Creating an interface to VR Juggler for the applications user

by

Michael Ronald Cook

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Engineering

Program of Study Committee:
Carolina Cruz-Neira
James A. Davis
James H. Oliver

Iowa State University

Ames, Iowa

2002

Graduate College
Iowa State University

This is to certify that the master's thesis of

Michael Ronald Cook

has met the thesis requirements of Iowa State University

Signatures have been redacted for privacy

TABLE OF CONTENTS

TABLE OF FIGURES

ACKNOWLEDGEMENTS

Absolutely first to be acknowledged shall be my parents. Their constant support, encouragement, and sincere intentions for me have formed everything I am and everything I will be.

Dr. Davis - numerous times - went *beyond* the extra mile to help with all the paperwork that came along with changing schools and majors between my undergraduate and graduate years.

Johnathon Gurley, the Java man, made the Java port of the VR Juggler player more than a good idea. The Windows version of the player showed me that the player design worked and was a good idea. The Java version will hopefully show everyone, everywhere that the player is a good idea.

Dr. Cruz-Neira was a great help in finding a direction for me to focus my scholarly interests for this thesis and guiding me through the process. With this direction established, she allowed and trusted me to explore it in my own fashion.

Finally, while it'd be difficult to list them all here, crucial to my academic advancement and success were all the teachers, professors, and instructors whose classrooms and laboratories often showed me more than the inside of a book or echoed verbatim the recitation delivered years earlier.

My parents gave me the backyard and told me I could have the stars; my teachers showed me how I could reach some and enjoy the rest. From here I will continue to travel as the better person they have all made me.

ABSTRACT

Virtual reality (VR), particularly in the successful wake of video games, is leaving research laboratories to become a tool of its own. The use of VR will continue to grow, and as it does, there will be the demand for standards and established mechanisms to simplify its use. One of the most important components of effective VR systems is the software that controls the applications. As VR becomes more popular, its success will depend on how easily the software is to install, upgrade, and apply to a user's particular need.

VR Juggler is currently one of the most used VR software toolkits by research and industrial groups. With a growing number of VR Juggler sites around the world, it is critical to enable the use of VR Juggler applications without the need to know the development behind them. Additionally, since VR Juggler continues evolving, these application users need a tool to help them manage the different versions and configurations.

This thesis focuses on the design of a tool that automates the setup of VR Juggler to run its applications. The initial design and implementation of this tool, the VR Juggler player, is presented, followed by a discussion of successive design refinements based on user input. The final VR Juggler player design provides functionality and portability across different computer platforms.

CHAPTER 1. INTRODUCTION

The abundance of computer software can be overwhelming. Endless web sites have shareware, freeware, adware, and commercial software. In this environment, software lapsing in visibility for as little as a month can be forgotten and lost, quite possibly forever. The software that remains is that which either entertains us, proves to be useful or handy, or proves to be necessary to do work. In each of these areas of software, one of the most important parts of any piece of software is how easy it is to install, upgrade, and apply for a user's particular need – or its usability. Software that is complex, bulky, or difficult is sought by competition and often consumed. One of the determining factors of a software's mortality is its usability.

1.1 Virtual Reality

Virtual reality (VR) is the process of synthesizing real life situations or simulating abstract worlds to the senses, and in its vernacular or colloquial form is regarded to employ the use of sophisticated computer hardware and software [4]. However, virtual reality can take on many forms. In its most simple form, 3D glasses, providing the user with the illusion of depth, could be considered a rudimentary form of virtual reality as it pulls the user into the movie's environment. A simple graphical program that allows a user to test the strength of gear teeth made of different metal alloys allows this user to evaluate materials and designs in the realm of virtual. These examples are on the simple end of virtual reality. On the complex end, there are environments like Iowa State University's C6 in which users are immersed in a three-dimensional space and whose movements can be tracked to create interaction with the worlds. Complex simulations can

include changing simulated objects and behaviors at runtime, such as engines in cars, or even the individual components of an engine.

Figure 1.1: The C6

Virtual reality can be used, in varying degrees, to create immersive worlds. Being immersed means that the user has little distraction from real world elements that can detract from the virtual experience. Virtual reality, although still emerging as a technology, is not a new idea. As is often the case, science follows fiction. In 1953, Ray Bradbury wrote about "parlor walls" in his novel Fahrenheit 451 [2]. Instead of a television, people watched dramatized events unfold from the confines of big walls. The wife of the main character in the book, Mildred, was particularly interested in this technology and asked her husband about getting the fourth wall. If this fourth wall was a projection on the floor, Bradbury would have predicted the CAVE Automatic Virtual Environment (CAVE). This CAVE, after which Iowa State University's C2 (now C4) virtual environment is modeled, uses projectors and mirrors to display images on ten foot walls that are in front of, to the sides of, and beneath the user [6].

Figure 1.2: The C2

Immersion is also accomplished in both the C2 and the C6 by using special glasses that affect the way that images are shown to the eye. By tricking the eye, images on the screens of these environments can appear to come out from or into the walls. Often, new users or guests of the C2 will duck or jump when they think they will run into something in this environment because it looks like the virtual objects are coming right at them.

A final possible component of the virtual reality experience is interaction. Interaction helps make the virtual world a lot more real to the user, and can bridge the gap between virtual and reality. Television has accustomed people to watching events unfold before them without their influence. The buzz over the last couple years has been to bring out new "reality" shows. These still do not bring the viewer into the scenario or put the user in any situation. This kind of show only shows the viewers that are sitting on their couches all the more things that they are not doing. There is no "reality" for the viewer. Virtual reality applications that allow the user to change things, alter variables, or interact with other objects in the virtual world. This could even include other people in either other virtual environments or in the same one. Besides including how the user affects the environment in a virtual world, the virtual world can affect the user.

Depending on how the user reacts, the virtual world could change itself, and perform a number of actions either triggered by the user or by simulation. It could adapt itself to make navigation easier if it senses the user might be having trouble. If the simulation is to learn how humans find paths, the virtual world could change itself as a subject makes its way through. [7]

Interaction can also be used for a number of training purposes. Medical procedures can be practiced numerous times on virtual patients, thus making surgery on a real person much safer. Simulations can be used to teach students how to drive before they get behind the wheel of a car, semi, bulldozer, school bus, or any other wheeled vehicle. Flying simulators can be used to teach pilots how to fly and how to execute dangerous maneuvers. They can practice firing on targets without having to fire live rounds or missiles. NASA has space shuttle simulators that use virtual reality to prepare astronauts for the dangerous task of flying a shuttle into space and returning it to the Earth.

1.2 Purposes of VR

VR can be applied in a wide array of areas and problems. Reduced risk or danger, enhanced data exploration, and reduced cost are typical examples of areas in which VR has been successfully applied [3].

The first area, reduced risk, includes testing new medical experiments on virtual people so that a real one is not put at risk. Doctors can test how their virtual patients react to new procedures. They can then analyze these results comparatively to find if there are better ways to handle the new procedure or if the risks to a real person are acceptable. This can also pleases animal rights activists because it not only reduces the risk to humans, but also reduces the number of animals that are used when trying each new

procedure. Reduced danger may include simulating nuclear tests or following microwave signals – interaction that would not be safe for a human to experience in reality.

The second area, enhanced accessibility, means using virtual reality to see things that the human eye would otherwise not be able to. This could include simulating drug interactions, cell mitosis, molecule formation, or even atom configurations. These are things that are not readily accessible to a human, but the knowledge that we have may very well be replicated into data. This would then be used when the simulation is run. Simulation can also be useful for deep sea projects or projects that would be executed in space.

Finally, low cost can be very important when developing new products, and can greatly aid the bottom line of budgets. An excellent quality that virtual reality lends is that it can move through many different prototypes without cost of the models actually having to be built. When a model appears to work well in simulations, then scale models can be assembled and tested, but a lot of the guesswork in model evolution can be run through simulation. Virtual reality can also cost less due to hardware reusability. Instead of having to build 15 prototype models, a VR environment or a simulator can be set up somewhere, and left without having to be changed with each new prototype or even each new product. [12]

VR can also be used for recreational and entertainment purposes. Numerous games have been developed lately where the player is placed in a first-person view of the game and uses the keyboard and mouse to move the player around in an environment. A short list of these includes Max Payne, Unreal Tournament, Quake, Project: I'm Going In, and Return to Castle Wolfenstein. In each of these, players are allowed to move around in different "worlds", and interact with both the worlds and the other players in them. Some of these games even include options to play them in a "multiplayer" mode, which allows more than one person to be in the same game [1].

