Abstract

This paper presents a brief description of a method for running 2D text based or graphical applications from within a virtual environment. This is accomplished by a port of the X11 server which renders into a shared memory segment, which is in turn texture mapped onto the surface of an object in a 3D virtual world. Events that occur in the virtual space can be translated into pseudo mouse or keyboard input events, which are passed to the X server via shared memory queues, fifo's, or network connections.

This system allows any X application to be used from within a virtual space, without rewriting any software. Any number of such servers can exist and be manipulated inside of the virtual environment. Use of transparency as a background color allows text windows to float in space or be used as heads-up displays.

1 Introduction

Virtual reality has emerged as a new method of human-computer interaction. While extending the expressive power of both the computer and user, it usually involves leaving our existing software behind and requires the development of completely new applications. The many applications created using the Windows, Icons, Menus, and Pointer or WIMP interface, are generally not available and some would even argue are of little interest in virtual environments. Yet in the real world we still use countless flat surfaces for information display - from printed pages to television screens - and the utility of such displays is not diminished by working within 3D virtual reality.

As part of an effort to develop a synthetic environment system for scientific visualization and simulation, the desire arose to do things such as graph functions, display images and MPEG movies, and run Internet teleconferencing software. X11 programs already existed for all of these applications, and the idea of reimplementing them all seemed daunting. The idea of building an X11 server which actually renders in 3D was discussed, but an intermediate useful step was found, i.e. to render into a shared memory segment and use that memory as a texture applied to 3D geometry.

2 Implementation

The X11 Sample Server provided by the X Consortium is a highly portable software implementation of an X server. The first step in this project, involved porting the sample server to a pseudo hardware platform, i.e. making it functional completely independent of specific hardware. The approach taken was to create a server called Xmem which allocates a block of shared memory for its display frame buffer, and accepts pseudo keyboard input events from a fifo queue. Since the display is in shared memory, it can be accessed directly by other programs.

The first few bytes of the shared memory segment contain the width and height of the display, and whether it is color or monochrome. Following that is the actual display contents. Following the usual X convention, the server listens to port 6000 + display_number for client connections, as well as to the UNIX domain socket of /tmp/X11-unix/X1. It also creates an input fifo called /tmp/X11-unix/X1-events where external programs can send pseudo keyboard and mouse input.

Once Xmem is running, a window manager can be started on it, and any number of clients can connect to it in the usual X fashion. Keyboard and mouse input can be passed to Xmem down the Xn-events fifo. Any number of programs can write to this same fifo.

2.1 X Servers in an X Window

In order to see the virtual X server's output, an X program was written to display the shared memory...
segment in an X window. The program creates a window on an X server, attaches to Xmem's shared memory display, and repeatedly draws its contents into the X window. In addition, any keyboard and mouse events which happen inside of this application's window are converted into messages and sent down the input fifo to Xmem.

The result is a complete X server, running entirely within a window residing on another X server. This server within a window gets its own window manager and its own client applications. Since this server display window is nothing more than an X client program, the entire thing can be moved and iconified as with any other application.

The response time was far better than the author had hoped. Dozens of such events per second can be handled. Fully interactive use of the X server is possible in this way. Button presses, keyboard typing, window moves or resizes, are all nearly as responsive as on a native X server.

2.2 X Servers in 3D Space

The 2D array of pixels that make up the X display was used as a texture and mapped onto a polygon (or polygons) in the virtual world. The existence of hardware accelerated textures makes this possible. Silicon Graphics IRIS 4D workstations were used to test this texture mapping idea. A polygon is created in a GL program, and the shared memory X display is used as a texture and applied to the polygon. The user can then walk around in the virtual world and see the X display(s). These X displays could appear in many different forms: as large screen televisions mounted to the wall of a room, as tablets that can be picked up and carried around, as see-through screens floating in space or following the user as a heads-up display (HUD). Input events from devices such as 3D mice can be turned into messages and sent down the fifo to the X server.

3 Future Work

At least three developments would help make this system more useful. Foremost is to increase the server display update rate, either by hardware improvements in the speed at which texture contents can be changed, or by a new approach allowing direct rendering into non-orthogonal view windows\(^2\). Second, the resolution of most head-mounted display devices is far lower than that of workstations, making it difficult to read text information from within virtual environments. Finally, improvements in input devices such as 3D mice and head tracking techniques would make control strategies for 3D "desktops" easier to implement.

\(^2\)A fast path to texture memory could support many interesting ideas from algorithmic textures to live video projection.