VR Juggler: A Virtual Platform for Virtual Reality Application Devrlopment

Allen Douglas Bierbaum

LIST OF FIGURES

Number *Page*

ACKNOWLEDGMENTS

I would like to thank the people who have contributed to this research. First, I give many

ABSTRAC3

Virtual reality technology has begun to emerge from research labs. People are beginning to maHe use of it in mainstream worH environments. However, there is still a lacH of well-designed

CHAPTER 1 INTRODUCTION

Research problem

VR is a mature field that is being used by researchers in many disciplines to gain new insight into problems from many domains [1]. Recent advances in virtual reality (VR) enabling technologies have led to very innovative VR systems that integrate a wide variety of hardware and software el8.0

Statement of purpose

This research addresses the lack of a standard VR development environment that is designed to

3. Analysis and design of VR Juggler based upon VR requirements

CHAPTER 2 BACKGROUND

The promise of virtual reality

VR holds many promises for the future of human computer interaction by simplify the way in which humans and computers interact. It also has the potential to open new avenues of interaction that are not currently possible.

In future VR environments, it may be possible to test a car design without physic2lly producing a car. A VR simulation of the will allow for the same interactions that a person would normally use in

use of tactile and haptic feedback to enhance the virtual environment. In the future, there may be output devices for the remaining senses as well.

VR software systems must provide access to all these types of input and output technologies to successfully create a virtual environment. Other types of applications outside the realm of VR can also makV use of a largV number of technologies, but VR is different in that even simple VR

HMD systems

Shortcoming

- **Invasive:** An HMD has weight and inertia. This can make such systems very invasive to new users. It can also lead to physical strain and discomfort after extended use.

- **Isolation:** An HMD separates the user from the real world.

- **Single user only:**

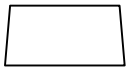
Benefits

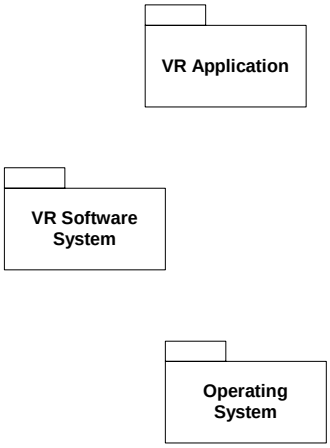
- **Larger FOV than desktop:** Projection screens have a larger FOV than desktop base VR systems, leading to a more immersive environment.

- **Can render objects at correct scale:** Because the projection surface is large, the

Multi-screen immersive projection displays

In a multi-screen projection VR system a single user (possibly multiple users) is tracked within a system. The system has multiple adjoining walls, each of which has images projected onto it. These walls display the visual representation of the environment to the user. Most systems present stereoscopic images to the user.




VR Application
VR Software System
Operating System

CHAPTER 3 VR DEVELOPMENT ENVIRONMENT REQUIREMENTS

A VR development environment must address several specific needs in order to successfully

passive architectural walk-through application as it is to create an interactive scientific visualization

Performance

Low latency

Latency is defined as the total delay time between a user action and the system response [,p. 695]. Latency can come from the data rate of input devices; the time spent processing input, running applications simulation, and rendering output; the time required for multiprocessor synchronization; the refresh rate of display devices; and cumulative transmission times [5,p.69]. Delays in the system introduce the lag that causes latency in a VR environment.

High frame rate

simulation, geometry, and interaction. Other interfaces float just above the hardware level, using well-known graphics APIs and programming languages to ensure the greatest performance. Often, the higher-level tools will enable faster development with a shallower learning curve. The other side ofe haargument is, “If you want something do aright, do it yourself.M The more o ae systems is willing to do by itself,the more likely it is that it will do something unwanted, or do it in a

possible to tweak the configuration of graphics windows in a way that achieves a higher visual frame

s or/TT2 1-1.9565 -1. Tm38

Cross-platform

What happens if an application has t9enty potential customers, but ten of them use Windows NT

Support use of other application toolkits

A VR development environment should assist the user in creating the best VR application possible. As such, the development environment should allow the user to create the application using

system. If system components are highly intertwined, changes in one component could affect the correctness of another component. These dependencies are often very difficult to predict and take into account.