1.3 Definitions

The concept of *virtual reality* has already been introduced, and the purpose of repeating here is to establish the parameters of its use within this thesis. For the purposes of this thesis, "virtual reality" will mean running any software simulation on and computing resource using any hardware configuration. This software can have any degree of immersion, any range of interaction, and can be simple or very complex.

VR Juggler is an open source, freeware virtual reality software package written and maintained at Iowa State University. According to its website, VR Juggler "is scalable from simple desktop systems like PCs to complex multi-screen systems running on high-end work stations and super computers" [13]. This means that one of the goals for VR Juggler is to create a VR library that ports across different hardware configurations and architectures without the need to recode the program. In fact, the same program can be run with different hardware configurations with only different arguments passed to it.

In the context of this thesis, a *distributed system* will consist of two or more networked computers. Each computer can access files on the other computer and the computers should be on the same network for faster communication. The configuration of the files on the computers can be thought of as client/server. The computer that is running the application is the client and the computer(s) that host(s) the versions of VR Juggler would be server(s). The application itself can even be on the server and run on the client. Minimally, the client will have a local configuration and its own VR Juggler disk space to which it has exclusive read and write/delete access to.

Another topic in this thesis that will be addressed is *computer security*, which has numerous aspects and they range from encrypting a single, particular file on a computer to securing the entire computer from theft. Security, in the context of this thesis, will refer to the unwanted altering or deleting of files. The mechanisms employed in the application

are in no way all-purpose, and can only protect a user from deleting files from the locally or remotely stored files while using the application. It will be up to the user or the administrator to make sure that the files are securely stored using operating system mechanisms or other security programs if it is determined to be a concern. However, deleting these files is nothing more than inconvenient since they are not crucial to the system and can be easily downloaded again.

1.4 Motivation

Virtual reality, particularly in the successful wake of video games, is leaving research laboratories to become a tool of its own. The use of virtual reality will continue to grow, and as it does, there will be the demand for standards and established mechanisms to simplify its use.

Having VR Juggler applications available for users to view can show off the power and usefulness that VR Juggler can bring to the VR Juggler these users. Before having to know how to use VR Juggler, users should be able to use it to run applications. A simple interactive graphical user interface (GUI) that handles the complexity of setting up VR Juggler for execution will help new users and allow application developers to quickly share their applications. Many users may choose to continue using such a tool as an automation tool even after learning how VR Juggler works.

This thesis addresses the need of a tool for running VR juggler applications by designing and implementing the VR Juggler Player. The player's intention is to bring VR Juggler and VR Juggler applications to those people that are not computer savvy or are unfamiliar with VR Juggler. This player also takes advantage of the web in ways that will be shown later in this thesis. Application developers may also feel more inclined to use

VR Juggler if they know that their users will have available a player that will make using their application easier.

VR Juggler is distributed as an Open Source product, available for free to everyone from the VR Juggler website[1]. The VR Juggler player will make VR Juggler accessible to these same people. Without having to know what is going on, users will be able to point and click their way through the setup that is required to run a VR Juggler application. This is something that is desirable to both the Virtual Reality Application Center's (VRAC's) clients and to the global VR Juggler community.

1.5 Scope and Context

This thesis will review previous efforts to make VR Juggler easier to use and discuss how they worked and what did not work about them. It will then discuss the need for a VR Juggler player. Following this, the design will be shown for a player whose intentions are to keep track of VR Juggler versions and to make using VR Juggler applications easier. This player will, of course, be available via the web for users to download. The VR Juggler player will not be a universal solution for VR Juggler applications. Since VR Juggler adds to programming functionality available to the application developer, it is possible to create applications that would fall out of the boundaries of the player. Therefore, a simple application will be used to test basic functionality. Finally, this player will be tested in various situations using various platforms.

[1] http://www.vrjuggler.org

CHAPTER 2. BACKGROUND

2.1 What is VR Juggler?

VR Juggler is a virtual reality development environment. Used for both developing and running virtual reality applications, it incorporates existing graphics libraries such as OpenGL and OpenGL Performer™ with a standard application programming interface (API) that is portable across machine architectures and operating systems. An application may need to be compiled anew for each platform, but in general, the program code should not need to be modified. This API links to the VR Juggler libraries and tools that are installed on the computer in a place that has been specified to it. This place is defined to the computer by the environment variable `VJ_BASE_DIR`. When a VR Juggler application is run, it looks here for the controls that it needs.

VR Juggler also tries to abstract the devices used to interact with the world from the application itself. This is done through separate, and usually multiple, configuration files that are specified when the application is run. This is how the same application can be run on a desktop PC with a monitor and mouse for testing, and then shown in its full glory in the C6 to awe spectators. VR Juggler comes with a set of configuration files that can be used for defining the interaction between the user and the application. However, the application does not know where to find these files. These need to exist in the same directory as the application or the use must input the entire path to the configuration files.

Once VR Juggler is in the directory that `VJ_BASE_DIR` points to and the configuration files where the application can see them, the program can be run. These two pieces of VR Juggler are essential to any VR Juggler application, but can be difficult for the inexperienced user.

2.2 Web-based Viewers and 3D Browsers

There are a number of tools popularly thought of as virtual reality, though they are mostly interactive 3D viewers or multimedia tools. This misnomer should be duly noted, and a few of these multimedia – not virtual reality – applications will be examined here. These tools in general provide very little interaction for the user and cannot provide any immersive features.

Commercial players exist for 3D but they do not support immersive display and interaction required for virtual reality. The Virtual Reality Markup Language (VRML), is a language that was written specifically for virtual reality on the web. It uses geometric primitives to create models and functionality can be added with JavaScript. Cosmo Player is one of the most popular players for this type of web-based environment.

A drawback of a VRML world is that it can be difficult to view offline. Saving a .wrl file from its uniform resource locator (URL) will contain the world's geometry, but only links to the text and other images that the user is responsible to find. Generally one would need to open the VRML script, look through it, and piece together the addresses of the textures. Another shortcoming of VRML is that it is not very well suited for scientific simulations.

Apple also makes a package for its QuickTime movie viewer. There are three kinds of movies available to the VR author. The first is a panoramic view of a scene that the user navigates by rotating about a point, much like being spun around, and can look up and down. The second type of movie is the movie of an object. An object is presented to the user who is then able to turn it around using a mouse and look at different sides of it. Finally, there is a scene. A scene can have both objects and panoramic views in it. Rudimentary world interaction allows the user to pick up predefined objects and move between panoramic views by clicking on "hot spots", like a web link [5].

QuickTime Virtual Reality (QTVR) has a fair balance of positive and negative attributes. The QTVR movies look very nice in contrast with geometric-based worlds. However, this is because photos must be taken of all of the objects and scenery to be used in a QTVR movie. These photos must also be numerous and consistent enough for the software to be able to connect them together. The software provides enough interaction for its use, but is not at all equipped to handle scientific or real life environments or simulations. QTVR creates aesthetically pleasurable virtual reality worlds, but does not allow for the functionality that VRAC needs.

2.3 Previous Work

There has also been some work to develop web-based, or web compatible, tools for the distribution and use of VR Juggler applications. At VRAC, there have been several efforts to automate the use of VR Juggler applications. The current system that starts VR Juggler applications is called *Demofly*. Demofly is a web-based tool that is available on the private VRAC intranet, but cannot be used or accessed by outside users. It is exclusive to the IRIX operating system running on SGI machines as it uses shell scripts that facilitate the use of demonstrative VR Juggler applications. Users of Demofly had only to click on the web browser links to start up a VR Juggler application or to shut it down. Demofly is shown in figure 2.1.

