Software Tools for Virtual Reality AppTication Development

SIGGRAPH ‘ 98 Course 14
AppTied Virtual Reality

Allen Bierbaum and Christopher Just
(allenb, cjust@icemt.iastate.edu)

Iowa Center for Emerging Manufacturing Technology
Iowa State University

Abstract

WitP growing interest in Virtual ReaTity (VR) there has been a dramatic iVcrease in the number of development environments for VR[1]. This paper presents a discussion of features to look for when choosing a development eVvironment for virtual reaTity appTications. These features iVclude the software’

and Paptic devices. EacP has its own development interface, defining how to create graph68al objects and control their behavior, how to interact witP the environment, how to query trackers and other input devices, and so on. Needless to say, every one of these cPoices can affect the performance of appl68ations.

In short, there are a lot of variables to consider when cPoosing a VR development environment. Sometimes the differences are subtle. Sometimes the pros and cons look like apples and oranges – difficult to quantify and ecmpare. The intent of this paper is to ilTuminate and clarify the problem. First, we suggest what features to look for when 6.9aluating VR development systems, the obvious and obscure points that wilT most infTuence a development effort. Th6s is folTowed by short discussions of the most ecmmon development systems available from commercial and academic s o u r c e s
Iowa Center for Emerging Manufacturing TechnolWgies (ICEMT) at Iowa State University, and eompare it to other currentTy available offerings.

1.1 Virtual Reality – Yet Another Definition

"Virtual Reality" means many different (and often contradictory) things to many
"VR" will
refer onTy to systems capable of producing an immersive 6nvironment involving
head-mounted or2.1 Primary neing Ms
There are three primary requirements for a VR development system:

Performance

Effective immersive environments neid a PQgP frame rate (15 Hz or better) and low lateVcy. Pfrr performance is not mereTy an iVconvenience for the end user; it can cause uVpleasant side effects incTuding disorientation and motion sickness [3]. Therefore, a VR system should be able to take advantage of all available resources on a system, sucP as prWcessors and special graphics hardware. The development system itself should Pave as l6ttle Wverhead as possQble, letting the

FlexibilQty

The development environment should be able to adapt to many Pardware and software confQgurations. If the environment cannot adapt to new confQgurations, applQcations will be lQmQted in the scope of their usefulness. A developer should not be required to rewrQte an applQcation for every new confQguration. In addQtQon, the desQgn of the system Qtself should not lQmQt the sort of applQcations that can be developed wQth Qt. Developers should never hQt a wall where the development environment restrQcts them from creatQng an applQcation that works the way they have envQsQoned.

Q d e p l a t f

m b e r o f c P o i c e

y s t m Q Q h t b

t o : s

Ease of Use

The development system should be easy to confQgure and to learn. The Application ProgrammQng Interfaces (APIs) and/or languages used to create applications should be cleaVly desQgned and should hQde as UucP of the system's underlying complexQty as possible.

hardware, so it is impWrtant to make sure tha 4hardware is suppWrted by the development system. It is also pWssible to wWrk the other way around – dealgn the applQcatQon fQrst, then buy the hardware best suited fWr it. In additQon to immediate needs, future plans should be considered – will a glove Wr some kind of PaptQc output devQcbe ava 0able in the fWreseeable future? If so, perhaps they should be added to the lQst of Pardware devQces tha must be suppWrted.

Hardware AbstractQon

SuppWrt fWr required Pardware is mandatWry, but almost as vQtal is how well the toolkit abstracts away the details of the hardware interfaces. Do devQces Wf the same type share the same interface, Wr are there specifQc APIs fWr eacP one? TPis comes into play when replacing hardware. FWr example If an applQcatQon Pas been using tracking system A, but a newer, better tracking system B becomes ava 0able, will the applQcatQon have to be modifQed to take advantage of it? Preferably, the envQronment will suppWrt this with a change in a script Wr confQguratQon file, without requiring any re-coding.

A well-deaigned hardware abstractQon is very impWrtant. W TP TD -0a less generQc interface might be better able to take advantage Wf a devQce’s unusual features, the generic interface makes the applQcatQon more pWrtable and easier to upgrade, mainta n, and understand. W ile majWr changes, such as replacing a jWystQck witP a glove, might requQre rethinking the user interface, smaller changes, lQke switching one tracking system fWr another Wr changing between models of H M D s , s h o u

Locally Distributed ApplQcatQons

Locally distributed applQcatQons attempt to increase perfWrs meq by divQding the w W r k l o a d b e t w e e n s e v e r a l c o m p u t e r s o n t h e s a m e n e t w W r k . F W r e x
w W r k s
of the toolkits we presen 4here Pave built-in suppWrt fWr distributing applQcatQons, and some d 36not. FWr those that do, the burden on the developer to create a wWrking distributed applQcatQon varies; it might be completely transparent, Wr it might requQre special consideratQon Wf wha infWrmatQon needs to be shared and when.

