# Virtual Reality Interaction Techniques

*Michael Rorke*
Computer Science Department
Rhodes University
Grahamstown, 6139
South Africa
**mrorke@cs.ru.ac.za**
http://cs.ru.ac.za/homes/g97rc001/

*Prof. Shaun Bangay*
Computer Science Department
Rhodes University
Grahamstown, 6139
South Africa
**cssb@cs.ru.ac.za**

*Prof. Peter Wentworth (HOD)*
Computer Science Department
Rhodes University
Grahamstown, 6139
South Africa
**cspw@cs.ru.ac.za**

## Abstract

This paper addresses the problems associated with interaction in immersive virtual reality and makes recommendations as to how best to deal with these problems, thereby producing a usable virtual reality interactive environment. Immersive virtual reality means that the users are *immersed* or contained inside the environment in which they are working. For example, they are able to turn their heads and look around, as well as use their bodies to control the system.

The work in progress involves a study of various virtual reality input devices, some designed and implemented as part of the project. Additionally, the paper describes a simple framework for separation of the interaction and application parts of a virtual reality system in order to facilitate an object oriented approach to the implementation of the recommendations, and to the building of future virtual reality applications which incorporate these ideas.

## Keywords

Virtual Reality; interfaces; 3D widgets; interaction.

## Introduction

With the current growth in the power of the desktop computer and the growing availability of dedicated graphics rendering hardware, virtual reality is becoming more and more mainstream in its application. PC Magazine [Ozer, 1998] predicts that **standard** desktop machines will soon be equipped with 3D accelerator cards, with performance levels on these machines reaching four times the level of performance on current, high-end workstations. The next generation Intel$^{tm}$ chipset (nicknamed *Katmai*) will support an extended set of native instructions, specifically supporting 3D graphics rendering.

While the hardware now exists to support virtual reality applications, the software tools available in this field are still lacking in usability.

## Interaction Problems in Virtual Reality

Virtual reality aims to allow the users to interact with the system they are using on a more intuitive level e.g. via gestures and movement as opposed to typing commands on a keyboard. These added interaction possibilities are what makes virtual reality so exciting in its application. Unfortunately, they also make it difficult to learn and use efficiently. The most notably successful applications of virtual reality all fall into the realm of spatial visualisation [Mine, 1997 (2)], with little or no attempt made to allow direct object manipulation. The main reasons for this lack of interactive usability can be summarised as follows:

1) The precise manipulation of objects in the virtual environment is difficult. While one is able to accurately track the positions of objects and represent this visually, the lack of haptic feedback makes it difficult for

the user to precisely manipulate objects. Haptic feedback is the term given to 'feel' of an object. This 'feel' is a result of the weight of the object pulling the hand downwards, and the pressure exerted by the fingers on the object, all of which the brain registers and uses to help accurately position or manipulate the object. At present, virtual reality is able to reproduce only the visual information about the object. Thus, the users may see their 'hands' holding an object, but, at no point do they actually believe that they are holding something real.

2) The equipment used to capture data about the user for the virtual environment (e.g. magnetic trackers, which provide a 3D value for their position and orientation) is often prohibitively expensive. Thus one is forced to use a limited number (usually 2 or 3) of them. This, in turn, restricts the amount of input one is able to receive from the user, and thus, restricts the usability of the system.

3) Another problem associated with many, if not all, virtual reality input devices is that of limited precision. There is a limit to the accuracy of the readings that these devices produce, and they are also prone to an effect known as ‘drifting’ whereby the value from the device changes when it should remain steady (e.g. if the tracker has not moved).

4) The lack of physical work surfaces in the virtual environment is also a major interaction problem. People depend on naturally occurring physical constraints to give them some idea of the behaviour of objects (e.g. a book pushed over the edge of a desk will fall to the ground [Mine, 1997 (1), (2)]). These physical workspaces also often provide some form of support for the user, alleviating fatigue and allowing more precise manipulation of limbs.

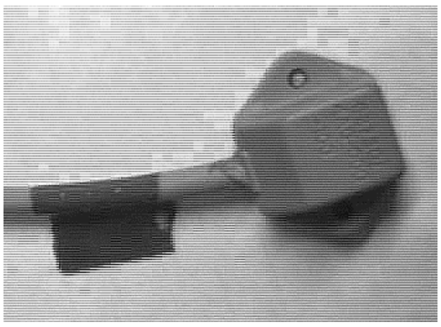
Figure 1 : Polhemus$^{tm}$ Magnetic Tracker

5) Virtual environments also lack a common interface. The standard Windows, Icons, Mouse and Pointer (WIMP) interface is now common on all computer platforms, and the user is easily able to identify common interface elements and begin productive work with little or no learning curve. Unfortunately, no such common interface exists in virtual reality. There is no set of standard input devices for example. So the user is forced to start from a very basic level whenever a new piece of software is encountered.

