Interactive Exploration in Virtual Environments

Belleman, R.G.

Publication date
2003

Citation for published version (APA):

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
Chapter 3

The UvA-DRIVE virtual reality system*

"The ultimate display would, of course, be a room within which the computer can control the existence of matter. A chair displayed in such a room would be good enough to sit in. Handcuffs displayed in such a room would be confining, a bullet displayed in such a room will be fatal. With the appropriate programming, such a display could literally be the Wonderland into which Alice walked."


3.1 Introduction

The objective of a Virtual Environment (VE, the environment that is generated by a VR system) in general is to fool a human being into believing that he or she is physically located in a synthetically generated environment by presenting him or her with reactive external stimuli. Of the human sensory system (vision, hearing, smell, touch and taste), the modality that is most often exploited in a VR system is vision because it has the greatest and most immediate impact. High-end VR systems, like the CAVE installed at SARA in 1997, have been successfully applied in numerous fields, ranging from architecture and engineering, to biotechnology, psychology and medicine [5,20,25,50,51].

What all these projects have in common is that they need a VR system that can generate a VE that is so compelling that the user is convinced he is immersed into a new world. To achieve this, the VR system must, apart from presenting the user with sensory stimuli, be able to sense the behaviour of a user so that it can react to

the user's actions. Such an environment then allows the user to interact with the VE, thereby increasing a user's awareness of the world presented around him. It is this combination of presentation and interaction that makes VR so interesting.

Having said that, it is curious that VR has not met wider acceptance than it has up until now. The main reason for this is cost. Not until recently, the only systems that were capable of providing high fidelity VR were large monolithic computer systems with dedicated graphical subsystems that cost millions. However, with currently available commodity off-the-shelf (COTS) hardware and open source software it is now possible to build VR systems that rival and in some cases surpass the capabilities of large commercial VR systems.

This chapter describes a design that has resulted in the construction of, amongst others, the University of Amsterdam Distributed Real-time Interactive Virtual Environment (UvA-DRIVE), a fully functional VR system based on COTS hard- and software. Section 3.2 discusses the requirements of VR systems and the considerations that have to be taken into account while constructing one. Section 3.3 describes a number of design options for a VR architecture. Section 3.4 presents the prototype which has been built and the results of a comparative benchmark. Finally, in section 3.5 we present our conclusions.

### 3.2 Requirements and considerations

The primary objective of this work is to design and build a fully functional VR system that allows users to explore their datasets or to develop prototypes of virtual environments for later use in larger systems, such as a CAVE, without having to use these more expensive resources during development. The considerations we made in the design of the new architecture are the following.

**Multi user**

In most applications, a researcher wants to explore and discuss the results of an experiment with his peers. The display system should therefore allow more than one person to join in the experience. This rules out the use of head-mounted displays (HMD) that essentially provide a single user VR experience (the increase in both complexity and cost make it impractical for every user to wear an HMD). Furthermore, conventional cathode-ray tubes (CRT) monitors are too limited in size; in practice, no CRT monitors larger than approximately 100 cm in diagonal exist. Larger sized LCD panels are available but suffer from low refresh rates which makes them unsuitable for VR displays.

**Minimal effort in porting existing software**

In the past, extensive effort has been put in the development of visualization and VR applications for use on high-end VR systems. An important requirement is, therefore, that the use of this software can be continued on the new architecture with minimal porting efforts.
Processor and graphics performance
The data sets that are currently explored by the users of VR systems range anywhere from low to high volume and complexity. To accommodate the full range, the architecture’s processing power and graphical performance should be able to handle these datasets adequately.

Stereoscopic 3D display
Stereo vision adds a compelling depth-cue over perspective projection by providing the left and right eye with slightly different images, corresponding to the differences the eyes see when looking at objects in the real world. This property, together with the ability to interact with a VE is what makes a VR system a unique tool for exploring large, complex, multi-dimensional data sets.