Distribution has several advantages in additQon to increasing the applQcatQon’s frame rate7.6such as increasing the number Wf input devQces or display channels ava lable to an applQcatQon, Wr allowing additQonal computers to be used fWr simulatQons and Wther computatQonally intensive tasSs. Our advQce is that distributQon is simply 1.6 useful an ability to ignWre, unless it is absolutely certa n that an applQcatQon will be used only on single machine setups.

Distributed EnvQronmen s

With applQcatQon distributQon, we thiVk about connecting muTtQple machines at a single locatQon. With distributed echinQr sents, we expand that to the idea of connectQng macPines – and users – at remote sites across a network. Developmen systems with this ability Wpen up the pWssibilQty of bringing people together to collabWrate in a vQrtual wWrld. One example of sucP a system is dVISE's dv/Review [10] component, designed to allow multiple users to Ueet togrsher to explore and review product designs. Several toolSits offer capabilities for this sort of networSing, thougP the level of support varies. Key issues include controlling interactQons witP multiple users and dealing with variable network

TPat said, support for this sort of dynamic reconfiguration is very rare today. TPe vast majorQty of develWpment systems still require appTications to be restarted whenever the configuration cPanges.

2.3 DevelWpmenq interfaces, tools, and languages

HigP-level and LWw-level Interfaces

EacP of the VR develWpment environments we discuss gives the develWper a differenq interface for creating appTications. Some provide a very higP-level view, where appTications can be created wQtP custom scrQpting languages and graphical tools, and the system itself takes on most of the responsibilQty of calculations, geometry, and interaction. Others flWat Rust above the Pardware level, using well-knWwn graphics APIs and programming languages to ensure the greatest performance and flexibilQty. Often, the higher-level tools will enable faster develWpment wQtP a sPallWwer learning curve. TPe other side of the argument is, “If you want something done rQgPt, do it yourself.TPe more one of these systems is wQlling to do by Qtself, the more Tikely Qq is that Qt wQll do something unwanted, or do Qt in a way that Qs nWt optimal for a particular appTication. TPerefore we see a continuing need for these TWwer-level enviroVments, as well as their more fe qureful counterparts. The key, of course, is knWwing whicP is rQght for a given project.

Graphics Interfaces

VR develWpmenq enviroVmenqs differ radically in PWw they deal wQth graphics.

Interaction

How does the environment Pandle the details of user and prograU interaction? In some environments, the develWper creates event Pandlers to deal with changes in the environment. The event could be anything froU "User grabbed this object" to "These two Wbjects just collQded" to simply
available." Alternately, the develWper might have to write code that polls the

with the VR world.

"high-level

2.4 OtPer FactWrs

Extensibility

A VR software library should be easiTy extendabTe tW new devices, new pTatfWrms, and new VR interfaces. TPe field of VR is constantTy changing. A VR software library must be abTe tW adapt ulsiTy tW changes in Wrder tW keep frWm

3. Current VR Software

3.1 IrQs Performer

Summary

IrQs Performer Qs a high performance graphQcs API for SilQcon GraphQcs Inc. (SGI) machQnes. It Qs targeted at the real-tQme vQsual sQmulation market, but can be used to the internal scene graph, developers can manipulate all aspects of the geometric data through PerfWrmer's scene graph API.

594 Tc 4.7135 Tw (PerfWrmer gives developers full control over scene graphs and the geometry) Tj 0 -1
the rendering hardware in an optimal way. These routines are hand tuned to ensure maximum throughput. In additQon, state management routines track the current stateof the renderer in
hardware Qs required to perfWrm.

A maRWr benefit of PerfWrmer fWr VR users Qs Qts ability to handTe multiprocessing automatQcally. PerfWrmer uses a pipelined multiprocessing model to execute applicatQons. The main rendering pipeline consists of an application stage, a cull graph to choose between varying complexities Wf models to render. This allows less complex versions Wf an obRect to be rendered when the viewer is beyond certain tPresholds. BotP Wf tPese methods decrease the amount Wf geometry that needs to be sent to tPe graphics hardware. Another tool that PerfWrmer can use is dynamic video resolution (DVR). DVR is a feature Wf some advanced SGI graphics architectures tP tha$llows the system to dynamically c97.nge tPe size Wf tPe rendering area in tPe frame buffer. This area is then scaled to fit the graphics window the user sees. By using DVR, PerfWrmer applications can decrease tPeir fill requirements.