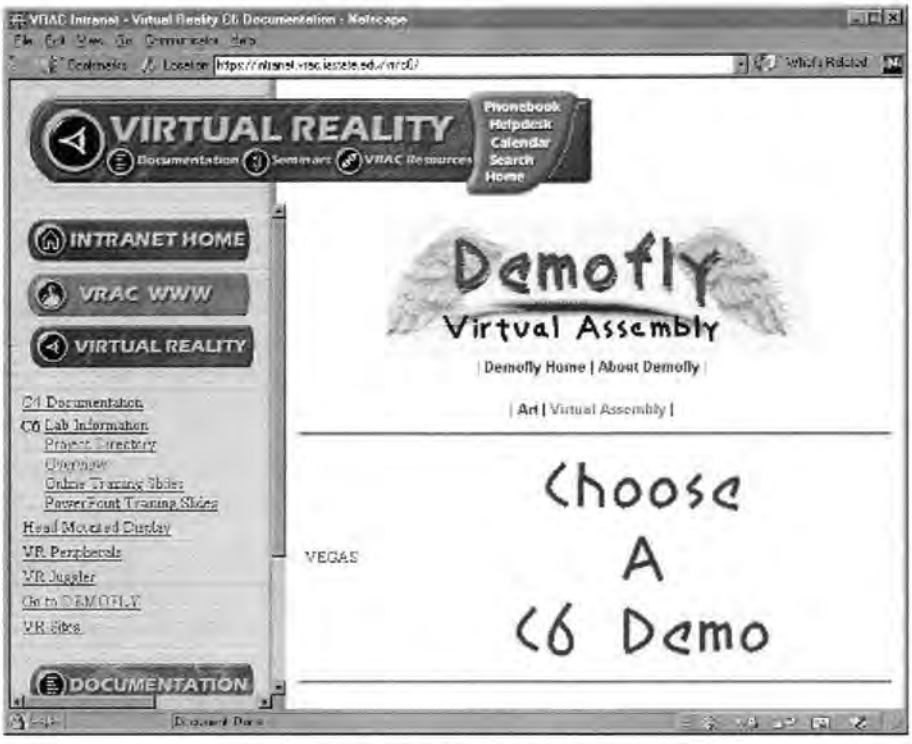

Figure 2.1: Demofly

In 2000, Shaohua Liu published a creative component that focused on creating an application that would serve as a plug-in for the Netscape Navigator web browser [9]. However, this plug-in approach can limit the handling of VR Juggler applications. A web browser will be needed to run the plug-in. Users on the Microsoft Windows platform might have to install the browser, as one comes preinstalled and some choose to simply use that one.

2.4 Usability

Following in the tradition of the mouse, windowing environments, and even the automobile, things that make users' lives easier are the things are used. The VR Juggler Player will make using VR Juggler applications and VR Juggler easier. Ease promotes

use. The player will automate the VR Juggler shuffling that will be required of the user, easing that from their responsibility. The player has options to help users manage their use of disk space without the users' needing to know how it is being handled. The VR Juggler player is designed to be easy to use for either the home or corporate user. The player will also remove from the user the responsibility of moving the configuration files to where they need to be and if the user does not know which configuration files to use, there is a default setting on the player for a standard desktop PC workstation.

CHAPTER 3. CONCEPT TO IMPLEMENTATION

3.1 VR Juggler Player Design

The goal of the VR Juggler player is to make the running of a VR Juggler application as seamless as possible, by making the user only input a few necessary arguments. Appearing to be seamless will come from the integration of the version acquisition, installation of the VR Juggler version, installation of the configuration files, and running the application all from a convenient, user-friendly graphical interface.

Running a VR Juggler application requires the following actions be taken:

- The appropriate version of VR Juggler must be on the system where the application can see it.
- The environment variable `VJ_BASE_DIR` must point to where it is installed so the application knows where to find VR Juggler.
- The needed configuration files for input and output must be in a place that the application can find them.

The VR Juggler player should encapsulate this process from the user and hide most of the work so the user does not need to know what is happening.

Figure 3.1 is the initial design sketch for the VR Juggler player. Roughly, the elements on the right should be on the local computer. Those on the left can be either remote or local. It was decided that the VR Juggler player should not be stored in the directory where VR Juggler is kept because the player itself will be moving files in and out of this directory, and it was desirable to prevent the player from performing file operations on itself. In the case a distributed system is taken advantage of, these would also need to be separated as the administrative configuration file (admin config) would

need to have read only file access and the VR Juggler base directory should have both read and write access.

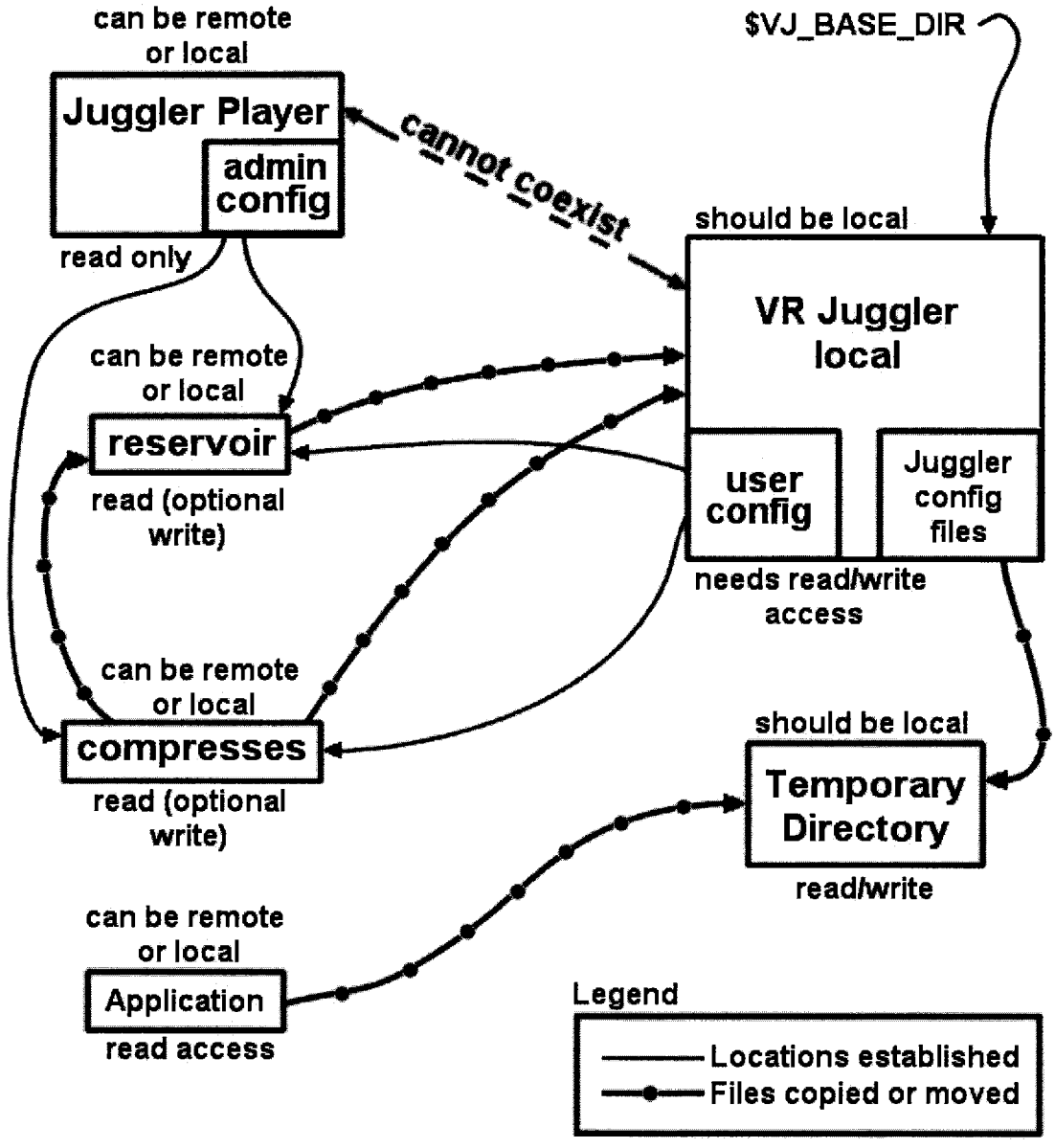

Figure 3.1: Concept for VR Juggler Player

3.2 Environment Variables

VR Juggler uses an environment variable to tell applications where it exists. When an application is run, it looks at the directory to which this variable points for VR Juggler. It will be important that the version of VR Juggler to which this variable points is the correct one.