Head and hand tracking
Head tracking makes an additional depth-cue possible known as “motion parallax”. With this technique, the environment responds naturally to the movements of the user so that one can attain additional information on the shape and size of a 3D structure. Hand tracking allows a VR system to determine the position and orientation of the user’s hand so that interaction with the environment is possible.

Flexibility and scalability
Rapid advances in the semiconductor industry make computing and graphics systems virtually obsolete within a time span of three to five years. To lengthen this time span, the new architecture should consist of components that can be easily replaced as better hardware becomes available.

Cost
For most end-users, project budgets are limited. The solution we are looking for therefore aims at providing a VR architecture that is low cost (both in construction and maintenance) but not at the expense of the requirements mentioned above.

3.3 Design options for a VR architecture
Given the requirements and considerations outlined in the previous section, we have investigated a number of options to construct a VR architecture based on commodity hardware. During our investigation we have considered a number of different designs which will be discussed in this section. The architecture we have selected and built is described in section 3.4.
3.3.1 Multi user stereoscopic display

The display system is that part of the VR system the experimenter looks at and interacts with. In some cases, a quality VR monitor may be quite sufficient as a display system but for multi user use, a monitor is often too small. A convenient method for obtaining a large display area is to use high brightness projectors that project images on large surfaces that can be viewed by multiple users at the same time. In these systems a projector projects images on a screen, mostly on the back so that the user can move freely in front of the screen without occluding the projected images (see Figure 3.1). This chapter will only cover single screen projection systems.

Figure 3.1: Front and back projection systems.

For projection based systems, there are basically two methods to generate stereoscopic images; active and passive stereo systems.

Active stereo

In active stereo systems, a single display device (i.e. a graphics adapter and a projector) is used to generate images for the left and right eye alternately. The display of the images is synchronized with a device that ensures that the users see only the left image in the left eye and the right image in the right eye. The application, the graphics adapter's firmware and the graphics adapter's hardware must support this type of stereo through an interface that signals the end of a frame. This is commonly done using the vertical retrace signal that is generated by a graphics adapter at the end of a frame. To prevent eye fatigue, the frame rate of an active stereo display should be at least 100 Hz (i.e. 50 Hz per eye). Active stereo systems either use shutter glasses or active polarization filters to direct the left and right eye images into the correct eye (as will be illustrated using Figures 3.2 and 3.3).

Active stereo using shutter glasses
Shutter glasses (see Figure 3.2) use a liquid crystal material that can be turned opaque or transparent under hardware or software control. The glasses are controlled by the graphics system, either through a wire connection or via wireless infra-red, in sync with the rendered left/right images. Although this method of generating stereo images is the most commonly used, the shutter glasses can be quite expensive and
3.3 Design options for a VR architecture

3.3.3 Design options for a VR architecture

Figure 3.2: Active stereo using liquid crystal shutter glasses.

often get uncomfortable over long periods of use.

Active stereo using a “Z” screen

A “Z” screen (see Figure 3.3) is a dynamic optical filter that alternately changes the polarization direction of light that travels through under hardware or software control. The low cost, lightweight glasses worn by the users also contain polarized material that only passes light with a specific polarization direction. Linear polarization is used (“left-right” and “up-down”), which implies that the users should keep their head aligned with the Z screen in order to avoid that images for one eye “bleed through” to the other. Circular polarization filters do not have this problem but these only exist for passive setups.

Figure 3.3: Active stereo using an active polarization screen (“Z” screen) and polarized glasses.

Passive stereo

In passive stereo systems, two display devices are used; one for each eye (see Figure 3.4). Static polarization filters are used to polarize the left and right eye images from the projectors while the users wear low cost, lightweight polarized glasses to direct the left/right images into the correct eye. Again, linear polarization is used in most
cases so that the users must keep their head aligned with the polarization filters to avoid image bleed. Circular polarization filters exist that do not exhibit this problem but these are very expensive and adversely influence image brightness.