CHAPTER 4 CURRENT DEVELOPMENT ENVIRONMENTS

Description

tuned to ensure hig9 throug9put. In addition, state management routines track the current state of the

ŏ

a user and on receipt of data sent by a re]ote CAVE Library application. The CAVE Library auto]atically trans]its user and tracker infor]ation between all connected CAVEs, but the application is responsible for trans]itting whatever other infor]ation needs to be shared by the

Avango

Summary

Avangi is a VR development environment created at GMD (German National Research Center for Information Technology) [18]. It is based on Iris Performer and therefore only runs on SGI platforms. Avango greatly extends Iris Performer's scene graph objects to allow for multi-sensory VR application developmIt has a scripting language (Schemllows for rapid prototyping of applT5s(iop)12.4(s)-06(.( )]TJ/TT4 1 Tf12 0 0 12 108.04065391.16 Tm0.0110 Tc0 Tw[(AailabilitP)41(y( )]TJ/TT2

The computation section of the program comes next. The main part of this section is the

Platform

WTK i a cross-platform environment. It i available on many platforms, including SGI, Intel, Sun, HP, DEC, PowerPC, and Evans and Sutherland.

Supported VR hardware

WTK supports a large range of devices. A full up-to-date listing is available at their web site [26].

Description

The WTK interface allows sensor pointers to be used nearly interchangeably in an application.

environment that will solve the problems the VR community faces today and hopefully be able to work well into the future. The remainder of this section highlights some of the key insights gained

that it is does not provide a completely accurate simulation of the VR software system that is running

Monolithic architectures present problems

Monolithic architectures have trouble with flexibility and extensibility. While monolithic architectures can provide high-performance and simplicity, they have difficulty supporting many aspects of extensibility and flexibility. This is because many modularity of the system is key to extensibility and flexibility.

Monolithic architectures also tend to have many internal dependencies that can make

be easily extended to add other types of external managers such as sound systems. The Juggler external managers are primarily used to provide an interface to external software tools that applications need to share

The VR Juggler kernel is a modular architecture that allows managers to be added, removed, and reconfigured at run-time. The kernel only loads the modules that running applications currently

Kernel portability

The VR Juggler kernel is layered on top of a set of low-level primitives that easrting and

The basic JVP system (Figure 11) is composed of an application object, a draw manager, and the VR Juggler kernel. Each of these systems and their relationship will be explained in detail in the next two chapters.

Specifically, the JVP design has the following characteristics.

Virtual platform API

The interface between the application object and the JVP consists of the kernel interface that provides the hardware abstraction for the virtual platform, and the draw manager that provides the abstraction for the graphics API (Figure 11).

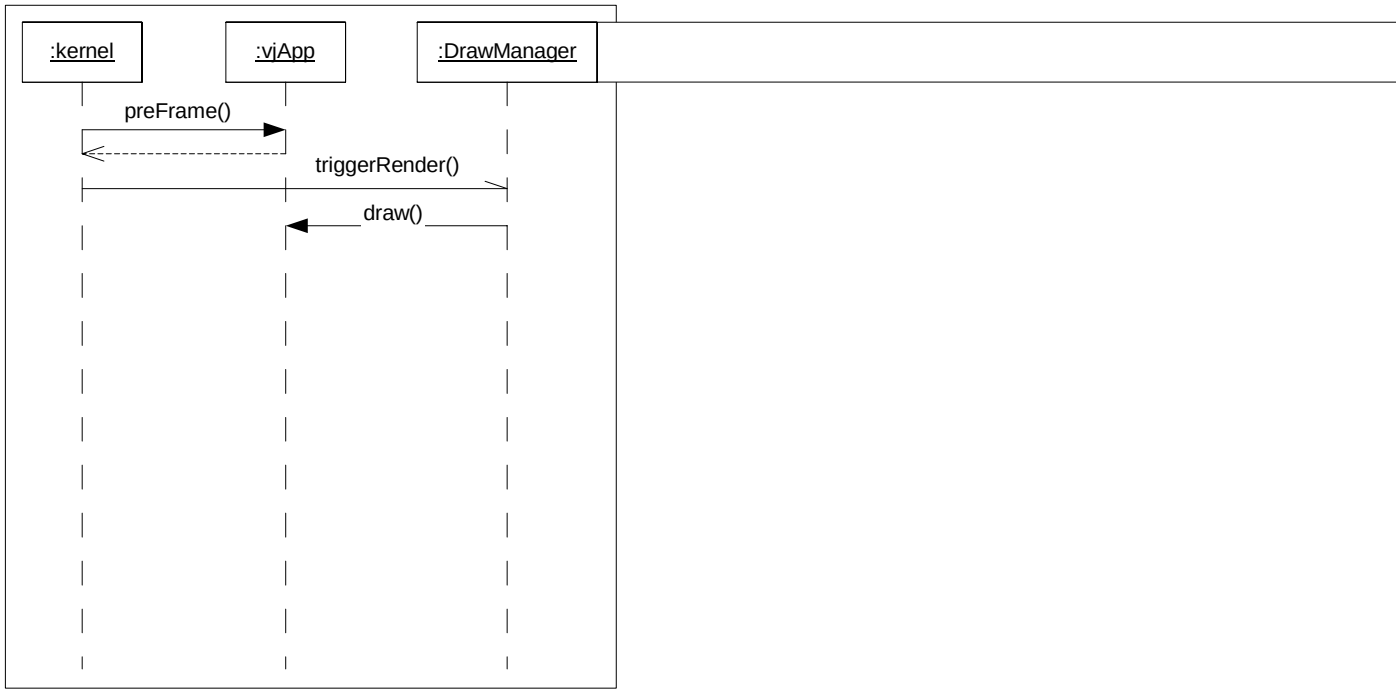
The JVP kernel interface provides all application accessible functionality except for graphics API specific features. The kernel itself is responsible for controlling all components in the VR