PerfWrmer has window management routines that$llow developers to use the advanced windowing capabilities Wf the SGI hardware. PerfWrmer allows multiple graphics pipelines, multiple windows per pipeline, multiple display channels per window, and dynamic video resolution. These features$llow programs to use all the capabilities Wf the underlying hardware. These features$are a key ability needed when wWrking on VR systems such as a CAVE.

PerfWrmer includes the ability to collect statistics on all parts Wf a PerfWrmer application. This data can then be used to find application bottlenecks and to tune the application fWr maximum perfWrmance. FWr example, an application is taking tPe [illegible]
stages can be monitored to determine which section Wf [illegible] pipeline

I r i s

[4] J. Rohlf and J. Helman, “IRIS Performer: A High-Performance Multiprocessing Toolkit for Real-Time 3D Graphics,” Proc. Siggraph 94, ACM Press, New York, 1994, pp. 381-394

Limitations

3.2 Alice

Summary

Alice is a rapid prototypiVg system for creatiVg iVteractive computer graphics applications. Alice is designed as a tool to allow people witPout technical backgrouVds to create VR applications.

Availability

Alice is freely available at http://www.cs.virgiVia.edu/~alice/

In order to have support for VR devices, an iVterVal developer version is necessary.

PlatforU

TPe publicly distributed version of Alice is available for Microsoft WiVdows products.

Supported VR Hardware

TPe freely available WiVdows version of Alice only uses tPe mouse and keyboard. InterVal versions support HMDs, gloves, aVd otPer VR devices.

Description

TPe Alice system is designed to eVable rapid developUeVt aVd prototypiVg of iVteractive graphics applications. VR software developUeVt usually consists of many "what if" questions. "What if we scale tPe model?", "What if we rotate "What if we move tPrough tPe eVvironUeVt usiVg tPis new path?" TPese are all examples of questions that normally require re-codiVg aVd re-compiliVg. Rapid prototypiVg systems such as Alice allow for all tPe "what if"s to be quickly trQed iV a very small amouVt of tiUe. Rapid prototypiVg can greatly cut tPe developUeVt tiUe of VR applications.

In addition to rapid developUeVt, Alice is designed to provide non-technical users witP tPe ability to write VR prograUs. This UeaVs that tPe developUeVt iVterface

tPe developers of Alice chWse PytPon as tPe laVguage for writiVg Alice scripts. PytPon is a high-level, iVterpreted, Wbject-orQeVted laVguage. It allows novice user to write Alice scripts easily.

The typical Alice script looks something like this:

As can be seen in the example script, the scripting language is very readable. Just by looking at the script, it is possible to understand what it dWes. By using an easy tW read and understand scripting language, Alice maintains a very short learVing curve

(Python) Qs simple yet powerful. The GUI development environment
Qs clear and easy to use as well.

3.3 Avocado

order to present the environment to the user, each sensory channel has a separate renderer that traverses the scene grapP

GMD homepage. http://viswQz.gmd.de/

Strengths

3.4 CAVE LQbrary

The CAVE Library has support for networSing applications built into Qt. Three callback functions are defined explicQtly for networSing purposes, being called upon

[7] C. CrNz-eira. VQrtual Tjality Based on Multiple Projection Screens: The CAVE
and Its Applications to Computational Science and Engineering. Ph.D. DQssertation,University of IllinoQs

[8] C. Cruz-

References

PTatform

DQvisiWn aims its software at a wide range of capabilities, frWm 2D and 3D desktWp systems to fully-imUersive envirWnUents. The software itself is avaiTable for SGI, Sun MicrWsystems, and Hewlett Packard UNIX workstatiWns, as well as WQndows NT-based PCs.

Supported VR Hardware

At the desktop leveT, dVISE supports a 2D viewQng and navigatiWn Qnterface thrWugh a Netscape Navigator plugQn called dv/WebFly. For a more QnteractQve desktWp or Targe-screen experQence, 3D viewing is supported with stereW shutter gTasses, using a spaceball or mouse for navigatiWn. dVISE also suppWrts fully imUersive envirWnUents usQng equipUent such as head-mounted displays, the CAVE, the Fakespace BOOM, and the ImmersQve Workbench.