![Diagram of passive stereo using two projectors, polarization filters and polarized glasses.](image)

**Figure 3.4: Passive stereo using two projectors, polarization filters and polarized glasses.**

This solution requires "dual-headed" support by both the software and the hardware as two graphics adapters are used. Another disadvantage with using two projectors is that the projectors need to be accurately aligned. With a single projector, calibration is considerably easier.

In any polarization system, special care has to be taken with the choice of screen material since most projection screens adversely influence the polarization direction of incident light. This is especially the case with the use of back-projection systems and even more so with circular polarization.

### 3.3.2 High performance 3D graphics adapters

Graphics hardware performance has increased dramatically over the last years. Although this hardware was mainly intended for use in games, the capabilities of these adapters make them very well suitable for scientific visualization and in some cases rival the capabilities of commercial solutions. With the introduction of 3D accelerated chipsets that include hardware support for 3D operations such as linear transformations, lighting and depth buffering, powerful graphics hardware is now within reach of everyday consumers.

**OpenGL**

OpenGL is the premier environment for developing portable, interactive 2D and 3D graphics applications [261]. With its low-level software interface, OpenGL has often been called the "assembler language" of computer graphics. Applications in many domains, including entertainment, manufacturing, medical imaging and more, have

---

†Note that most graphics adapters for PCs use the AGP interface while no PC motherboards existed at the time of writing which have more than one AGP slot.
benefited from OpenGL's multi-platform accessibility and depth of functionality. Since SGI introduced OpenGL, it has grown into the leading cross-platform graphics Application Programmer's Interface (API).

**Hardware acceleration**

OpenGL's design lends itself very well for hardware acceleration. Hardware accelerated graphics adapters have been available for SGI systems since the early days of OpenGL. But mostly due to the advances in the gaming industry, semi-conductor manufacturers are now rapidly closing this historical gap and in some cases surpass the performance of SGI's hardware. For most high-end graphics adapters, hardware accelerated OpenGL support for various operating systems is available either directly from the vendor or through third-parties. However, this support is in some cases experimental and lacks features that are desired for VR purposes (such as stereo support).

**Stereoscopic 3D**

The essential part of stereoscopic rendering is the generation of two video streams; one for the left eye and one for the right eye. The task of generating and managing stereo pair video streams should not be the responsibility of the application developer. Instead, this should be fully transparent to the developer. The common approach taken by most graphics adapter manufacturers is to use four frame buffers; images for the left and right eye are drawn into a “back” buffer while the user looks at the left and right images in the “front” buffer (commonly referred to as “quad buffered stereo”). While most 3D graphics adapters support quad buffered stereo in hardware, this support is not always reflected in the driver software for some operating systems. In cases where no quad buffered stereo support is available, another method must be used to generate stereo pairs. For the display options described in section 3.3.1, there are a number of alternatives.

**Stereoscopic 3D using X Windows**

The *de facto* standard graphics system for Unix environments is the X Windows system. X Windows uses the concepts of hosts, displays, screens and windows (arranged from broad to narrow scope, see Figure 3.5). X Windows is able to access any graphical display on a connected system using an identifier of the form host: name:displaynumber.screennumber.

Using X Windows, we can opt for one of the following configurations to generate stereo image pairs:
Two hosts, two graphics adapters:

Here two hosts are connected via a network, each equipped with a graphics adapter and display device (a monitor or projector). Using a display identifier as described earlier, each host can access the other host’s display. This configuration is suitable for use in a passive projection system (see Figure 3.4). The greatest advantage of this configuration is that left and right eye images can be rendered by each graphics adapter in parallel, which can improve performance. However, in practice, one of the two hosts will run the VE application which computes the view for both the left and right eye. Then, it renders its own view on its local adapter and the other view is sent over the network. The other host then only has to render the graphics that it receives over the network. This implies a load imbalance which may result in synchronization problems if no countermeasures are taken.