Either the version of VR Juggler can be loaded into where `VJ_BASE_DIR` points, or `VJ_BASE_DIR` can be changed to point to where the version has been installed. Because not all Windows operating systems allow global environmental changes without restarting the computer, it was decided to keep `VJ_BASE_DIR` static and change the contents of the directory to which it points.

3.3 Version Management

VR Juggler designs intend to maintain compatibility between versions. However, implementation problems often make successive versions of VR Juggler incompatible with each other. Fortunately, the 1.0 family has been mostly binary compatible, meaning that the binary executables compiled from this version family can usually be run with different versions from the 1.0 family. When VR Juggler 1.1 is introduced, this compatibility will be lost as version 1.1 does not appear to be completely compatible with 1.0. In the general case, and for the foreseeable future, it is to be assumed that a VR Juggler application should be run using the version of VR Juggler with which it was compiled. The burden is now placed on the user to make sure that the installed version of VR Juggler matches that of application. The best indication a user can find to determine what version of VR Juggler is installed is to find the base directory for VR Juggler (`VJ_BASE_DIR`), and open the RELEASE_NOTES file using an appropriate viewer. This is not an overwhelming amount of responsibility for a user, but still more than one might

like just to use an application. One of the goals of the primary goals of the VR Juggler player is to make version management and acquisition easier for the end user, even relatively transparent. The VR Juggler Player will also mask how the versions of VR Juggler are being maintained and - after installation - where.

However, since the VR Juggler applications do not report what version of VR Juggler they require to run, it will be important to know which applications run on which versions of VR Juggler. This can be done a couple different ways. The first would be to simply include the version of VR Juggler in the filename as shown in Figure 3.2.

Figure 3.2: Version-incorporated Filename

The second option would be to create a directory structure in which applications are stored according to their necessary VR Juggler version. A simple example of this is shown in Figure 3.3.

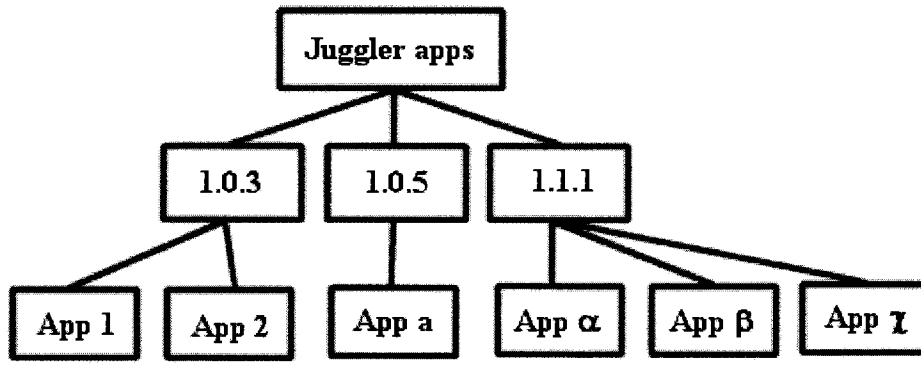

Figure 3.3: Example Directory Structure

3.4 Configuration Files

VR Juggler uses configuration files to specify how an application should be run, with respect to its hardware and input and output. These configuration files allow the user to specify at the start of the run of the program what kinds of input to expect, where it will come from, and how to handle it. For example, different configuration files would be used for a standard PC setup where a keyboard and mouse are the input devices than a setup where the input is a wand and trackers. They also specify how the resulting output should be displayed, whether it be to the four walls of the C4, the six walls in the C6, or a single monitor output.

These configuration files will need to be input to the VR Juggler player since they can change with each run of an application. The same application can have its output or input redirected to any devices that the computer can see, and where it looks or writes depends on the configuration files that the user has input. These configuration files, once specified, will be copied to the same place as the application file.

3.5 Version Acquisition

Since VR Juggler versions are not expected to be compatible with each other, it is important to be able to find the version of VR Juggler needed for a specific application. The VR Juggler player also takes care of this. When the player is installed, it asks for the location of two directories. The first is a reservoir directory, which contains uncompressed versions of VR Juggler. These uncompressed versions should be on a close, fast network connection or even on the same system so that copying the files is quick and easy. Upon failure in finding the needed version of VR Juggler in the reservoir, the player then looks in a directory called "zips". This directory contains versions of VR Juggler in compressed form. So they're faster to transfer, these could be stored on a

slower connection, most likely on another computer. The reservoir and zips directories would be ideally within a local area network. These would be versions hosted locally, and would be the versions that the company, organization, or user frequently uses.

If the version of VR Juggler that the user specified to the player cannot be found in either the reservoir or the zips directory, then the player turns to the Internet where it then attempts to download the VR Juggler version from an ftp server on sourceforge.net. All possibly-used version of VR Juggler should be kept on sourceforge for users to find. After an attempted download, the player checks that the file was successfully downloaded and handles it according to the settings. Should this final option fail, the player will report that the requested version of VR Juggler is not available and suggest that the user either try again or check that the version is valid. A flowchart of the version acquisition can be seen in Figure 3.4.

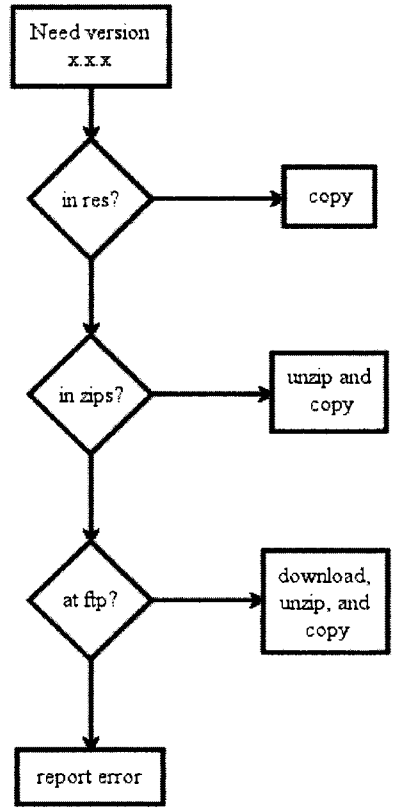

Figure 3.4: Version Acquisition Flowchart

Readers might argue that storing just the compressed versions of the VR Juggler versions would be sufficient and that there is no need for the reservoir. However, the reservoir and the compressed files do not need to exist in the same place. The reservoir could even be stored on the local system for very fast copying. By allowing this flexibility, this approach allows the user and/or the administrator to take advantages of as many aspects of the computer systems and network as possible with some strategic planning. Also, if the user or administrator really don't want the reservoir to be filled, or even exist, there is a configuration that would prevent it from being filled or used.

3.6 Temporary Directories

The VR Juggler player uses the temporary directories on each platform for storage of configuration files and a copy of the application while it is running. If the program were halted due to some catastrophic event, the application file and configuration files might be left in a different directory that the user does not know about and may not know how to clean. Since the user does not necessarily know what files are being used, the user may not even know what these files are, should (s)he even come across them. Using the temporary directory assures that these files are in a specific place that users will typically not come across. Also, these temporary directories are cleaned out by the operating system, so if the files should be left in the temporary directory, it will be cleaned out automatically. Across different operating systems, this is a common directory that should be available to any user on any operating system.

3.7 Security

To keep the user from doing something undesirable, the VR Juggler player implements application level security that is set up in configuration files prior to being run. These configuration files can lock out functionality and prevent the user from running applications on hardware to which they are not allowed, prevent the user from deleting files from the reservoir or compressed files pool, or prevent the user from saving files into those same directories.

3.7.1 Layered Security

Security in the VR Juggler player comes is dual-layered. Two configuration files are set up when the program is installed and can be edited if they need to be changed later. They are in locations that are known to the player. The first is the administrative

configuration file. This is stored in the same directory as the player application. When multiple people are using the same application executable on a network, this directory can have read access, execute access, but lock out write access and still function properly. The administrator will lock out whatever functionality they wish without the users being able to alter the configuration. The administrative configuration is read into the player first. It establishes default settings for the player and tells the application which of these settings can be changed.