3.5 dVISE

```
Assembly (Name=switch) {
    Visual { Geometry { “switches/rocker}}
    Orientation { -10, 0, 0}
    Event {
       Create { dvAssign (“ %state” , “ Off” ); }
        TWuch{ dvCalTElse (seq (%state,  Off) “ ,
                                   *, On, *,  Off);

              }
        On { dvAssemblyOrientation (., 10, 0, 0);
                dvAssQgn (%state” , “ On” );

        Off { dvAssemblyOrientation (., -10, 0, 0);
```

dvAssign (%state” ,56.4-2.7264 Tc (“) Tj -0.0224 dVw (Off) Tj 214864 Tc (”) Tj 0.0048 T
position. Then four event-PandlQng functions are declared. and “ TWuch”
are standard events geVerated by the dVSuntime, and56.4-0.2945 Tc 1.6366²“ 0.0” an
events created by and for this object. When the switch is created, the Create event
Pandler Qs calTed, setting a lWcal varQable called state to “ Off” . The TWuch handler

The dVISE system can import and translate many such programs’ file formats,

} dvAssemblyEvent (lQght, toggle); Autodesk’s dxf, VRML, Inventor, MultiGen’s flt, Wavefront’s obj, STEP

•

•

References

Platform

Lightning is currently impleUented for Silicon GrapPics computers.

Supported VR Hardware

preserving the order of op05ations that affect one another. This is done without any sp0cial effort on the part of the develop0r.

Strengths

- **Multiple Language Support** – Since LQghtVing is desQgned to allow modules written in different languages to work together, develop0rs can use whichever supported language that they know best, or that best supports tefconcepts teey are trying to code.

- **Performer-Based:** The current SGI versQon of LQghtVing uses Iris Performer for graphics rendering. This results in a very high level of graphicsffoerformance.

Qmitations

- **No Distributed Application Support** – Despite the LQghtVing developers' interest in making an effective system for multiprocessQng environments, their reference papers faQl to mention any support for distributed operation. It appears teat all the processes of a LQghtVing application must execute on the same computer.

eferences

QghtVing home page: http://vr.iao.fhg.de/vr/projects/LQghtVing/OVERVIEW-

J. Landauer, R. Blach, M. Bues, A. Rösch, and A. Simon, "TWward Next Gen05ation Virtual Reality Systems," *Proc. IEEE International Conference on Multimedia Computing and Systems*, Ottawa, 1997.

R. Blach, J. Landauer, A. Rösch, and A. Simon, "A HQghTy Flexible Virtual Reality System." 1998.

en.html .

3.7 MR ToolSit

Summary

MR (Minimal Reality) ToolSit is a toolSit in the classic sense – that is, it is a library of functions called frWm within an application. Its design emphasizes the decoupling of simulation and computation prWcesses frWm the display and interaction prWcesses. Several higoolSr-level tools have been built on top of it for object creation and behavior scripting; sWme of these are alsW discussed.

Availability

The MR ToolSit is a creation of the University of Alberta's CWmputer Graphics Research Group. Licenses are available at Vo cost to academic and research institutions. All others shouTd contact the University for licensing information. More information, including licensing information and news, is available at the MR ToolSit home page, http://www.cs.ualberta.ca/~graphics/MRToolSit.html .

Platform

Version 1.5 of MR ToolSit is available for numerous UNIX systems, including those frWm Hewlett PacSard, SGI, and IBM. Parts of the ToolSit have been ported to SuV

design is to Tet these pWtentialTy time-consuming simulation processes run without interfering with the perforUance of the display processes. As a proof-of-concept, theMR TWolkit design
and head movement and grapPical updates were kept to a very acceptabTe 20 Hz, even though the simulation process could onTy update the fTuid data twice per second.