These problems with virtual reality interaction were all identified at an early stage in its development, and many different solutions have been proposed for specific problems.

## Investigating the Problems

In order to better understand these problems, a simple interaction system has been implemented. The system has been implemented using the Rhodes University, CoRgi virtual reality programming libraries, and initially ran on a Silicon Graphics$^{tm}$ Octane. The system has since been ported to

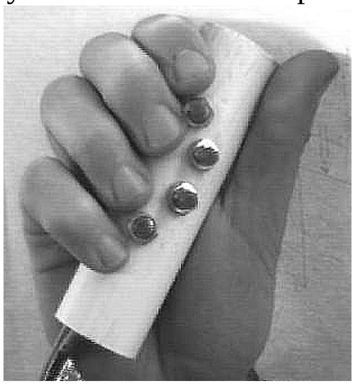
Figure 2 : Virtual 'Stick' Input Device

an Intel$^{tm}$ P2 system running Linux. The input device currently consists of a Polhemus$^{tm}$ magnetic tracker (Figure 1) and an simple device comprised of a small handle with four switches built into it (Figure 2), called a *Virtual Stick.*

The stick is held in the user' s hand with a button assigned to each finger. The tracker is placed inside the stick, allowing the system to accurately trace the position of the users hand. The four buttons on the stick are used to represent the state of the user' s fingers to the system. Using this device one is able to get a simple, yet usable representation of the user' s hand, within the system. The virtual stick is similar in design to one-handed keyboards, known as *Chord Keyboards*, but the addition of the magnetic tracker makes it sufficiently different. Additionally, unlike the Chord keyboard, the intention of the stick is to input specific human gesture information into the system, not arbitrary combinations of on or off for a set of switches. The stick is designed to cheaply replace more complicated input devices, like *Datagloves*, while still providing the same usability.

Different combinations of on/off states of the four buttons, representing different combinations of open and closed fingers on the user, are used to implement simple gesture recognition in the system. Currently the system understands two gestures. The first gesture (Figure 3) consists of all the buttons being closed i.e. all the fingers of the user' s hand closed around the stick. Using this

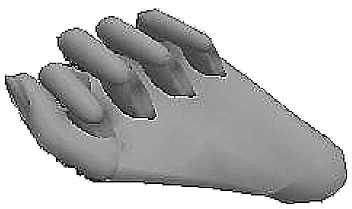
Figure 3 - The ' hand' in grabbing mode

gesture, the user is able to pick up and manipulate objects in the world. The current manipulation technique consists of simply attaching the object being manipulated to the object representing the user' s hand, thus mimicking the movements of the user' s hand. The second gesture (Figure 4) is a pointing gesture. This is represented by closing all the fingers, except the index finger. This gesture is used to operate the menu objects that are present in the system.

## Initial Usability Study

This system was used as the basis for a set of qualitative usability tests, carried out on a group of people from the Rhodes University campus. The subjects were presented with a virtual world, containing three chess pieces, and asked to ' pick-up' and rotate the chess pieces using the stick input device. All of the participants were taken from outside of the Computer Science department, and had limited knowledge of computers, and no specific knowledge of virtual reality. All were able immediately to identify the hand and once the participants had become accustomed to the system, a process taking about 3-5 minutes, they were able to complete this task fairly easily. This simple example demonstrates the power of a virtual reality system. In order to have accomplished the same task using conventional Windows$^{tm}$ based 3D software, like 3D Studio, the participants would have had to familiarise themselves with the interface, choose the correct view, choose the correct widget and then use it in the correct manner - tasks that would have certainly taken in excess of 30 minutes. All these tasks were accomplished intuitively using the virtual reality system.

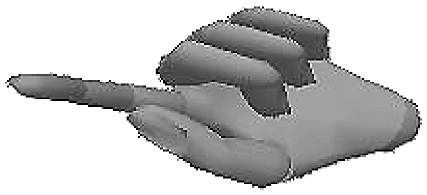
Figure 4 - The ' hand' in pointing mo

The second task that the participants were asked to complete was to place one of the chess pieces exactly on top of the other. None of the participants was able to complete this task successfully. All were able to move the first chess piece to a position almost on top of the second, but none was able to position it exactly. Many of the participants also became fatigued after about 15 minutes. This was consistent with the various interaction problems discussed in the previous section.