Following the administrative configuration file, the user administrative configuration file is read. Only settings that the administrative configuration file has allowed can be changed. This can keep the user from pointing at the wrong directories or setting up bad configuration strings. This file is kept in the directory to which `VJ_BASE_DIR` points, which must have read, write, and execute access to the user.

If either of these configuration files are omitted or a setting not specified in both configuration files, the default value is to allow the user to change or use the functionality.

3.7.2 Lock Functionality

Each of the options that the player offers can be disabled for or by the user. The administrative configuration file is checked first for functionality to be disabled. As mentioned previously, the administrator would set up this configuration file to lock out all the things that (s)he does not want the user to change. For example, if the users were sharing a reservoir, and the administrator does not want any one user to be able to purge this, (s)he could disable that button by changing the configuration file. Because of the order that the configuration files are read, the administrator can lock out any or all functionality and know that the user cannot turn it back on or unlock it.

3.7.3 Files' Security

The security of the actual files, such as the applications, the versions of VR Juggler, or the configuration files, is not handled or checked in the program. Anyone that has access to a file is allowed to run or use it. This makes the security of those files dependent on the OS. In Microsoft Windows, if you can see it, you can use it. With a Unix system, you have to make sure you have the correct permissions. Unfortunately, Unix can't stop someone from overwriting a file in a directory to which they have permission. So a potentially corrupt file could be put into a directory if an FTP transfer is interrupted or halted. There should not be a problem with overwriting a file or directory that is corrupt as it should have already been checked for the version.

CHAPTER 4. RESULTS

4.1 Various Platforms

The three platforms that were the focus of study for this thesis were IRIX on an SGI, Linux on an x86 machine, and Windows. The VR Juggler Player was tested on each of these platforms to make sure that it ran properly. Although this list is not exhaustive of all the platforms used by VR Juggler users, it is sufficient to get a sense of how well the player will run on other similar *nix operating systems. Windows XP was not tested.

4.2 Testing

Testing of the VR Juggler player was done on multiple operating systems and computers. The native Windows version was of course tested on Windows and ran successfully. The files were transferred properly to the temporary directory, the application was properly handled, and the `VJ_BASE_DIR` was filled with the proper version of VR Juggler. Purging directories was easy and fast. The VR Juggler versions were stored according to the user's preference. Figure 4.1 shows the Windows version of the VR Juggler player in use.

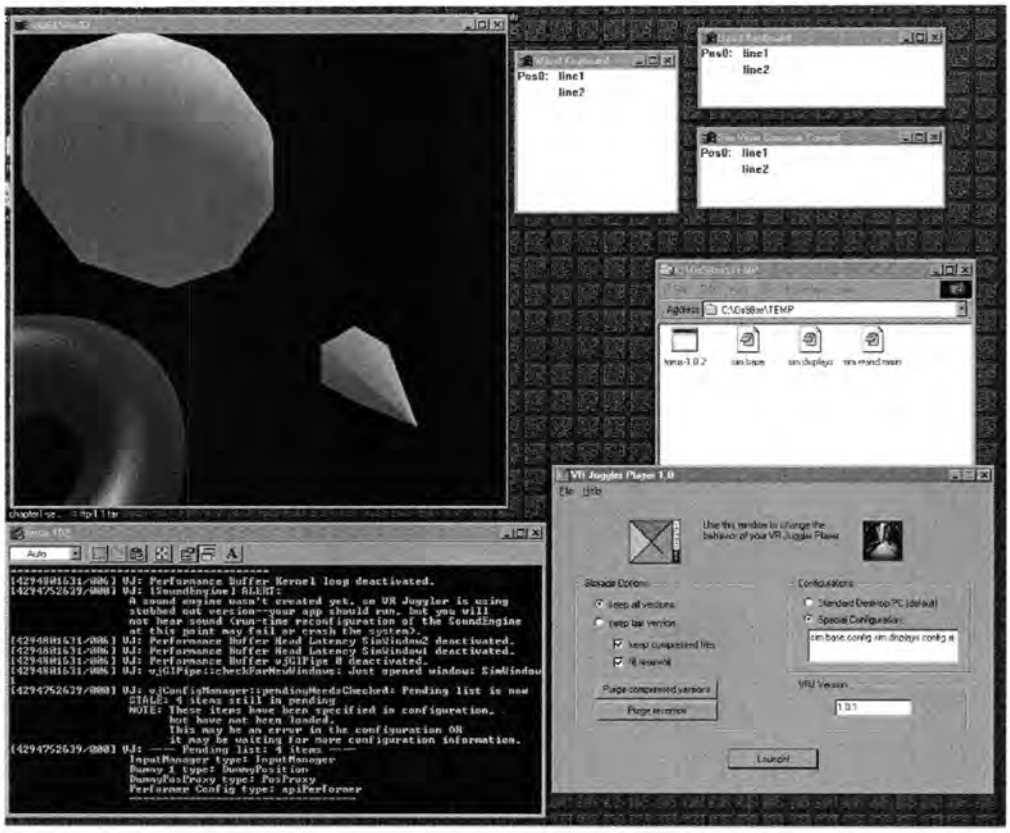

Figure 4.1: Native Windows version of VR Juggler Player in Use

4.3 Refining the Design of the VR Juggler Player

The design of the VR Juggler Player has been refined in several iterations as its capabilities and usability were tested. The first version, 0.1, is shown in figure 4.2 below. Readers will see that it has a lot of functionality, and could work very well. However, upon testing there were a couple important of features that could be added to make the player more usable. In version 0.1, users only had the option to keep the compressed versions of VR Juggler. It was assumed that they would want to keep all the versions in the reservoir if they could. In version 1.0, a checkbox was added below the existing one to allow users to decide if they wanted to fill the reservoir with uncompressed versions of VR Juggler. A purge button for the VR Juggler versions was also included in version 0.1

of the player. This button was originally going to purge both the compressed and uncompressed versions of VR Juggler with the same button. This could be undesirable. A user could want to clean up all the compressed versions of VR Juggler, while leaving the reservoir intact for fast file copying. Or the user could want to reclaim all the disk space that all the old versions of VR Juggler were using in the reservoir, but keep the compressed versions so that in case the version was again needed, it wouldn't have to be downloaded again. To accommodate both of these possibilities, a button for each was added and the old one removed. In version 1.0, there is now a button to purge the reservoir, and a button to purge the compressed files. While a third button could be added to purge both at the same time, it does not seem necessary as saving one button click is not much more useful, especially when the user will already be clicking in that area. Some readers will notice that the `OK, Apply, and Cancel` buttons were removed between versions 0.1 and 1.0 of the player. These did not seem applicable for the application. The radio button to keep the most used version of VR Juggler was also removed because there was not a good place to store this value, and instead of creating a place to store it, the option was simply eliminated.

When the player was ported to Java, there were no functional changes made to it. Running VR Juggler on *nix machines adds one step that the Windows machines didn't have, but does not change the application's functionality. The VR Juggler Player downloads the VR Juggler source for *nix and compiles it for the user. This was done to treat *nix platforms the same. On the next few pages, are figures 4.2 through 4.6, which show the different versions of the VR Juggler Player.