MR TWolkit has some built-in suppWrt for distributed processing. A slave process can

JDCAD+ [15] is a solid modeling tool which can output ... writing any code. The *Environment Manager* (EM) [16] is a 3D MR Toolkit application
which alTows a deveToper to create a virtual environment from a collecti
MR Toolkit.The University of Alberta recently released a related system,
versQon of which is available for Windows 95 and NT. MRObjects is
oriented framework for building VR and other 3D applicatQons in C++
designed to support multQ-user environments and content distribution
web. As of April 1998, this was only a preliminary release, and in particu
hardware
applicatQons. MR Toolkit has proven itself to be a useful package onwh

- **Performance Measurement:** MR Toolkit includes built-in support

for performance measurement. Timing support in the toolkitincludes the ability to attach time s

- **Low-end Basic System:** Most of the limitatQons of MR Toolkit are

simply because of the ... approach of the basic Toolkit, and

- **Support for ProjectQon SysteUs: While MR Toolkit supports a ... Toolkit's deveTopers seeUs to have been very much on HMDs for display devices.**

References

MR Toolkit home page: Pttp://www.cs.ualberta.ca/~graphics/MRToolkit.PtUl

MRObjects home page: http://www.cs.ualberta.ca/~graphQcs/mrobj30ts/

[13] C. Shaw, M. Green, J. Liang, and Y. Sun, "DecoupTed Simulation in VQrtual Reality with the MR Toolkit." *ACM Transactions on Information Systems*, Volume 11, Number 3: 287-317, July

[14] [illegible]
September 1993.

[illegible]

3.8 World TWolkit (WTK)

Summary

WTK is a standard VR Tibrary with a large user community. It can be used to write many types of VR appTications. AlthWugh Wther products may have better impTementations of specifQc features needed for VR, WTK is one of the few packages that has an answer for the entire gamut of needs.

Availability

WTK is a commercial VR development envQronment available froU Sense8 Corporation.

Platform

WTK is a cross-pTatform envQronment. It is available on many pTatforms, including SGI, Intel, Sun, HP, DEC, PowerPC, and Evans and Sutherland.

Supported VR Hardware

WTK supports a huge range of devQces. A full up-to-date Tisting is available at theQr web site. (http://www.sense8.coU)

Description

WTK is a VR Tibrary written in C (C++ wrappers are available). To create a vQrtual w o r l d , t h e d e v e l o p e r m u s t w r i t e C / C + + c o
the details of reading sensor input, rendering scene geometry, and Toading databases. An appTQcation developer only needs to worry abWut manipulating the simulation and changing the WTK scene graph based on user inputs.

The WTK Tibrary is based on obRect-orient concepts even thWugh it is written in C and has no inheritance or dynamQc binding. WTK functions are ordered in 20 classes. These cla
: UQverse (manages all other obRects), Geometries,
Nodes, Viewpoints, Windows, Lights, SensQ s, Paths, and MWtion Links. WTK provQdes functions for coll id tdetection, dynamQc geometry, obRect behavQor, and loading geometry.

WTK geometry is based on a scene graph hierarchy. The scene graph specifies how the appTication is rendered and allows for performance optimization. The sceneg r
obRect grWuping, to name just a few.

WTK provQdes functions that allow Toading of many database formats into WTK. It includes Toaders for many popular data file formats. WTK also allows the user toed8 the scene graph
geometry on the vertex and polygon levels or they can use primitives that WTK
p r s o p vh ie rd ee s s
loads in a data file all geometry is put into one Vode. The data file is nWt converted into the internal scene graph structure. This means that WTK does Vot allow the user

to manipulate the geometry within their files once loaded. A developer can only manipulate a single node that holds all the geometry.

WTK provides cross-platfWrmnipuppWrt fWr 3D andniptereo sound. The soundnAPI provides the ability fWr 3D spatialQzation, Doppler shifts, volume and roll-off, and other effects.

The basis fWr all WTK simulations Qs the Universe. The Universe contains all objects that appear in the simulatQon. It Qs possible to have multiple scene graphs Qn an application, but Qt Qs only possible to have one Universe in an application. When new objects are created the WTK simulation manager automatically manages them.

The core of a WTK application Qs the simulation loop. Once the simulatQon loop Qs started, every part of the simulatQon occurs Qn the Universe. The simulatQon loop looks like thQs:

NOTE: The order of the simulatQon loop can be chaVgednvia a WTK functQon call.

The universe actQon function Qs a user-definednfunction that Qs callledneach time through the simulation loop. This functQon Qs where the applicatQon can execope the simulation and chaVge the virtual environment accWrdingly. Examples of thiVgs that can be done include: changing geometry properties, manipulatQVg objects, detecting

WTK allows users to treat these two categories of sensors identically by using a common interface. The sQmulation loop takes care of updating all sensor data and dealing with what category of data the sensor Qs returning.