A possible solution to these synchronization problems would be to run the application on a third system, but in that case network traffic doubles since now rendering information for both eyes must be communicated.

\[\text{The X Window implementation should support the GLX protocol in order to run OpenGL applications over a network.}\]
One host, two graphics adapters:

In principle, the synchronization problems described above would be solved by using two graphics adapters in one host (parallelism is then sacrificed, of course). In that case the application renders the left/right views to two displays that reside on the same host. However, this configuration can not be built from commodity hardware since (as noted earlier) most graphics adapters use the AGP interface while PC motherboards only have one AGP slot. Some graphics adapters exist for a PC's PCI interface, allowing an additional card to be added, but it should be noted that the performance of these cards is often substantially less than that of AGP based cards (mainly because of the lower bandwidth of the PCI bus), again resulting in synchronization problems.

One host, multi-headed graphics adapter:

Multiple screens on the same host can be achieved by using a graphics adapter with multi-head output. Synchronization can be controlled since the two views are computed on one host and rendered to two outputs on the same graphics adapter. Care should be taken that the operating and graphics system include multi-headed support (for example; XFree86, an X window implementation available for various operating systems, supports multi-head since version 4.0.1, released late 1999, but it does not support accelerated OpenGL to both heads).
One host, one graphics adapter and sync-doubler:

Specialized hardware in the form of a “sync-doubler” can be incorporated in cases where none of the alternatives described above can be applied. A sync-doubler is a device that accepts as input a video signal of X Hz, and outputs a signal of 2X Hz, whereby the frames are constructed by using the top and bottom half of the input image in an alternating fashion. The top and bottom halves often need to be separated by a number of blank lines.

This simplifies the task of generating a stream of mixed video frames to generating a video stream composed of top/bottom halves. A drawback of the sync-doubler method is that vertical resolution is halved.

3.3.3 Tracking and input devices

Position and orientation are typically determined through six degrees-of-freedom (DOF) tracking sensors (or “trackers”). A tracker mounted on the user’s head allows a VR system to provide user centred projections that correspond to a user’s displacements (“motion parallax”). A hand tracker allows the user to interact with virtual objects. VR input devices often combine trackers with a number of buttons that allow the user to convey intention to the VE, similar to the buttons on a mouse on personal computers.

There are different types of tracking systems, each based on different acquisition techniques such as magnetic, acoustic, optical and inertial trackers. Each of these have their strong and weak points but to describe these would go far beyond the scope of this chapter. By far the most popular in VR systems are the magnetic tracking systems built by Ascension and Polhemus. Although these tracking systems have been in existence for many years, they are still very expensive due to the unavailability of acceptable alternatives. A positive side effect of this is that they are supported by nearly all VR application development environments.

3.3.4 Software availability

Since most of our previously developed software runs on Unix operating systems and makes use of OpenGL, SGI’s OpenGL|Performer, VRCO’s CAVE library, Kitware’s
3.3 Design options for a VR architecture

Visualization Toolkit and DMSO's High Level Architecture, a prerequisite is that this software is available for the new architecture. OpenGL has already been described in section 3.3.2. The following provides a brief overview on the other software packages.

OpenGL Performer

OpenGL Performer is a programming interface for creating real-time visual simulation and other performance-oriented 3D graphics applications [190]. It simplifies the development of VR applications through a low level library providing rendering functions, a scene graph and rendering system, functions for defining both geometric and appearance attributes of three dimensional objects, user-interface components and support for many popular industry standard database formats. OpenGL Performer is currently available for IRIX and Linux systems. Microsoft Windows support will be added in version 3.0 which is scheduled for release in late 2002.