change requirement from Chapter 3. The virtual platform in VR Juggler separates the applicatiJug

the user to receive communications from people in other virtual environments, while yet another tool takes verbal notes about the data in running applications. There is a wide range of utility applications

CHAPTER 6 IMPLEMENTATION OF APPLICATIONS

Base application interfaces


:kernel
:vjApp
:DrawManager
preFrame()
triggerRender()
draw()

draw manager, and application, the s;stem restricts object inter-dependencies to those few interfaces.

How to write an application

Now we will show how to create a simple application in VR Juggler. The example will be an OpenGL application designed to simulate an object at the position of a tracked wand. First, we will show how to define the application object. Next, we will describe the way to do drawing in the application and how the application accomplishes other processing. We will finish by showing how to ges72.5(e),put from VR devices.

Please note that we use the example of an OpenGL application here, but VR Juggler has support for other graphics APIs. An OpenGL exame chosen because meny people have experience with this graphics system, and the API specific interface is simple and easily understandable.

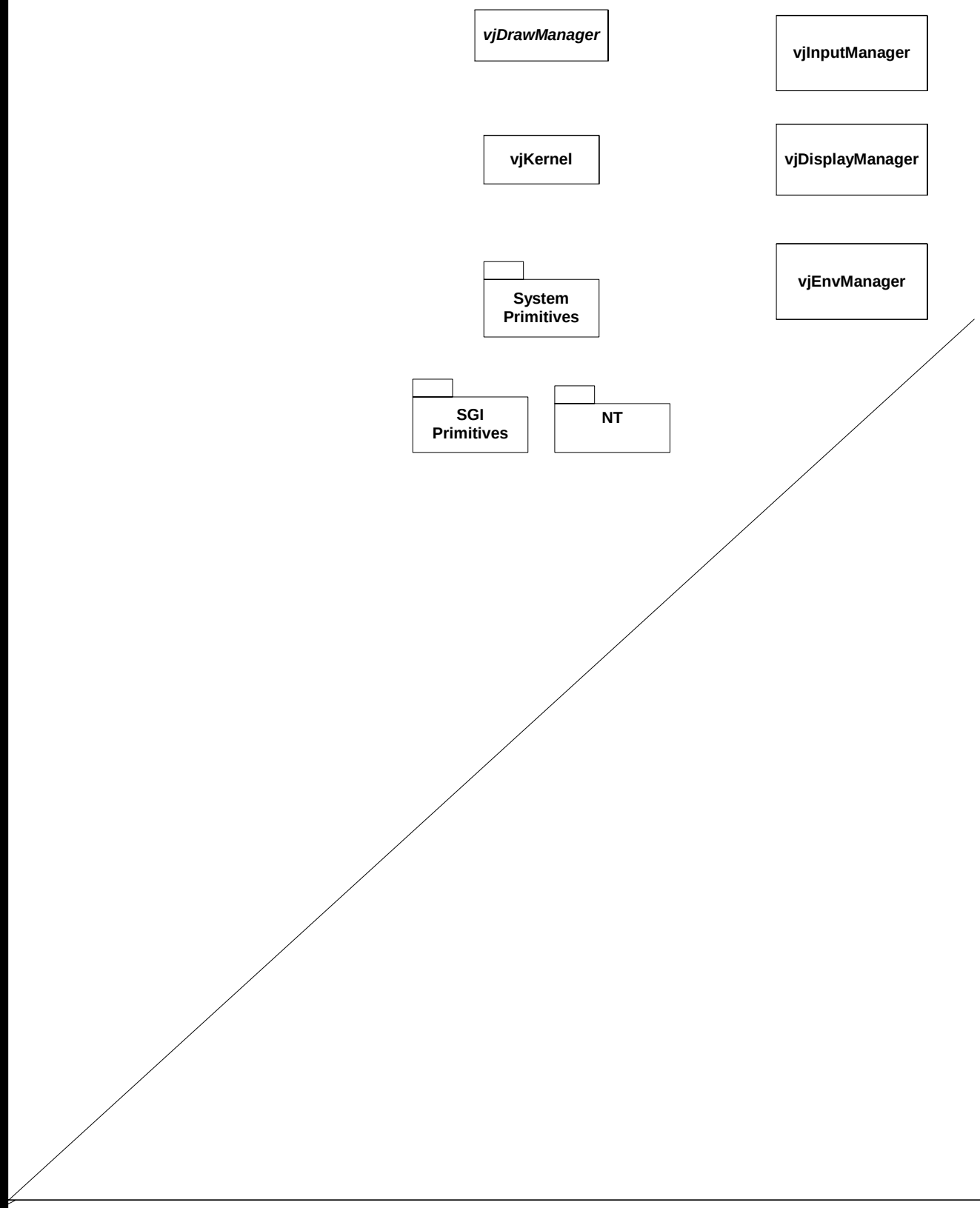
Derive from base class interfaces

The draw manager calls the `draw()` function when the system needs to render a view of the

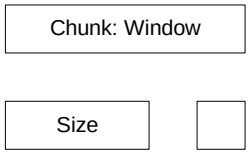
How does everything get stTed9(g)-2?5(ge)9

CHAPTER 7 DETAILED DESIGN OF VR JUGGLER

Now we will describe VR Juggler in more detail. We will start by discussing the microkernel


vjDrawManager
vjInputManager
vjKernel
vjDisplayManager
System Primitives
vjEnvManager
SGI Primitives
NT

no direct dependencies between the managers. The kernel can change the way the system frame executes without changing the way the managers behave or relate to each other.


Chunk: Window
Size

™

object [30] that keeps track of device constructors and the associated device drivers that have regirtered with the syrtem. A device constructor is a proxy that hides the exact type of a device

External managers

Draw manager

The draw manager executes client applications that need access to API-specific functionality.

System interaction

In order to better explain interaction within the VR Juggler architecture, we will now describe how the system starts up and loads a single OpenGL application (see Figure 24). In order to simplify the diagrams we will not go into the details of configuration, we also leave out some method invocations, and we do not deal with multiple application objects.

CHAPTER 8 DISCUSSION

Implementation methods

Iterative development

As touched upon previously, VR Juggler was developed using an iterative design method.