## Solving Interaction Problems

The first and most difficult problem to solve, is the lack of haptic feedback. There is no easy way to simulate the weight of an object to the user. It is possible to simulate, to a certain degree, the ' pressure' felt by the hand as a result of holding the object. The methods for simulating the feel of an object range from electrical stimulation of the nerves of the fingers, to the usage of air sacks to put pressure on the fingers. None of these methods satisfactorily reproduces the sense of touch that a user has when holding a real object. An alternative solution to this problem is to provide real world equivalents of the objects in the virtual world, which the user is able to physically pick up, thus utilising the full range of haptic feedback. The *Virtual Tricorder* [Wloka, 1995] is an example of this idea, whereby a 3D mouse is used as the principal input device, and given a representation in the virtual world, corresponding to its physical size, shape, etc. Allowing the constrained movement of objects [Bowman, 1995] is another method that helps solve the problem of accurately placing objects inside a virtual environment. Constrained movement means allowing the object only to move in a certain direction at any given time. For example, the user may choose to constrain the object to move only along the x-axis, in which case, the y- and z-axis changes that come from the input device are simply disregarded. This idea can be further extended to the case where numerical input of data (from the keyboard or some virtual representation thereof [Mine, 1997 (2)] is allowed for the precise placing of objects.

The next problem mentioned is the prohibitive cost of the input devices used, especially the cost of the magnetic trackers on which most immersive virtual reality systems rely so heavily. The problems associated with having only a limited number of trackers present can often be alleviated using mathematical methods like inverse kinematics whereby the positions of limbs, joints, etc. of the user that are not directly tracked can be estimated, based on the known positions of their other limbs, joints, etc. Another possible solution to this problem comes from the fact that, as the technology behind these devices becomes more established, their price will drop. The problems associated with the limited precision of the input devices is often a matter of the technology behind the device. As with the problem of cost, as the devices are used, more and more research goes into their manufacture, so they will become more accurate. Other problems, like drifting, can also only be solved with better hardware.

The lack of physical workspaces in the virtual world is currently a topic of much research. The addition of ' workbenches' and touch sensitive tablets [Mine, 1997 (2)], in both the virtual and real worlds of the user, look to go a long way towards alleviating this interaction problem. But such devices/objects often restrict the movements of the user in one way or another, and thus their introduction into a general interaction system may have detrimental effects. They have proved very useful in solving specific interaction problems, but, as yet, no single device exists which solves general interaction problems.

Probably the biggest drawback to the field of virtual reality at the moment is the lack of any unifying framework for interaction [Mine, 1997 (1), (2)]. The reason for this lack of unity in the field of virtual reality interaction stems from the fact that there is no standard set of input devices, and, even with assumptions made as to what input devices are to be used, the range of interaction possibilities makes it difficult to settle on a ' common group' of actions which will be able to service the whole of the virtual reality field. A further problem arises from the fact that the desktop interaction metaphor (Windows, Icons, Mouse & Pointer) is no longer sufficient, yet the ' real-world' metaphor where an object' s ' use' may be determined by looking at its physical constraints (for example), is also not applicable, due to the lack of information one is able to convey to the user about the objects in the virtual environment i.e. no haptic feedback, for example. There are also other very fundamental differences between the desktop and virtual reality interaction metaphors. For example, in virtual reality, the user can be considered to be *inside* the interface [Mine, 1997 (2)]. As users move around the world, the interface elements that they use to interact with it must move around as well, in order to be easy to locate and reach. These elements also take up valuable space on the display, so they must also be kept out of the ' field of vision' of the user when not needed. Proprioception [Mine, 1997 (1), (2)] is the term used to describe a person' s sense of the position and orientation of their body. This idea has been used, with some success, to solve the problem of where to place interface elements so that they are always at hand when the user needs them, yet not constantly obscuring the display. The interface elements are attached to different parts of the user' s bodies e.g. behind their heads, out of the field of view, yet they can always be easily reached when needed.

## 3D Widgets

The success of the WIMP interface can be attributed mainly to the fact that it provides users with a familiar set of tools, no matter what particular application they may be using. Virtual reality can offer this same familiarity, but not just widgets that look and act like widgets that the user is familiar with from other computer programs. Rather, virtual reality can take the idea a step further, presenting the user with widgets which look and act like objects or tools with which the user is familiar from the real world.