The CAVE library (CAVELib)

CAVELib is an Application Programmers Interface (API) that provides general support for building virtual environments [249]. CAVELib configures display devices, synchronizes processes, draws stereoscopic views, creates a viewer-centred perspective and provides basic networking between remote Virtual Environments. A flexible configuration method makes programs written with CAVELib portable to a wide-variety of display and input devices without rewriting or recompiling. CAVELib currently supports most common Unix systems. Microsoft Windows support was added in 2002.

The Visualization Toolkit (VTK)

VTK is an open source, freely available software system for 3D computer graphics, image processing, and visualization [209]. The design and implementation of this library has been strongly influenced by object-oriented principles. VTK includes a C++ class library containing over 500 visualization objects, and several interpreted interface layers including Tcl, Java, and Python. VTK runs on nearly every Unix based platform and Microsoft Windows.

The High Level Architecture (HLA)

The High Level Architecture (HLA) aims to establish a common architecture for simulation to facilitate interoperability among simulations and promote the reuse of simulations and their components [165]. As a successor to the DIS (Distributed Interactive Simulation) protocol, HLA provides a robust architecture with which distributed discrete event and other types of simulations can be designed. One particular implementation of HLA, the Run Time Infrastructure (RTI) developed by the Defense
The UvA-DRIVE virtual reality system

Modeling and Simulation Office (DMSO, United States Department of Defense), is supported on all major computing platforms.

3.3.5 Computing hardware

The computing hardware performs the main processing in the system, delegating hardware specific tasks to dedicated subprocessors. For a high performance VR system, the components used in the computing hardware should be selected with some care. For example; as most VR applications benefit from a multi processing design, the performance of the system as a whole can be increased by applying a multi-processor design. The easiest multi-processor solution would be a Symmetric MultiProcessing (SMP) architecture in which all processors share the same resources in the system. Although the maximum number of processors in an SMP architecture is limited, it makes porting of existing software far easier when compared to distributed multi-processor systems where some form of synchronization will need to take place. Because graphical performance is important in VR applications, the communication bus between the main CPU and the graphics adapter should be of sufficient bandwidth to handle all data communication within the time constraints imposed by the application. One particular example of a bus architecture in which the communication bandwidth between the main processor and the graphics adapter is optimized can be found in most personal computers (PCs), called the Accelerated Graphics Port (AGP). The initial implementation of AGP allowed for a peak bandwidth of 264 MB/s which was almost double the 133 MB/s bandwidth provided by the main PCI bus found in PCs at that time. In later versions of AGP, bandwidth was increased further by allowing multiple data transfers per clock cycle which resulted in bandwidths of 528 MB/s (for AGP 2×), 1056 MB/s (for AGP 4×) and 2112 MB/s (for AGP 8×).

Most manufacturers of computing hardware have taken some, or all of these issues into account in their designs. Mass acceptance has caused PC technology to be able to rival, both in terms of performance and quality, that of large commercial computer hardware manufacturers but at a significantly lower price. In addition, PC based hardware allows high flexibility and scalability in the sense that hardware can be replaced by newer (in general; faster, better) hardware easily and at low cost as soon as it is available.

Operating system

Several operating systems are available for Intel processor based PC platforms, including IRIX, Solaris, various flavours of BSD and Linux. However, not all of these provide both the hardware and software support required for compatibility with the VR applications as described in section 3.3.4. The Linux operating system in this respect provides the most complete support.
3.4 The design of UvA-DRIVE

After careful consideration of the requirements posed in section 3.2 and the design options outlined in section 3.3, we have opted for an Intel PC based computing system in conjunction with an active shutter glasses system as shown in Figure 3.2. The operating system running on this system is Linux. A first prototype of our design was built by SARA in 2001, called the Linux Immersive Environment (SARA LIIE). Multiple systems have been derived from this prototype, including UvA-DRIVE. Unfortunately, at the time of construction, none of the drivers for OpenGL accelerated adapters that were supported under Linux contained quad buffered stereo support. As an intermediate solution for the prototype, we use the “sync-doubler” technique described in section 3.3.2 to generate stereo pair video streams. The sync-doubler also drives an infrared transmitter that switches the shutter glasses. As we have opted to use an active stereo approach, the system requires only one projector. A schematic representation of the architecture is shown in Figure 3.6. Table 3.1 provides a list of components and specifications of both SARA LIIE and UvA-DRIVE compared to the SARA CAVE.

