtual environment, propagating the state of shared resources, and managing concurrency.

Object Managers: The Object Managers create the link from external systems to the state of components in an active virtual environment. Changes in external systems cause the appropriate Object Manager to update the state of a corresponding VEC/VES, which in turn causes an event to be triggered in the Object Database. The event is sent to the Session Manager Object, which responds by propagating the new state out to the concurrent users. Alternatively, when the Session Manager object updates the state of a component in the Object Database in response to user interaction, an Object Manager can be triggered to perform some action in an external system.

Collaboration Object. We only loosely define the Collaboration Object here as it can function to provide services as basic as a text chat or as advanced as incorporating streaming video teleconferencing into the virtual environment.

By combining these components with the object-oriented structure of the virtual environment and the power of CORBA to support the distributed nature of the system integration, we have all of the elements necessary to create a powerful user interface to multiple, heterogeneous systems.

Conclusions

The broader scope of our work has been to support advanced applications within a virtual environment. One of the most desired features of advanced virtual

environment applications is the ability to collaborate with a group of distributed users, while working with complex information from varied systems. Our Virtual Environment CORBA Facility, combined with our novel object-oriented representation of a virtual environment has been successful in providing a method for accomplishing this task.

Our experience with working in this domain has made us all too familiar with the inherent complexity of developing distributed applications, even with CORBA. A well executed CORBA Facility will provide a level of abstraction from these difficulties and will allow developers to exploit the power of distributed virtual environments as user interfaces to complex systems.

We are currently developing applications, for the National Imagery and Mapping Agency, as described in a previous section, with the goal of increasing productivity for analysts who must collaborate and cooperate to solve complex problems. We have demonstrated the feasibility of the first tier of our architecture by constructing a single user virtual environment entirely with the object-oriented structure illustrated in figure 4. We have partially validated the second tier architecture (figure 5) by conducting some simple feasibility experiments designed to illustrate the linkage of CORBA objects to the state of objects within the virtual environment structure. A full scale application, employing both tiers of the architecture with external linkage, state maintenance, and multi-user access is currently under construction.

Future Work

Our immediate future work will focus on validating the second tier of our architecture. We believe there are three areas of long term interest that will further our ability to support cooperative work in virtual environments.

Development Suites

In order to allow developers to take full advantage of our Virtual Environment Facility, we will investigate the development of a tool suite for developing virtual environment applications. The three most complex components to develop are the VRML Object, the Component Object, and the Object Managers. We will focus effort on creating automated methods to assist developers in creating these low level components of the virtual environment and linking their state to external systems. It is unlikely that all complexity can be abstracted from development. Applications of this technology are not generic and in fact will be widely varied. We anticipate spending considerable effort in this area.

Portable Components

By developing standards for component interaction and the necessary security protocols, it would be possible for users to develop reusable, portable components. A user could carry these components along with them to foreign virtual environments to facilitate specific needs of a user or to allow for the immediate expansion of a virtual environment. For example, a user may develop a virtual whiteboard that can be carried with the user as she enters new virtual environments. This would allow the user to share custom information with other users in the environment and allow those other users to interact with the information as prescribed by the component developer. This technology would allow for an impromptu meeting or discussion, with all the necessary presentation materials readily available.

Intelligent Agents

The representation of the virtual environment is designed around the notion of Intelligent Agents. This was due to the inherent complexity of the domains that we felt would benefit the most from advanced virtual environment applications. Although we feel the virtual environment concept provides a great deal of value, we believe that explicit support of Intelligent Agents will vastly improve the usability of these applications. Some such services that Intelligent Agents may provide are: assistance accessing features of a particular component, navigation assistance, locating other users with similar interests or expertise, or seamlessly establishing various modes of communication with other users (text/voice chat). The list of agent services is only limited by the imagination of the developer and is naturally associated with the particular virtual environment application. Our future work will focus on the development of an agent-based approach to providing user-specific views of classified information within a multi-user virtual environment where the users are of varied levels of security clearance.

References

1. Braham, Robert and Comerford, Richard, *Sharing Virtual Worlds*, IEEE Spectrum, pp.18-25, March 1997.
2. Baker, S., *CORBA Distributed Objects Using Orbix*, ACM Press, NY, NY, 1997.
3. Coble, J., *The Evaluation Of Virtual Environment Architectures And The Development Of Mave: A Multi-Agent Architecture for Virtual Environments*, University of Texas at Arlington Master's Thesis, December 1997.
4. Coble, J., Harbison, K., *MAVE: A Multi-agent Architecture for Virtual Environments*, to appear in the Proceedings of 11th International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems, June 1998, Benicàssim, Castellón, Spain.

5. Couch, J, et al, *Late Night VRML 2.0 with Java*, Emeryville, CA, Macmillan Computer Publishing Co, 1997.
6. Hartman, J and Wernecke, J, *The VRML 2.0 Handbook: Building Moving Worlds on the Web*, Reading, MA, Addison-Wesley Publishing Co., 1996.
7. Leigh, J. and A.E. Johnson, *Supporting Transcontinental Collaborative Work in Persistent Virtual Environments*, University of Illinois at Chicago. 1996.
8. Leigh, J., Johnson, A., DeFanti, T., *Issues in the Design of a Flexible Distributed Architecture for Supporting Persistence and Interoperability in Collaborative Virtual Environments*,. To appear in the proceedings of Supercomputing '97 San Jose, California, Nov 15-21, 1997.
9. Maes, P., *Agents that Reduce Work and Information Overload*, Communications of the ACM, Vol. 37, No.7,pp. 31-40, 146, ACM Press, July 1994.
10. Rockwell, Robert, *An Infrastructure for Social Software*, IEEE Spectrum, pp. 26-31, March 1997.
11. Russell, Stuart and Norvig, Peter, *Artificial Intelligence: A Modern Approach*, Englewood Cliffs, NJ: Prentice Hall Inc, 1995