Integration of Object-Oriented Databases with VRML in Virtual Environments

Damla Turgut, Nevin Aydin, Ramez Elmasri and Begumhan Turgut

Department of Computer Science and Engineering
The University of Texas at Arlington
P.O. Box 19015
Arlington, TX 76019-0015
E-mail: {turgut,aydin,elmasri,bturgut}@cse.uta.edu

Abstract. Virtual Reality Modeling Language (VRML) is widely used to represent, create, and display virtual reality objects and their environment. Some VRML applications require concurrent interaction by multiple users in a real-time distributed fashion. Such applications need a method for users to share and update the VRML objects in real-time. To allow concurrent shared real-time access, our approach is to store the VRML objects in an object-oriented database system (ObjectStore). In this paper, we present an architecture that allows multiple users to interact in a non-trivial way in such a shared VRML environment. We outline how the VRML world can be saved in ObjectStore.

Keywords: Object-Oriented Databases, Virtual Environment, VRML.

1 Introduction

Complex virtual reality applications are increasingly using VRML (Virtual Reality Modeling Language) to model the objects and their interactions [9, 4]. When dealing with VRML worlds and objects we often find the need to retrieve, manipulate and store the states of a VRML object as it changes over time. This is called VRML persistence and is gaining importance among VRML applications. As VRML applications become more complicated and process real-time data, the need for adequate persistence capabilities increases. VRML and object-oriented databases are relatively new fields rapidly gaining importance and popularity. However, developers have not yet produced a comprehensive solution to VRML persistence.

The rest of the paper is organized as follows. We describe the traditional MAVE (Multi-agent Architecture for Virtual Persistence) architecture [1], introduce concepts from VRML and give an application where VRML persistence is useful in section 2. Storing VRML objects in an ODBS (Object Database System) has the advantages of sharing the VRML model among multiple real-time applications. In addition, other ODBS functionality, such as query languages and views, may be utilized. Section 3 gives an overview of PSE (Persistent Storage Engine) [7] for the ObjectStore [6, 8] ODBS, which was used in our system to provide persistence. Conclusions are drawn in section 4 by showing how to achieve VRML persistence within the MAVE architecture and future work.

2 VRML and MAVE

VRML (Virtual Reality Modeling Language) [4] integrates 3D graphics, 2D graphics, text and multimedia into a coherent model. VRML allows us to use a computer generated 3-dimensional virtual environment which provides an intuitive interface to complex information. In VRML, a scene graph is a group of objects describing the structure of the virtual world that is being created. The primitive object types represent boxes, cones, cylinders, and spheres. The scene graph will be used to develop an EER (Extended Entity Relationship) model [3], which we will use to design the database for storing VRML objects in ObjectStore [6, 8]. A VRML node is analogous to a structure definition in a high-level language.

The objective of MAVE (Multi-agent Architecture for Virtual Persistence) [1] is to develop an agent-based architecture to support intelligent, reusable, distributed virtual worlds. MAVE is a two-tier architecture. The first tier is an object-oriented physical representation of the virtual environment that is designed to mimic the logical decomposition of the virtual world. The second tier is designed to support the needs for persistence, real-time interfaces to external data sources, distribution, and collaboration. The object level representation for MAVE is shown in Figure 1.

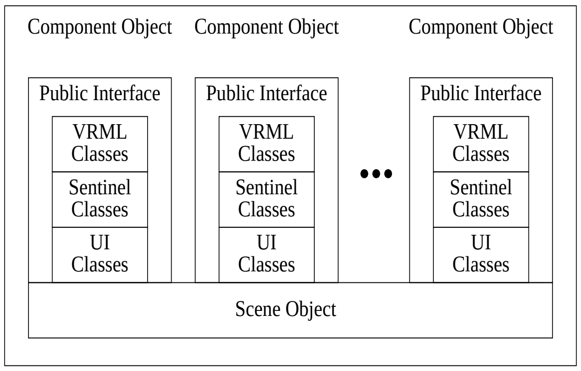

Fig. 1. Object-level Architecture

The virtual environment provides a framework in which distributed users can collaborate and share resources, including a variety of multimedia types. In the object level architecture, VRML is used as one element of the overall virtual environment. To understand MAVE, it becomes important to discuss the hierarchical nature of a virtual environment. A virtual environment is composed of scenes and each scene is composed of objects. In MAVE, the VEC (Virtual Environment Component) architecture maintains a physical granularity that mimics the logical decomposition of the respective elements of the virtual environment. VEC is the lowest level element in a virtual environment that can stand alone as a useful entity. Each VEC in the virtual environment has a corresponding VRML visualization. The system level architecture of MAVE is shown in Figure 2.

