VEWL: A Framework for Building a Windowing Interface in a Virtual Environment

Daniel Larimer and Doug A. Bowman

Dept. of Computer Science, Virginia Tech, 660 McBryde, Blacksburg, VA

dlarimer@vt.edu, bowman@vt.edu

Abstract: VEWL (Virtual Environment Windowing Library) is a library designed to provide a highly flexible interface metaphor for building window-based interfaces within a virtual environment. VEWL was built on top of DIVERSE and uses the object-oriented paradigm of signals and slots through the use of Qt. In addition to the windowing metaphor presented this library was designed to provide an API (Application Programming Interface) to allow rapid development of new interfaces. This paper will describe the design and implementation of this library and provide preliminary results of a brief usability study.

Keywords: Virtual Environment Windowing Library Interface API

1 Introduction

Most immersive virtual environments (VEs) provide a three-dimensional (3D) user interface. This makes sense for 3D navigation, object manipulation, and the like, but there are other tasks within VEs, such as menu selection, that require only one- or two-dimensional interaction (Bowman et al., 2001). By providing the appropriate number of degrees of freedom (DOFs), interface designers can increase user efficiency and reduce errors. One way to provide 2D interaction in a VE is through an interface metaphor already familiar to the user: a window-based environment.

In this paper, we present VEWL (Virtual Environment Windowing Library), a flexible, object-oriented API (Application Programming Interface) for developing applications using windows within an immersive virtual environment (VE). The goal of this library was to provide a way to quickly build interfaces and display information within a VE. This interface is not meant to stand on its own, but instead to augment the virtual world by providing additional information and controls. One of the main limitations of VEs is the lack of logical input devices like buttons, sliders, and menus. VEWL is a window manager that supports the use of menus, windows, and buttons.

2 Related Work

Most 2D/3D interfaces, such as pen and tablet techniques (e.g. Angus & Sowizral, 1995), have required some type of physical surface for 2D input. Pen and tablet techniques provide limited screen space and require both hands and two trackers. The goal of VEWL is to provide a 2D interface that does requires neither a physical surface nor input devices specific to 2D input. This is accomplished through virtual constraints instead of physical constraints.

There have been other projects related to 3D window managers including 3DWM (Robertson et al., 2000) and Task Gallery (Elmqvist, 1996). The goal of these projects is to move current 2D windows into 3D in an attempt to improve upon today's desktop window managers. VEWL, on the other hand, was designed to augment existing virtual environments. Other projects have explored the placement of 2D content on the walls of a CAVE (e.g. Dykstra, 1994). These systems are limited in their ability to take advantage of the 3D environment by tying too directly to current 2D interfaces. VEWL is built out of polygons using SGI Performer to provide a truly native and appropriately flexible solution to providing WIMP-based interfaces in a VE. Each window provides a 3D coordinate system allowing the windows to display either 2D or 3D content.

3 User Interface

A primary goal of VEWL was to provide a usable interface that was intuitive and familiar to the user. One of the main usability issues within a VE is the difficulty of specifying points with six degrees of freedom: x, y, z, heading, roll, and pitch (Hinckley, 1994). In order to constrain the input required we put content on the surface of a virtual sphere. By keeping content tangent to the surface sphere the user only needs to point where they want the mouse and the other information can be calculated to have the windows always face the user and placed a constant distance from the user as shown in Figure 1. The end result is a virtual surface on which the user can point to control a virtual mouse.

3.1 Virtual Mouse

We implemented VEWL to run on a 4-wall Fakespace CAVE™, using an Intersense IS-900 tracking system to track the user's head and a hand-held wand. The wand has a two-DOF joystick and four buttons. The user points the wand, and the virtual mouse location is defined as the intersection of the wand's direction vector with the sphere.

Once the virtual mouse has been defined the rest of the interface can follow traditional 2D interface metaphors. To keep things familiar to the user one button was used for selecting and clicking, and another for contextual menus, like left and right mouse buttons. Because there are four buttons on the wand we chose to use a third button as the universal move button. When the move button is pressed anywhere in a window, it moves with the mouse until the button is released. This allows the user to easily place windows without requiring a precise selection of the window's title bar. The fourth button was left unused.

3.2 Applications Menu

A user can right click anywhere on the sphere and a menu is brought up with a list of available applications. When selected, an application's windows are placed on the surface of the sphere. To keep things flexible the applications menu is configured via an XML file that describes what applications to add to the menu.

3.3 Windows

Windows are represented as polygons that are tangent to the sphere at their center and are not spherically warped along the surface of the sphere.

