VESS: Coordinating Graphics, Audio, and User Interaction in Virtual Reality Applications

Jason Daly, Bryan Kline and Glenn A. Martin
Institute for Simulation and Training
University of Central Florida
{jdaly, bkline, martin}@ist.ucf.edu

Abstract

VESS is a suite of libraries designed to aid in the creation of applications for virtual reality research. It combines the power of a scene graph library, a flexible input device library, and support for other senses (audio, haptics, etc.). It then adds higher-level libraries that integrate graphics and user input, creating a single, coherent platform that is flexible and easy to use. VESS handles the technical challenges facing virtual reality applications, leaving the developer free to focus on the details of the application itself.

1. Introduction

Any work involving virtual environments (VE’s) involves dealing with several basic technical challenges. Among the most important of these are graphics, allowing the user to see the VE and perceive the object and features it contains, and input, allowing the user to interact with the VE. These two areas must be addressed by most (if not all) interactive applications that make use of a VE. The graphics challenge can be handled by one of the many graphics libraries available. The numerous packages that provide a scene graph structure are especially appropriate for VE applications. However, there seems to be far less software available to help deal with the remaining challenge of interaction, so input and user interaction must be handled by the VE software package itself. The other senses (audio, haptics, and olfactory) are also important, and should be supported.

In addition to these basic issues, there is the issue of coordinating them into a coherent system that is intuitive and useful for creating specific VE applications. To address these challenges, we have created VESS, the Virtual Environment Software Sandbox. VESS is designed to utilize existing graphics and audio libraries to handle the graphics and sound challenges. VESS also provides a comprehensive and extensible input library to handle the various devices commonly used in virtual environment applications. Basic support for haptic devices is also included (olfactory devices may be added later). Finally, it coordinates these basic elements, providing a multi-layered application-programming interface for the programmer.

2. Previous Work

There are several examples of virtual reality software (too many to list all of them here). Some examples include older packages such as WorldToolkit from EAI [1] and MRToolkit from the University of Alberta [2]. There are also several recent packages available. CaveLib from VRCO [3] provides an application base for the CAVE virtual environment. VR Juggler from Iowa State University [4] provides a dynamically-reconfigurable VR application framework, supporting a variety of input and display devices. DIVERSE from Virginia Tech University [5] provides a set of libraries that support the creation of VR applications. VESS is also based on our experiences with a previous library we developed at UCF/IST, called VEL [6].

3. Design

We considered many design objectives when we created VESS. Among these were flexibility, to provide a solid foundation for a variety of applications; extensibility, so the library can be updated to support new devices and techniques; and portability, so it can run on a variety of hardware and libraries. To help meet these objectives, VESS is comprised of several C++ class libraries organized into three layers.

The low-level layer consists of the System, Math, Scene, Input, and Sound Libraries. The System Library
provides access to system-level resources, such as graphics pipes, screens, windows, and rendering contexts. It is responsible for determining the available system resources and making them accessible to the developer. Its features include a database loader, viewpoint management, and control of the rendering process.

The Scene Library provides an abstraction layer between VESS and whatever scene graph library is being used for graphics rendering. VESS defines its own common representation that works on any underlying scene graph library and creates a mapping from key nodes in the native scene graph to VESS scene graph objects.

The Sound Library encapsulates the underlying audio API in a similar manner as the Scene Library. The Sound Library provides all system-level functionality needed to control and utilize the audio hardware. Sounds can be linked to any object in the scene graph, and the user’s “listening point” can be tied to the viewpoint to create 3D positional audio effects.

The Math Library is a small library providing useful 3D graphics data structures and operations. A vector, a matrix, and a quaternion class are provided, complete with all essential operations.

The Input Library provides classes to handle various input devices and systems. Input systems are the classes that drive the input hardware and obtain the input data. Input devices are abstractions of the physical device with which the user interacts; they store the data that is retrieved by the corresponding input system. Applications read data from the input device objects to handle user interaction.

The mid-level layer of VESS consists of the Motion Library. The Motion Library coordinates input data from the Input Library and virtual objects from the Scene Library into a coherent system for controlling the motion of virtual objects. The library consists of a kinematics class that encapsulates the physical state of a virtual object, and numerous motion models that operate on the kinematics class. Motion models can be arranged as stages in a pipeline to create more complex motions.

The high-level layer of VESS contains the Avatar library. This library is designed to assist in the virtual entity construction and maintenance process by automatically creating and managing a series of objects from the lower level libraries based on the information contained within a configuration file. The classes in the Avatar library can create all of the necessary objects from the lower-level libraries automatically and manage their state throughout the execution of the application.

### 4. Features and Benefits

VESS is flexible and is designed to be useful for a wide variety of applications. It can be used to create a fully-immersive VR training system, complete with full-body motion tracking and graphical and audio output to a head-mounted display. It can also be used to create a simple desktop 3D model viewer, controlled by a standard mouse and keyboard. VESS can be easily extended to include new features. New input devices, rendering techniques, audio features, motion models, and avatar constructs can easily be added by following their respective interface. The modularity and multi-layer architecture of VESS provides portability. To move to a different scene graph, for example, only changes in the Scene and System Libraries are required. The mid- and high-level libraries consist only of VESS code, so no porting is needed.

VESS also provides several features not commonly found in other VR software packages. Among these are the power of the Motion Library, and the unique design of the VESS scene graph abstraction layer.

### 5. Results

The final VESS design meets our initial design objectives. It has been used for several projects at UCF/IST, and the users have given positive feedback regarding its ease of use. It has been used to create a simple desktop 3D model viewer; a prototype simulator for a future combat system, complete with haptic feedback; and an immersive team training simulator incorporating networked virtual entities, full-body motion tracking, and positional audio. VESS has proven to be a powerful tool for VR application development.

### References