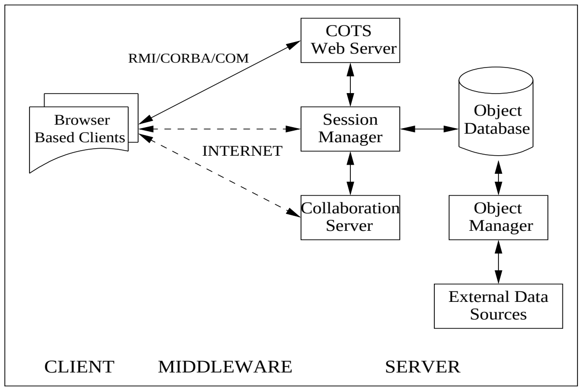

Fig. 2. System-level Architecture

The client runs a traditional Web browser with a VRML viewer. We used the Netscape Plug-in Cosmo Player to execute VRML scenes described in this paper. The VRML data originally comes from a COTS (Commercial Off The Shelf) server. Shared VECs are managed and updated by the session manager. Persistent VECs are stored in the object-oriented database and one or more object managers allow other programs or events to change the data.

3 PSE and ObjectStore

Many issues arise when considering how to add persistence to VRML. A VRML persistence scheme should distinguish between static VECs and dynamic VECs. Previous work on VRML persistence has used traditional file storage techniques. Our prototype uses an object-oriented databases to provide direct support for persistence of objects. Three object-oriented databases we considered for the MAVE project are ObjectStore [6, 8], PSE (Persistent Storage Engine) [7], and PSE Pro. PSE is originally intended for small single user databases, and is not intended to support high volumes of updates or queries over a large collection of objects. PSE Pro does support large databases. ObjectStore provides object storage for both Java [5] and C++ [2] objects, and supports very large databases and multiple users concurrently accessing multiple databases.

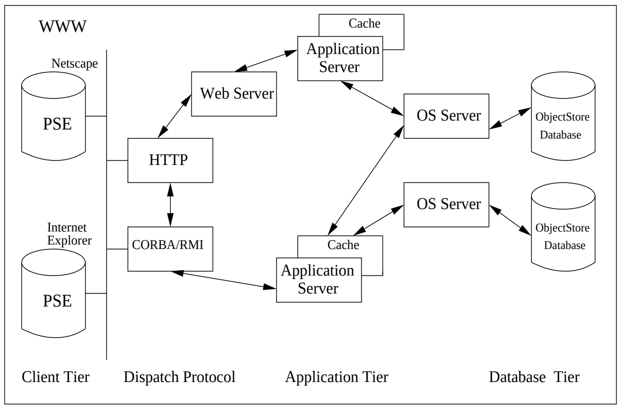

Fig. 3. Initial Database Server Architecture

MAVE must support multiple users with concurrent accesses. The clients must always have the most recent state of the object being used and VECs must be stored and accessed efficiently. Figure 3 shows our initial database server architecture. The *client* tier uses an ordinary web browser with a VRML plug-in. In addition, each client must run a copy of PSE to locally cache objects relevant to the local visualization display. The PSE API is a subset of the Object Store API. Thus, ObjectStore server can freely use PSE features and functions on the client. The VECs are downloaded from the ObjectStore server to the PSE. When the client wants to commit changes to the VECs, the modified objects are saved from PSE to ObjectStore. This approach will reduce network traffic and increase performance. The *protocol* tier is responsible for passing requests and responses among clients and applications. The actual protocol chosen could be CORBA, RMI, Serialization, or Sockets. The *application* tier provides a buffer between the client requests and the database. MAVE applications often have many users accessing the same database through the same web server. To prevent the web server from becoming a bottleneck, we introduced application servers. The web server can forward client requests to any number of application servers. The *database* tier is responsible for ensuring that all application components share access to distributed data.

The MAVE architecture thus allows us to decouple the GUI logic and the persistence storage logic. The VECs can contain arbitrarily complex GUI code. Application logic is closely linked to the database and can be easily tuned to manage large system loads.

4 Conclusions And Future Work

This paper has incorporated VRML persistence into the MAVE architecture. This offered an initial description of the use of ObjectStore for persistent storage of VRML objects. The MAVE database server architecture must support several features: multiple concurrent access, browser (Web) based clients, notification by the database of any changes in data, locking and synchronization, and multi-threading support. Future work will include complete implementation of several test cases to evaluate ObjectStore's ability to meet these requirements. The test cases will demonstrate the interaction between Java, VRML, and the object-oriented database. Our final architecture will include a web server, which dispatches transactions to one or more application servers.

References

1. Coble, J. and Harbison, K. "MAVE: A Multi-agent Architecture for Virtual Environments" *Proceedings of 11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, June 1998.
2. H.M. Deitel, P.J. Deitel, *C++ How to Program*, 3^{rd} Ed., Prentice Hall, 2001.
3. Elmasri, R. and Navathe, S. *Fundamentals of Database Systems*, 3^{rd} Ed., 2000, Benjamin-Cummings.
4. Hartman, J. and Wernecke, J. *The VRML 2.0 Handbook*, Silicon Graphics Inc, August 1996.
5. Lea, R., Matsuda, K., and Miyashita, K. *Java for 3D and VRML Worlds*, New Ride Publishing, 1996.
6. ObjectStore User Reference, *Object Design Inc*, www.odi.com, 1998.
7. OSJI and PSE Java discussion lists, *Object Design Inc*, www.odi.com, majordomo@odi.com, 1998.
8. Rogers, P. Off the Record Research: Object Database Management Systems, *Object Design Inc*, www.odi.com, 1997.
9. VRML, VRML-EAI, VRML-dbwork discussion lists, maintained by VRML Group, www.vrml.org, majordomo@vrml.org, 1998.