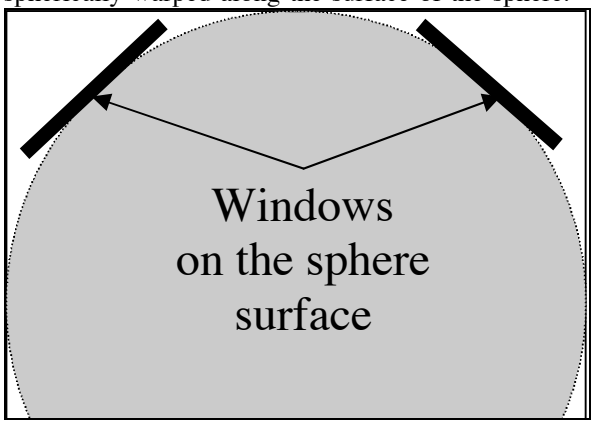

Figure 1 - Sphere Diagram

All of our current examples (see section 4) use 2D windows; however, windows containing 3D graphics are supported and encouraged. The mouse is projected onto the window using a ray cast from the wand through the sphere to the surface of the window. The location where the ray intersects the window is where the mouse event is generated in window coordinates. Users can then press logical buttons on the window or simply use information displayed by the window. As stated above, the windows can be moved using the third mouse button. The example windows all have a title bar that provides a button to close the window as well as a label. Windows do not require this title bar and can provide any interface they want.

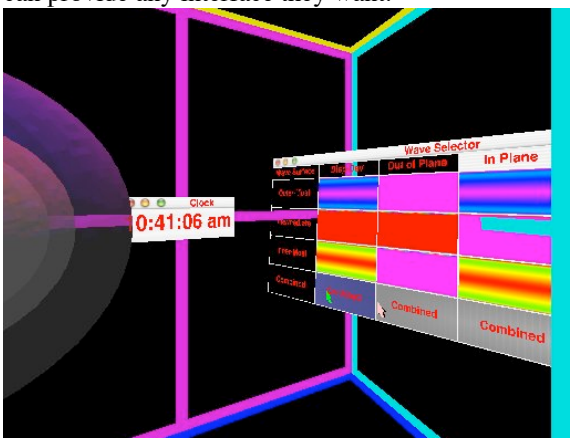

Figure 2 - Clock, Wave Selector & Glyph

4 Sample Applications

We have developed two prototype applications to demonstrate the power and flexibility of the API and the user interface metaphor. The first application is a simple clock that the user can place anywhere. It

serves as an example of how to provide additional information to the user to aid in the creation of information rich virtual environments (Bowman et al., 1999). It also demonstrates how content can be dynamically updated.

The second application replaces a current web-based interface for selecting different 3D wave models. This application shows a real-world use for VEWL, and provides an interface that appears to be much more efficient at selecting models than the previous web-based interface.

The Wave Selector provides 16 virtual buttons. Each button displays an image that represents the surface of the wave to identify the model. The user may use the virtual mouse to select the 3D model they would like to look at. When the user selects the model it is loaded and displayed in the CAVE. This is an improvement over the old interface, which required the user to leave the CAVE and click on an identical picture on a webpage to load a new model. The user can now compare and contrast models much faster than before without having to leave the CAVE.

5 Design and Implementation

The design of VEWL was informed by several design goals:

1) Reusability – Because this is an API all objects are designed to be modular
2) Ease of use – Like the user interface, the programmer interface was equally important when designing and implementing the API.
3) Extensibility – Objects were designed to encapsulate base functionality while not limiting what the programmer can gain access to.
4) Documentation – If an API is going to be used by developers then they will want good documentation.

To accomplish these goals we implemented all VEWL widgets using similar class names and structures to Qt and used SGI Performer calls to draw the widgets. This is what allows the 3D widgets and what provides good performance because the widgets are not rendered to a bitmap and then shown as a texture, although they can be.

In addition to building on top of Qt, we also used DIVERSE, which interfaces with various input devices. DIVERSE is designed to be a modular API for building reconfigurable, scalable, extensible, and device independent VEs (Kelso et al, 2002). The combination of Qt and DIVERSE provided a powerful, yet flexible foundation from which to build a window manager and widget toolset. VEWL was designed as a DTK shared library that can be loaded along with any existing DIVERSE application without having to change a line of code. This was essential because VEWL was designed to augment existing VEs, not to replace them.

VEWL was also designed to allow plug-in applications that can be compiled separately. The clock and Wave Selector mentioned above are both compiled separately from the rest of the library and are loaded into the applications menu via an XML configuration file. The goal was to provide a window manager that can dynamically load applications.

6 Usability Evaluation

We conducted a preliminary usability evaluation to see if the interface was easy to learn and use. Five student subjects participated in the study. Before the evaluation began we gave a demonstration of how the virtual mouse worked and described the functionality of each button. Subjects used the interface to open, move, and close windows for 2-3 minutes to allow them to get used to the CAVE and overcome the initial learning curve associated with using the wand. After this exploratory phase, subjects began a more focused task-based evaluation.