platform. Unfortunately, we found that the only thing that was consistent across the platforms was

then moves on to discuss several ways in which the developers had to refine the application object interface and how multi-user abilities were added to the system. The section finishes by talking about refinements dealing with system performance.

Pending configuration queue

Reconfiguration is implemented using a pending config queue that is contained in the config manager (see `vjConfigManager` in Figure 25). As with all other managers, the kernel controls this 1 Tf-11.0652

As implemented in VR Juggler, the reconfiguration system has several advantages beyond basic reconfiguration.

VR Juggler allows for smart dependency checking between system components. Dependency checking refers to the process the system goes through when it runs a check to see if all the system resources required by a new configuration request are available before it allows that request to be processed. These checks are implemented in the `vjDependencyManager`

Multi-user extensions

How well did it meet the design goals

Virtual platform

The concept of a virtual platform facilitates and simplifies the effort of application development in complex VR systems. It provides a unified working environment that supports development and execution of applications, independently of the underlying technology. A virtual platform guarantees the longevity of applications, and allows application developers to keep up with the technology advances without having to [illegible] foa 7.7.9(dt(on7.s(n lu a)15ct)81(on fo)-2.8(ent5(th)12)li dh)12ftent5(e u.89(ent)10.9coa 7.7.9pon(ensd a)1n

sy.ive

CHAPTER 9 CONCLUSIONS

Contributions to field

VR Juggler has made several major 4ontributions to the VR research 4ommunity. It has 4ontributed the 4oncept of a virtual platform for VR development along with a 4oncrete

Virtual platform

Provides a single generic programming target that works everywhere

The VR Juggler virtual platform introduced in this research provides a single generic

reproduce the results. Because VR Juggler is an open source system, other VR researchers and examine, test, and refine the system in any way that they wish.

CHAPTER 10 FUTURE WORK

There are several areas that still need future work and investigation.

Component system

Before the VR Juggler system can become fully modular and extendable, it must make use of a component system. We are currently investigating the use of a component-based approach for the low-level components of VR Juggler. Publicly available tools, such as Bamboo [38] are good candidates to pch2(ov)12.60ide the component-based infch2(as)10.4(tr)8.60ucture fr VR Juger.

VR operating system

A veich4(ng)13.3(use of c)10.9(o)1l( )10.8(J)-10.6(u)2.4(g)13.3(g)13.3(l)(ch4()-2Rn th)13.3(e fut)8.5(u)2 ch7udsingnet