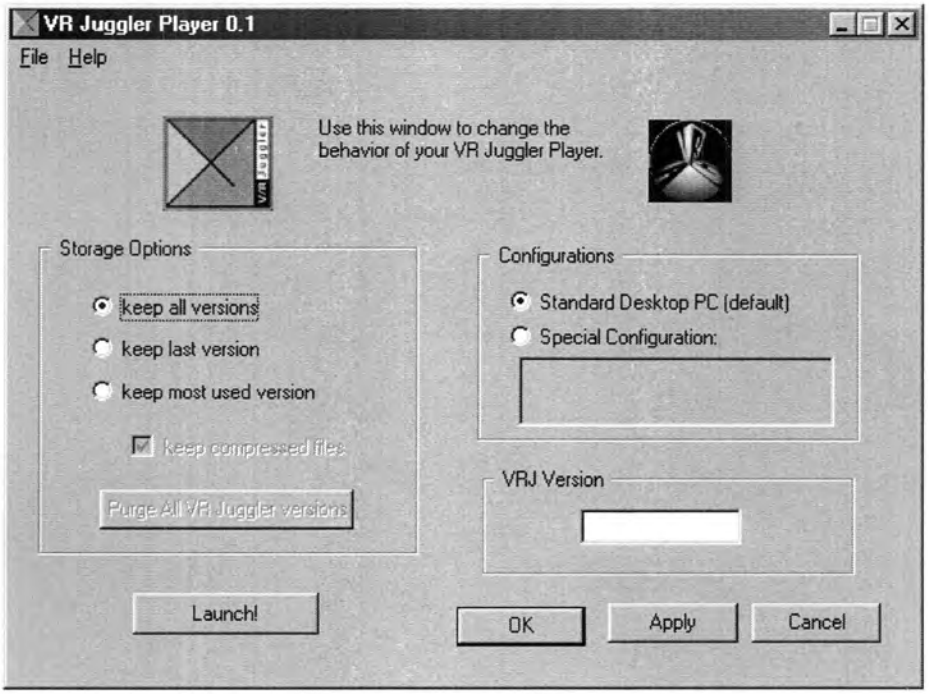

Figure 4.2: Windows Native Code VR Juggler Player, Version 0.1

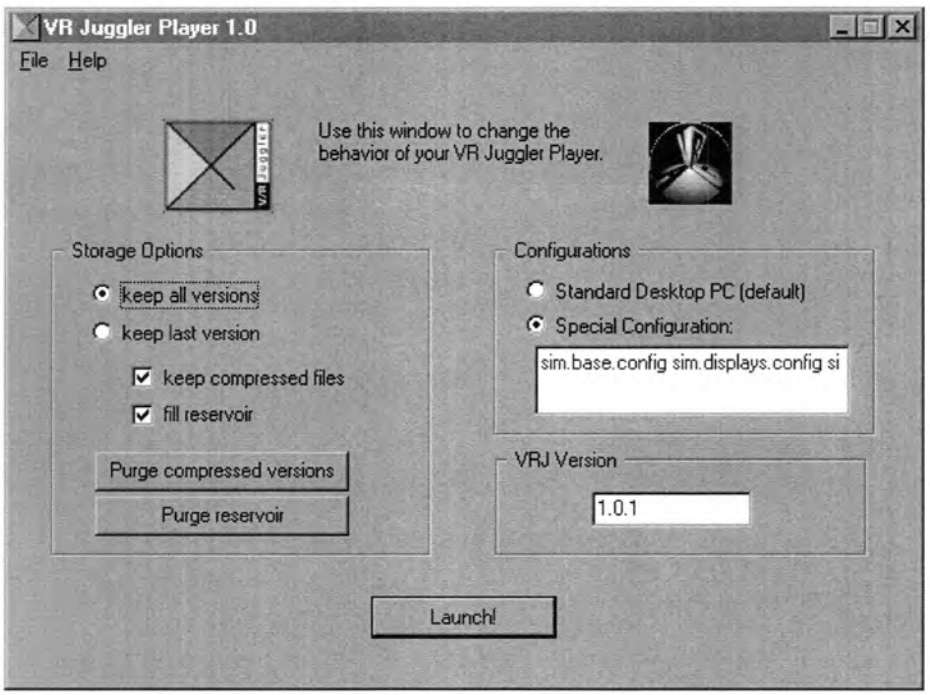

Figure 4.3: Windows Native Code VR Juggler Player, Version 1.0

Figure 4.4: Java Version on SGI machine running IRIX

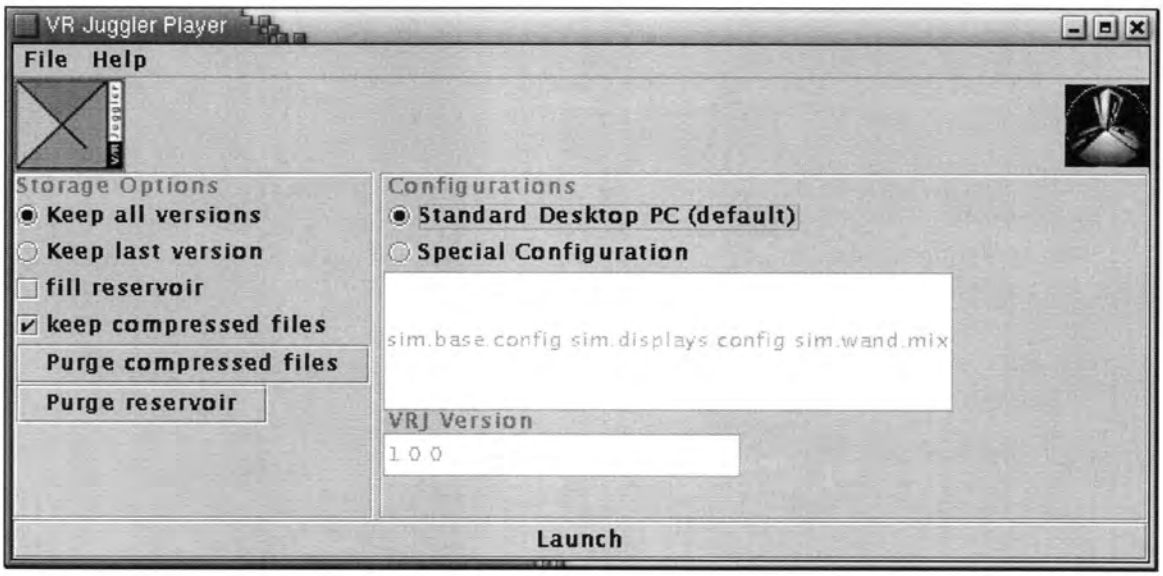

Figure 4.5: Java Version on Linux x86

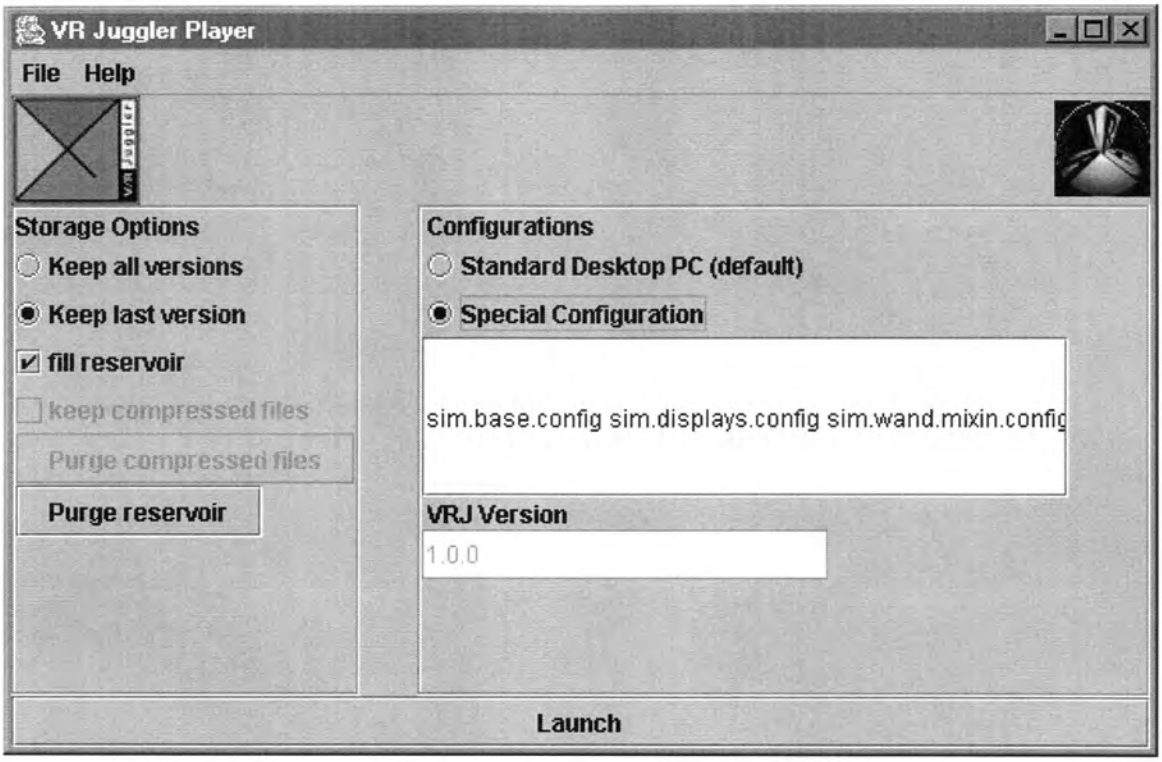

Figure 4.6: Java Version on Windows

CHAPTER 5. DISCUSSION

5.1 Native Coding vs. Portability

VR Juggler is intended to be a cross-platform virtual reality tool. As such, the player should have such goals as well to be aligned. Originally, the intent for the player was to create separate front ends for the two major operating systems, Windows and *nix, then write the functions for each that were platform-specific. This, however, proved to be difficult as most of the work that the VR Juggler player does directly relates to the platform on which it resides. The first versions, 0.1 and 1.0, of the player were written in native code for Windows using the Windows API. This version completely lacked portability.