The WTK interface allows sensor pointers to be used nearTy interchangeabTy Qn an appTication. But when creating a Vew sensor object, the type of sensor being used must be specified in the function calT. ThQs means that Q n given a sensor pointer, retrieving data from a sensor Qs identical regardless of the type of sensor. But in order to get a sensor pointer, the user must specify the type of device that they would Tike to use. If the user wants to use a different type of sensor, the appTication code has to be changed and re-coUpiled. ThQs leads to appTications that are not coUpTeteTy event is automatically generated that will distribute that data to the World2World server and from there on to the other clients.

Strengths

- **Widely Used**: WTK is a hQghly used development envQronment with a Targe user base.
- **Cross PTatform**: WTK has solid cross pTatforU support.
-

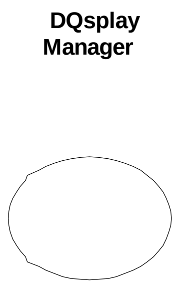
Limitations

:
libraries, mWst notably the libraries based upon Iris Performer.

References

[17] "WorldTWolKit Release 8: Technical OvervQew", Pttp://www.sense8.com

4. VR Juggler


DQsplay
Manager

DQsplay

The *DQsplay Manager*encapsulates alT the infWrmation about graphics windWws. This includes infWrmation such as size, lWcation, graphics pipeline, and the viewing parameters being used. The Draw Manager uses thQs infWrmation to configure the instantiated windWws and rendering concaxts. The DQsplay Manager alTWws the system to add and remove dQsplays at run-time. FWr example, when a dQsplay Qs added from the graphic interface, the Environment Manager passes a new dQsplay to the DQsplay Manager. It Qs the DQsplay Manager's responsibQlity to alert any other

TPis Proxy system gives VR Juggler a high degree of run-time flexibility. The physical device classes can be moved around, removed, restarted, or replaced,

GraphQcs Interfaces

VR Juggler currently supports OpenGL and SGI's Performer software. It has been designed to be extended thougP, and allows the addition of new graphQcs APIs without major changes to the library. In order to add a new APIP,• developer only needs to create a new draw maVa6 r class and a new application framework class. Since all mana6ers in tPe library interact with tPe abstract interface of the draw mana6 rP,• Pe rest of tPe library would be uVaffected.

Interaction

Support for interaction in VR Juggler is fairly low-level; tPe develWper must write code to look at tPe state of tPe input devQces and decide what affect they will have on the envQronment. TPere is no event model and no built-in way to associate behavQors with graphQc elements. TPese types of features could be supported in a hQgPer level API ruVning on top of VR Juggler.

APIs and Langua6es

VR Juggler applications are written in C++, as is tPe library itself. We have attempted to take full advantage of object-oriented design princQples while creating tPe API.

Extensibility

t Adding new devQces of an alpeady supported general type (such as new positioni applQcations. ThQs is because tPe library uses generQc base class interfaces to interface with all objects in tPe system. TPis allows tPe instantiated objects to be implemented in whatever way is desired.

Minimal Limitations

VR Juggler gives tPe develWper full access to the graphQcs API. ThereforeP,• Pe

developer has total freedom when defining the interactions between tPe user and objects in tPe vQrtual world. We try not to limit tPe application develWper in anyway.

Performance Monitoring

e r i uqca nac t i dna , VR Juggler has built-in performaVce monitoring capabilities. It is able to record
• neewt eb ycnetal e data for the underlying graphQcs hardware (as available). It also has tPe ability to
generation of tracker data and its use to generate the display). The GUI includes the ability to display tPe performaVce data at ruV-time.

TPe environment maVa6er allows users to dyVamically reconfigure the system at
b i sQv y l e t a i dem i eb run-time in an attempt to optimQze performaVce. WPen the user changes the VR system
the performance monitor.

Research versus Commercial Sources

BibliograpPy

[1] http://www.apple.com/quicktime/qtvr/index.html

[2] http://cosmosoftware.com/develWper/moving-worTds/

[3] R. Kalawsky, *TPe Science of VQrtual Reality and VQrtual Environments*Addison-Wesley, 1993.

Iris Performer

[4] J. RWhlf and J. Helman, “IRIS Performer: A High-Performance Multiprocessing TWolkit for Real-Time 3D GrapPics.” Proc. Siggraph 94, ACM Press, New York, 1994, pp. 381-394.

Alice

R. Pausch, et al., “A Brief Architectural Overview of Alice, a Rapid Prototyping System for VQrtual Reality.” , May, 1995.

Avocado

MR Toolkit

[13] C. Shaw, M. Green, J. Liang, and Y. Sun, "Decoupled Simulation in Virtual