The interface elements in the virtual world should all the modelled, as closely as possible, after real world equivalents. Thus, the user should be able to immediately identify the operation of the various elements, from their knowledge of the real world. For instance, the system described in the previous section uses a model of a human hand as the basic interaction element, with gestures being the basic operations. This method has proved very intuitive, and even novice users are immediately able to identify the interface element, and use it to manipulate objects in the world, based on their intuitive knowledge of the working of the human hand.

In order to produce an intuitive interface, the interface elements should adhere to the following guidelines [Norman, 1990]. Interface elements should have *affordances*. These affordances are elements of the objects that explain its operation to the user. For example, the virtual hand has fingers, implying that the user is able to grasp or point at objects. *Mappings* must exist between user actions and their effects on the system i.e. any input action by the user should produce a proportional output action in the system. *Feedback* is also a major factor contributing to a usable interface. The user should never be in doubt as to whether or not some action has been accomplished or not. Good feedback should naturally follow from good mappings [Bowman, 1995]. For example, in the virtual hand system, when an object has been successfully ' grabbed' , its representation is changed by adding a semi-transparent bounding sphere around the object being manipulated, and placing a set of x-, y- and z-axes onto the object. Feedback in the virtual hand system also follows from the fact that when a button on the virtual stick is closed, the finger on the virtual hand, corresponding to that button, closes, giving the user immediate information as to the purpose of the buttons and the working of the virtual stick. The fourth and final guideline is that of *constraints*, which has been covered in the previous section.

## Interaction Abstraction Framework

In order to best judge the relative merits of all the numerous solutions that have been proposed to solve these interaction problems, these solutions need to be implemented and their effectiveness tested. The system described earlier currently implements only the simplest interaction system, with none of the proposed solutions to the major interaction problems yet implemented. The system  has been designed to make it easy to implement different types of interaction system quickly, thus enable the user to pick and choose the one that best suited their current application.

Figure 5 gives a simple overview of the data that needs to pass between the interaction component, the system that it is controlling and the objects in that system. The interaction component needs to tell the system what object has been selected by the user, or what action is chosen (e.g. via a menu system), to be performed. The system, in turn, needs to return feedback information to the interaction component (e.g. changing the colour of active interface elements) and also needs to communicate control information. The interaction component also communicates directly with the objects in the system, performing *Actions* on them and receiving *Replies* back in response to these actions. Object dependencies are often best handled by the objects themselves, as opposed to trying to build them into the interaction component or system. Thus, if the interaction component wants to move an object, for example, it sends a request to the object to move. The object in turn decides whether or not to allow this movement. The issue of where to handle dependencies is not a simple problem, and often depends on the given system. In the generic case it is best to split this between the system, the interaction component and the objects themselves, as the particular problem dictates. As a simple example, consider the way in which the system handles the selection  of an item from a menu. The menu object in the  world contains  a group of  button objects, each of which can be activated by 'p ressing'  the button, with the hand in pointing mode. When such an event occurs, the interface component sends a *press* action to the object being acted on (a button in this case). The object itself then checks that the requested operation is allowed (dependency). As this is a valid operation, the interface component receives a number from the object (identifying the action to be performed) which it passes back to the system. This number uniquely identifies the button that has been pressed and