6.1 Tasks

The first task we asked them to perform was to open four clocks and place them in different locations around the CAVE. To accomplish this task, subjects had to make 8 menu selections and move 4 windows. We then asked them to close the windows and open the Wave Selector. This tested their ability to close windows and required two more menu selections and another window placement. Finally, we asked them to view all 16 wave models and then close the Wave Selector. This task required them to press buttons within a window. We watched for failed attempts at selecting menu items, pressing buttons, moving windows, and closing windows because these were the most primitive actions.

6.2 Survey

After using the interface, subjects completed a survey that asked them to compare the VEWL interface with similar desktop interfaces that they are familiar with. They rated the ease of using the menus, closing a window, selecting a button,

moving a window and pointing the mouse. Questions used a 7-point Likert scale, where 1 meant "near impossible" and 7 meant "as easy as a desktop computer". We also asked subjects about what they found most difficult and what was the easiest for them.

6.3 Results

The results were highly varied from one user to another. Some users had no missed menu attempts out of the 10 times they had to select a menu option. They were also able to select the menu options quickly. Another group of users missed the desired menu option in 3 out of 4 attempts, and it took them significantly longer to make each selection. When this group was asked what made it difficult, they said the mouse would move when they pressed the button (the so-called "Heisenberg effect" (Bowman et al., 2002)). This was most likely caused by the fact that there is no physical surface on which the wand rests. If not countered, the pressure on the button would move the wand. The average rating of the ease of using the popup menus was 5.0, which suggests that, while they were not as easy to use as their desktop counterparts, they were still quite usable.

Subjects had no problems moving windows or pressing buttons on the Wave Selector. When asked, many of the subjects said that these tasks were what they found to be the easiest and this is mirrored in their average usability rating of 6.2.

Closing the windows was the most difficult task for all the users, and some of them could not complete it at all. The main complaint was that the close button was too small and too close to the edge of the window. Despite the problems one subject was able to close all the windows successfully on his first attempt. This indicates that pointing the wand with precision is the limiting factor when it comes to closing windows. Similar results have been reported for ray-based selection of floating menu items in VEs (Bowman, 1996). This task got an average usability rating of 2.2 from the subjects.

6 Future Work

VEWL has many possibilities for future work. The API could expand to include many more widgets and interaction techniques. VEWL has the potential to be a unified API for augmenting virtual environments with information rich interfaces. Other areas for future development include improving the precision and accuracy of the virtual mouse, providing effective text input, and developing truly 3D widgets and windows.

References

Angus, I. And H. Sowizral (1995). Embedding the 2D interaction Metaphor in a Real 3D Virtual Environment. SPIE, Stereoscopic Displays and Virtual Reality Systems.

Bowman, D., Wingrave, C., Campbell, J., Ly, V., & Rhoton, C. (2002). Novel Uses of Pinch Gloves for Virtual Environment Interaction Techniques. *Virtual Reality, 6*(3), 122-129.

Bowman, D. Wineman, J., Hodges, L., and Allison, D. (1999) The Educational Value of an Information-Rich Virtual Environment. *Presence: Teleoperators and Virtual Environments* , vol. 8, no. 3, June 1999, pp. 317-331

Bowman, D. (1996). Conceptual Design Space: Beyond Walk through to immersive Design. In D. Bertol (Ed.), *Designing Digital Space* (pp. 225-2360. New York: John Wiley & Sons.

Robertson, G., et al,(2000) *The Task Gallery: a 3D window manager*, Proceedings of the SIGCHI conference on Human factors in computing systems, p.494-501, April 01-06, The Hague, The Netherlands

Kelso, J., Satterfield, S.G., Arsenault, L.E., and Kriz, R.D., (2002) DIVERSE: A Framework for Building Extensible and Reconfigurable Device Independent Virtual Environments, Proceedings of IEEE VR 2002, Orlando, Florida, March 24-28.

Hinckley, K., Pausch, R., Goble, J.C., Kassell, N.F., (1994) *A survey of design issues in spatial input*, Proceedings of the 7th annual ACM symposium on User interface software and technology, p.213-222, November 02-04, 1994, Marina del Rey, California, United States.

Elmqvist, Niklas, (1996) *3Dwm: A Platform for Research and Development of Three-Dimensional User Interfaces*, Department of Computing Science Chalmers University of Technology and Goteborg University SE-312 96 Goteborg, Sweden.

Dykstra, P., (1994), *X11 in Virtual Environments: Combining Computer Interaction Methodologies*, j-X-RESOURCE, vol. 9 no. 1, pp. 195—204, Jan 1994

Lindeman, R., Sibert, J., Hahn, J., (1999) "*Hand-Held Windows: Towards Effective 2D Interaction in Immersive Virtual Environments*", Proc. IEEE Virtual Reality '99, pp. 205-212.