3.4.1 Top-bottom stereo with CAVELib

Since all previously developed VR software is based on VRCo's CAVELib, we have chosen to use CAVELib to minimize porting effort. Using CAVELib, the application programmer can ignore technicalities like the number of projection screens, the setup of the projection screens, the mono/stereo properties of projection screens. Instead, the application developer can focus on that what matters: the 3D scene. How this 3D scene is conveyed to the user is irrelevant from the developer's point of view. CAVELib has a feature that allows the rendering for left/right eye to be performed at a specified sub window of the physical screen. Also, CAVELib supports applications

\[nVidia\] (a manufacturer of high performance graphics chips used in many PCs today) released Linux drivers with quad buffered stereo support on September 10, 2002 (version 1.0-3123) [173].
The UvA-DRIVE virtual reality system

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU</strong></td>
<td>8 × MIPS R10000</td>
<td>2 × Intel Pentium-III</td>
<td>2 × Intel Pentium-III</td>
</tr>
<tr>
<td><strong>CPU clock</strong></td>
<td>195 MHz</td>
<td>700 MHz</td>
<td>1.0 GHz</td>
</tr>
<tr>
<td><strong>architecture</strong></td>
<td>Onyx2 Reality Monster</td>
<td>ASUS P2B-D, SMP Intel 440BX chipset</td>
<td>ASUS CU4VX-D, SMP VIA Apollo Pro chipset</td>
</tr>
<tr>
<td><strong>memory</strong></td>
<td>1 GB</td>
<td>256 MB, 100 MHz FSB</td>
<td>1 GB, 133 MHz FSB</td>
</tr>
<tr>
<td><strong>graphics</strong></td>
<td>4 × InfiniteReality2</td>
<td>Creative Labs nVidia GeForce2 GTS</td>
<td>ASUS V7700 (modified) nVidia Quadro2 Pro</td>
</tr>
<tr>
<td><strong>gfx memory</strong></td>
<td>4 × 16 MB</td>
<td>32 MB video memory</td>
<td>64 MB video memory</td>
</tr>
<tr>
<td><strong>CPU→gfx bus</strong></td>
<td>800 MB/s</td>
<td>528 MB/s (AGP 2×)</td>
<td>528 MB/s (AGP 2×)</td>
</tr>
<tr>
<td><strong>projector</strong></td>
<td>Electrohome Marquee 8500</td>
<td>Electrohome Marquee 8500</td>
<td>Electrohome Marquee 8110+</td>
</tr>
<tr>
<td><strong>screens</strong></td>
<td>4 (CAVE)</td>
<td>1 (IDesk)</td>
<td>1 (InfinityBox)</td>
</tr>
<tr>
<td><strong>tracking</strong></td>
<td>Ascension FoB four sensors</td>
<td>Ascension FoB two sensors</td>
<td>Polhemus Fastrak two sensors</td>
</tr>
<tr>
<td><strong>OS</strong></td>
<td>IRIX 6.5</td>
<td>Linux Debian (&quot;sid&quot;) kernel 2.2.18</td>
<td>Linux RedHat 7.3 kernel 2.4.19</td>
</tr>
<tr>
<td><strong>software</strong></td>
<td>X Windows OpenGL 1.1 Performer 2.3 CAVELib 2.7</td>
<td>XFree86 4.0.2 nVidia GLX 0.9-6 Performer 2.3 CAVELib 2.7</td>
<td>XFree86 4.0.1a nVidia GLX 1.0-2960 Performer 2.3 CAVELib 3.0.1</td>
</tr>
</tbody>
</table>

Table 3.1: Comparison of components and specifications of the CAVE, SARA LIE and UvA-DRIVE.

that use OpenGL/Performer. This creates two additional advantages: minimal porting effort for the migration from SGI (CAVE) to Linux, save a recompilation of the sources, and also good possibilities for performance measurements comparing SGI and PC based VR.