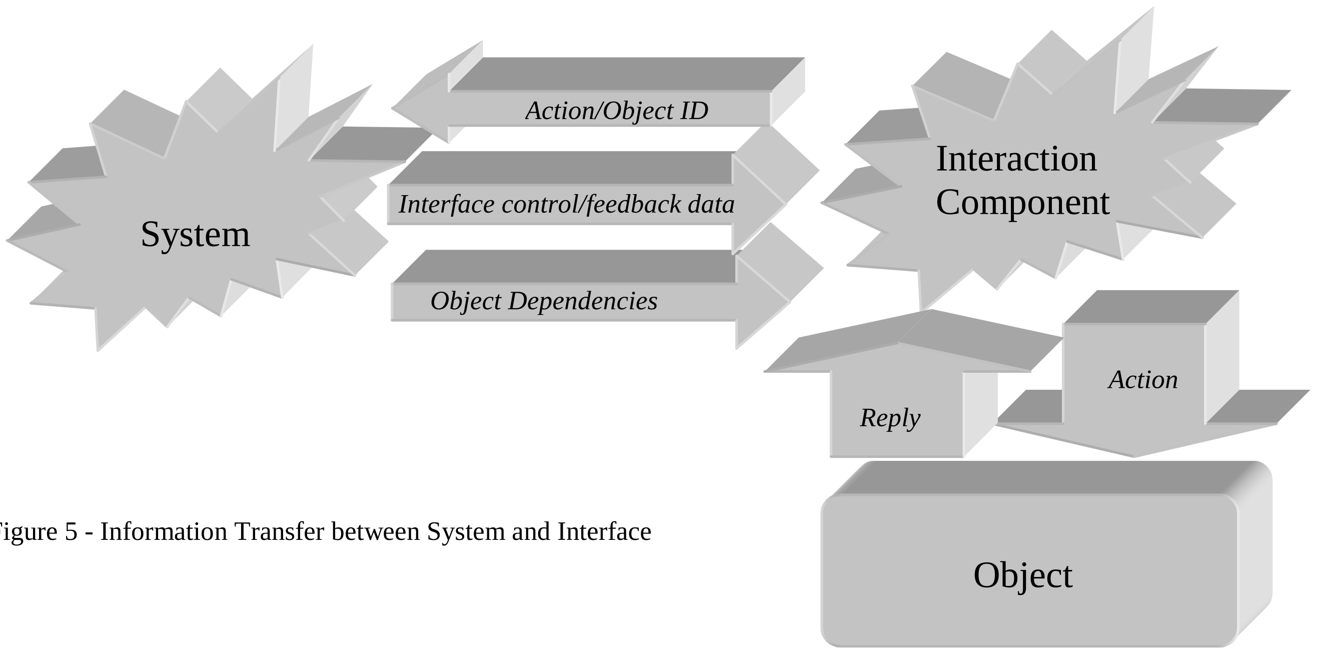

Figure 5 - Information Transfer between System and Interface

allows the system to take the appropriate action. On the other hand, if the user attempts to move the button with the hand in grabbing mode, the interface component sends a *grab* action to the button. In this case, the button has a dependency that doesn' t allow it to be moved independently of the menu it is on. Thus, the operation of grabbing a button will not succeed.

This abstraction of interface and system allows the developer to develop virtual reality applications independent of the input devices available on the user machine. As long as both the system and the interaction component share the same common interface, they are able to work with each other. The developer is also able to provide multiple interaction components to the user, each with its own merits, and allow the user to choose the one best suited to the operation required.

## Conclusion

Virtual reality is no longer hampered by the absence of appropriate hardware, but rather the absence of understanding about the medium and how to deal with its shortcomings. The sensory input missing from current virtual reality systems (e.g. the lack of haptic feedback) must be compensated for, in order to make these systems more accessible and usable to the general public. The methods presented in this paper provide the tools for overcoming these problems, providing a framework for creating better and more usable applications of the technology.

## Future Work

The next step will be to implement some of the proposed solutions to the major interaction problems, and to investigate the differences each makes to the overall usability of the system. Once this has been done, the system will be expanded to include those techniques that are deemed most useful. As an extension to the interaction framework idea, a GUI builder could be developed, similar in operation to Visual Basic, whereby the developer could rapidly generate virtual reality applications, using a standard set of interface elements and interaction techniques.

### References

[Bowman, 1995]

Doug A. Bowman and Larry F. Hodges***, User Interface Constraints for Immersive Virtual Environment Applications***, Graphics, Visualisation and Usability Centre, Georgia Institute of Technology, Technical Report TR95-26

[Mine, 1997 (1)]
Mine, Mark, Frederick P. Brooks Jr., and Carlo Sequin (1997). ***Moving objects in Space: Exploiting Proprioception in Virtual-Environment Interaction***. Proceedings of SIGGRAPH 97, Los Angeles, CA

[Mine, 1997 (2)]

Mine, Mark. ***Exploiting Proprioception in Virtual-Environment Interaction***. Doctoral dissertation, Department of Computer Science, University of North Carolina, Chapel Hill, 1997.

[Norman, 1990]

D. Norman, ***The Design of Everyday Things***, Doubleday, New York, 1990

[Ozer, 1998]

Jan Ozer, ***3D Computing***, PC Magazine, Vol. 17, No. 11 (June 1998), pp 118

[Wloka, 1995]

Wloka, M. and Greenfield, E., ***The Virtual Tricorder: A Uniform Interface for Virtual Reality***. In Proceedings of UIST' 95, ACM Press, November, 1995, pp. 39-40.