After the Windows application was written and tested, it was then ported to Java because of the platform independent nature of the language. This makes the program far more portable as Java programs can be run on many more platforms without the need to be recoded, whereas the Windows native would have to be completely recreated for *nix machines.

5.1.1 Java

In the interest of maintaining a single source from which all the versions of the VR Juggler player could be compiled, the Java version was created. This will also solve the problem that would be introduced of selecting a *nix window manager. Java will also allow the file functionality to only be coded, and changed if needed, once. Not only does this work for the *nix side of operating systems, but also for Windows. However, with the release of Windows XP [11], Microsoft has begun to create a rift between its operating system and Java, specifically the Java Virtual Machine, which is what interprets the Java bytecode. This action by Microsoft is done in response to a court

ruling regarding their own virtual machines they had been creating for their operating system. Windows XP does not come with a Java Virtual Machine. Virtual Machines can be downloaded for Windows XP from Microsoft and Sun Microsystems on demand, or at the user's request [10]. If the Windows-native code version of the player is abandoned, it will be very important to monitor developments in this area and package software for the player accordingly for forthcoming operating systems.

Unlike the native Windows version of the VR Juggler Player which is compiled directly into machine code, the Java version is compiled into the intermediate Java bytecode. Java programs are compiled into bytecode, which is later interpreted by the Java Virtual Machine on the platform where the program is run [8]. The size of the two programs, Java and Windows native, were compared to see how this difference in compiling affected it. Because of the way Java compiles each class into its own file, the Java version of the VR Juggler player has numerous files that are used, where the Windows version only has one. These Java files are added together in figure 5.1 shown below. Java also has the ability to run a program from a compressed Java archive (jar) file. The file size of this option is also compared in the graph in figure 5.1.

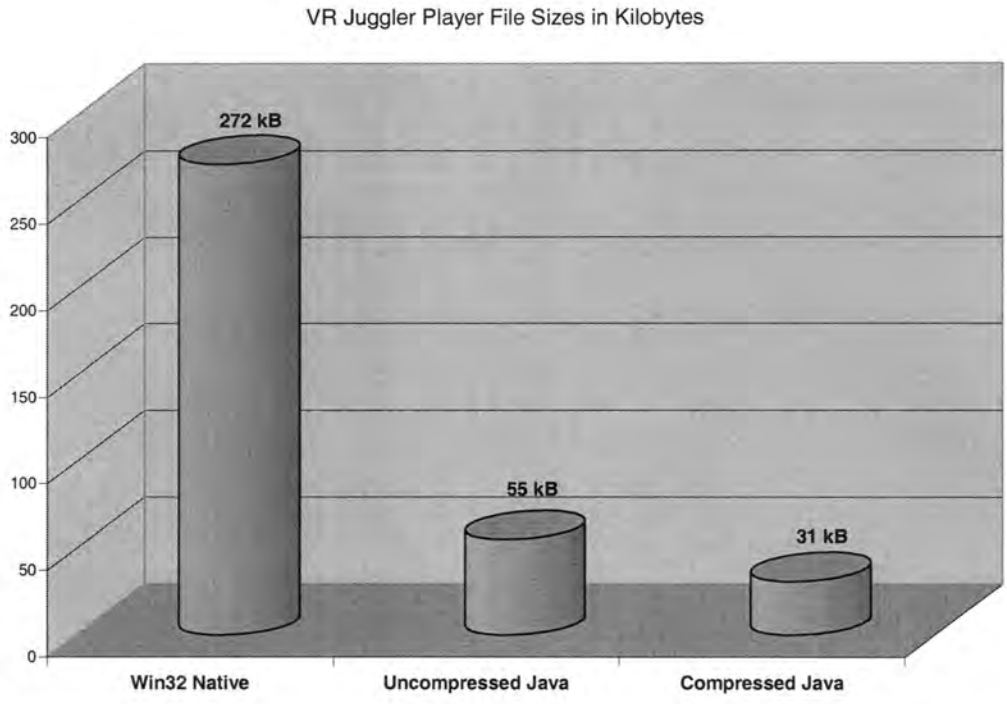

Figure 5.1: File Size Comparison

5.2 Time Trials

Because no specific *nix version was written, only the native Windows version can be compared to the Java version. The time that was compared was the application startup time. This will be from the time that the application is started to the time that the application has read the configuration files and created the final GUI. In this test, readers should expect the Java version to take longer because of the launch of the JVM to interpret the Java program, but the interest in doing this was to quantify how much longer it would take.

These comparisons were made on a 1.2 GHz computer with 256 MB of DDR RAM running Windows 98 Second Edition with an AGP video adapter that had 64 MB of video memory. While the Java version of the VR Juggler Player took a little over nine

times longer to start, this time is still less than one second. Compared to the native code this was much longer, but on its own still a decent startup time.

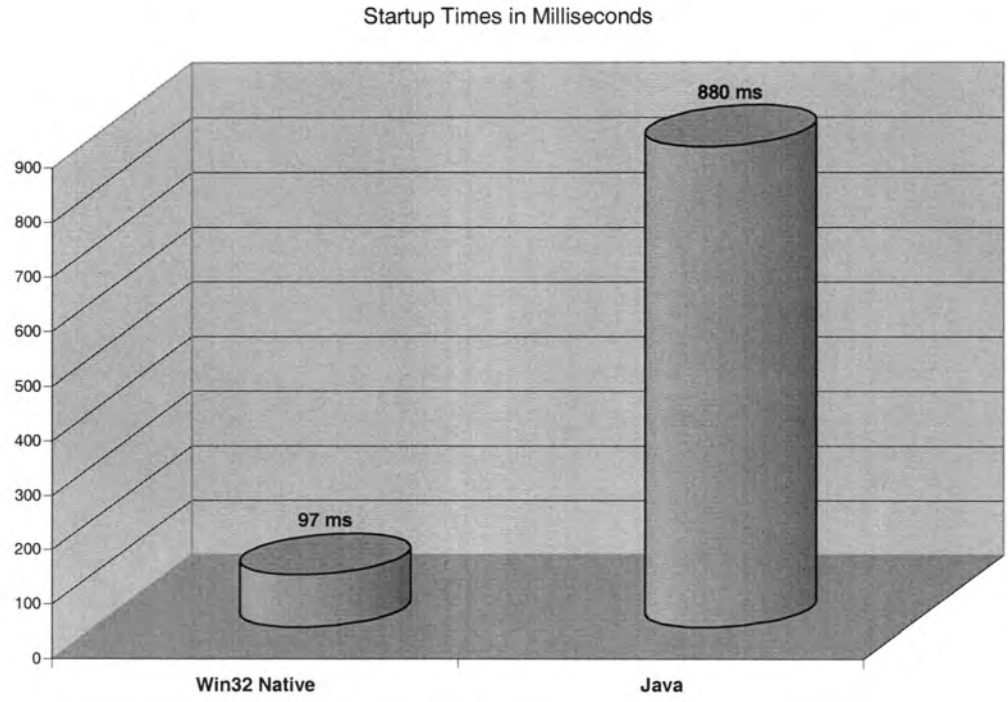

Figure 5.2: Comparison of Startup Times

CHAPTER 6. SUMMARY AND FUTURE WORK

6.1 VR Juggler Easier to Use

By automating the process of installing versions of VR Juggler to the directory pointed to by VJ_BASE_DIR, downloading versions from SourceForge if necessary, and placing specified configuration files with the application, the VR Juggler player makes using VR Juggler applications easier for the end user. Users no longer need to know how to manage VR Juggler themselves, and have the option not to want to know. The VR Juggler player can also expand the number of users that VRAC can reach with their research. Because the user no longer needs to be as technically informed, users in fields and companies who may be interested in virtual reality – without wanting to handle the technical aspects – may be able to enter this realm.