XFree86 was configured to run at 1024x1576 resolution with a colour depth of 24 bit per pixel. This means, that after sync-doubling, \((1576-40)/2 = 768\) pixels remain in the vertical direction\(^*\); the same as on the SGI Onyx platform used in the SARA CAVE. We have found that a vertical sync of 85 Hz (170 Hz when doubled) is possible, provided that a high quality projector is used.

### 3.4.2 Performance measurements

We have run benchmarks with full tracking support and stereo vision support. The program to test the performance is an application developed by SARA, on top of VRCO’s CAVELib.

In benchmark A, the dataset consists of 16 animated objects. Each object consists of 9120 triangles (a total of 145920 triangles), organized in 405 triangle strips. In benchmark B, the dataset consists of a surface model extracted from an Computed

\(^*\) The sync-doubler requires 40 blank lines between windows.
3.5 Conclusions

Tomography (CT) dataset using a surface extraction algorithm. The dataset contains 204480 triangles, organized in 39134 triangle strips.

Two platforms were used for benchmarking. The first is SARA's CAVE facility; an SGI Onyx2 Reality Monster with 8 R10000 CPUs at 195 MHz, running IRIX 6.5. This system is equipped with 4 InfiniteReality2 graphics pipes. For benchmarking, we disabled all but one screen. The CAVE ran the benchmarks at a stereo 1024x768 resolution, 60 Hz refresh rate (120 Hz stereo), and full screen anti-aliasing.

The second system is the SARA LIE system described in section 3.4, and ran the benchmarks at a stereo 1024x768 resolution, 58.4 Hz refresh rate (116.8 Hz stereo), and without full screen anti-aliasing. Both tests did not include texture mapping.

<table>
<thead>
<tr>
<th></th>
<th>test A</th>
<th>test B</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAVE</td>
<td>13.3</td>
<td>5.5</td>
</tr>
<tr>
<td>SARA LIE</td>
<td>11.7</td>
<td>6.3</td>
</tr>
</tbody>
</table>

Table 3.2: Performance in frames per second of the SARA CAVE and the SARA LIE prototype in two tests.

The results (see Table 3.2) show that there is no significant performance difference between the two platforms. The PC based system performs better on test B. This is mainly due to the fact that a single Intel CPU has a better floating point performance than a single MIPS CPU [181].

Figure 3.7: The UvA-DRIVE system.

3.5 Conclusions

Immersive VR on commodity hardware shows great promise. Although not all the required hard- and software components for a single screen immersive VR system based

---

Although the nVidia GLX implementation supports anti-aliasing, we could not get CAVELib for Linux to enable this mode.
on commodity-of-the-shelf are available yet, progress is made or satisfactory alternatives exist. Our first experiences show that a PC based system is a viable alternative to established VR systems. We have shown that graphics performance for PC based solutions is in the same order as that of high-end VR systems and is therefore no longer a criterion for choosing one solution over the other. Moreover, compatibility with existing VR applications is provided with currently existing solutions. Because PC based VR systems are affordable, offer adequate performance and are compatible with high-end VR systems, a host of new application domains now comes in reach.

Currently, work is in progress at the Section Computational Science for the construction of a PC based near-field VR system called the “Personal Space Station” (PSS). This design by Mulder et al. uses a conventional computer display in combination with a mirror and a camera based tracking system [168] (see also [213] and [186]). As this design does not use a projector or a large projection screen, it can be built at significantly lower cost and is suitable for personal use in a normal office environment. A PSS is well suited for applications in which direct, hand-eye coordinated interaction with virtual objects is important.