6.2 Abstracts Application from User

By masking the VR Juggler applications as data files as described below, the user doesn't need to know that these are actually programs that are being run on the computer. As far as the user is concerned, and for all practical purposes, these are data files, similar to a text document or an .mp3 music file. The only thing a user actually needs to know about the file is what version of VR Juggler with which to run it and which configuration files will specify the input and output (though the user does not need to know which are for what).

Users are used to filling out forms and inputting data for a number of graphical applications that they use at home. Many users, however, may not be used to running commands at a command prompt. This use has eroded primarily to those people that have

above average computer skills and the desire to do so. Presently, it is not an unfair assumption that the average user does not know how to use the command line interface to the operating system. Thus, running a VR Juggler application puts them in a foreign world. To be at all useful when running VR Juggler applications, they need to have an idea of how to manipulate the computer from a command prompt. It is easy to imagine that many users would rather not use VR Juggler than learn how to use a command prompt. By creating a graphical interface for VR Juggler, the player hopes to reach those users that are otherwise easily discouraged from such "techie" uses of a computer and abstract how the application is handled from the user.

6.3 Small Overhead

The VR Juggler player does not take much from an operating system. The player itself is pretty small, and does not require many resources to run. The application itself does not need to be very big, it just needs to be robust and clever. It needs to be able to tell if it can connect to various places and what it can see and do.

Installing the player itself doesn't add any more environment variables as it only uses the one that VR Juggler needs. While these can take very space and resources, adding a lot of them could eventually affect a system. If every program that was installed to a system had one or more environment variable(s) associated with it, the system may soon slow down. This is especially true if the PATH variable continues to be extended as the system would, on average, need to search more of the disk space to find its program.

The VR Juggler player is designed to be able to take advantage of networked systems by using shared disk space. In a Microsoft Windows environment, these shared spaces would be mapped as drives, and in a *nix system, they would be mounted onto the local file system. By sharing this space, many users running the VR Juggler player can

share one pool of compressed and uncompressed versions of VR Juggler, and any mixture therein.

6.4 MIME types Easier

Establishing a new MIME type in an operating system makes using file types a lot easier for the user. This MIME type would associate an icon with the files of type .vrj, thus masking the fact that they are executable to the user. By doing this, the type is given an icon which can be immediately recognizable to the user. The user knows when they see the specific icon that the file is a VR Juggler application and that clicking on it will open the VR Juggler player and launch the VR Juggler application on their command. This is not only true for desktop and folder icons, but it also holds true for other places that the file extension is used. One example of this would be a link in a web browser. If an application with a .vrj extension is the target for a link in a web browser, the operating system will look up what to do with the file. Upon finding the MIME type, it should present the user with two options: run the program from the web server (save it temporarily on the local system) or save the file to the local system so that it can be used upon completion of the download. By telling the operating system directly how to handle the MIME type, there is no need to tell other applications, such as web browsers, how to handle the .vrj files.

6.5 Future Work

6.5.1 Secure Online Database of Apps

VRAC may wish to be a hub of activity for collaborative, distributed virtual applications. The best way to do this would be done using a website on the Internet. This

website would be the front end to a database of applications that users could access from VRAC's servers. Since this database could potentially store VR Juggler applications that are sensitive in their content or nature, this database would need to be secure. Users would need to have logins and passwords that are encrypted and the transfers of the applications themselves would also need to be encrypted so that a third party observing the transfer would not be able to reconstruct the application. For anonymous, guest, or new users, there would be a section of general interest applications.

In the case of collaborative applications where more than one user could be connected in a virtual world, these could be scheduled on the webpage, so users could join a session in progress. Registered users could even schedule their own events that other people could join. If there are collaborative applications whose content or nature are sensitive, then these would also need to be secured with a login and password. Upon a user's login, (s)he would see all the applications available, the collaborative applications that are in progress (s)he is allowed to join, and future events in which (s)he could participate after downloading the application.

6.5.2 Installation binaries

To make the use of the VR Juggler player, thus VR Juggler itself, even easier, installation binaries should be built to facilitate easy installation of the VR Juggler player, to initialize the configuration files, and to be sure that the environment variables are set up properly.

These installation files would check a number of things on the target computer. The first would be to see if VR Juggler is installed somewhere on the system. This would be done by looking for a VJ_BASE_DIR environment variable. If one exists for the user, it will be assumed that this is where VR Juggler is installed. The installer will also, of course, prompt the user for directories for the reservoir and the compressed versions of

VR Juggler. These could be existing directories, which might even be shared and/or populated, or new ones.

The installer will also set up the MIME/file types on the target system for the user. By doing this, the installation program will create an association for the system between the file extension .vrj and the VR Juggler player. When the user opens a .vrj file, the operating system will look the type up in its table, and find that the VR Juggler player is the application to use to handle this type of file. It will then launch the player and open the .vrj file in it.

The installer should then present the user with two options: network install or local install. The network install would let the user tell the installation program where a copy of the VR Juggler player is already installed on a remote system that it has access to. The installer could then skip the installation of the player on the local machine. The association between the .vrj files and the application would then be pointed to that copy of the player. The only thing that would need to be set up on the local computer would be the reservoir and compressed directories and the VR Juggler base directory and its environment variable if it does not already exist.

The local installation of the VR Juggler player would be a typical on-the-desktop installation where the files would be copied to the computer on which the installation program is being run. The reservoir and compressed directories may, of course, still point away from the computer.

REFERENCES

[1] Bishop, L.; Eberly, D.; Whitted, T.; Finch, M.; Shantz, M. Designing a PC Game Engine. *IEEE Computer Graphics and Applications*, Vol 18, 1, Jan.-Feb. 1998, pp. 46 – 53.

[2] Bradbury, R. *Fahrenheit 451*. Simon & Schuster, New York, 1993.

[3] Brooks, F.P., Jr. What's real about virtual reality? *IEEE Computer Graphics and Applications*, Vol 19, 6, Nov.-Dec. 1999, pp. 16 –27.

[4] Burdea, G., and Coiffet, P. *Virtual Reality Technology*. Wiley-Interscience, New York, 1994.

[5] Christal, M. *QuickTime Virtual Reality for Educators and Just Plain Folks*. http://www.edb.utexas.edu/teachnet/QTVR/Index.html#About (date accessed: 6 May 2002).

[6] Cruz-Neira, C., Sandin, D., and DeFanti, T. Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE. *Computer Graphics*, August 1993, pp. 135-142.

[7] Cruz-Neira, C. Course #14: Applied Virtual Reality, ACM SIGGRAPH conference, Orlando Fl, July 1998.

[8] Lewis, J. and Loftus, W. *Java Software Solutions: Foundations of Program Design*. Addison Wesley Longman, Inc, Massachusetts, 1998.

[9] Liu, S. Bridging Immersive Application Development with the World Wide Web. 2001.

[10] Microsoft. *Open Letter to Sun*. http://www.microsoft.com/java/issues/openletter.htm (date accessed: 6 May 2002).

[11] Microsoft. *Microsoft Technologies for Java.* http://www.microsoft.com/java/ (date accessed: 6 May 2002).

[12] Oliver, J., Vance, J., Luecke, G. Cruz-Neira, C. Virtual Prototyping for Concurrent Engineering. International Immersive Projection Technology Workshop, July 1997, pp. 51 – 57.

[13] VR Juggler. *The Virtual Reality Platform.* http://www.vrjuggler.org/frontpage.html (date accessed: 6 May 2002).