Design Issues for Virtual Reality Systems*

Roger Hubbold, Alan Murta, Adrian West, Toby Howard

Advanced Interfaces Group
Department of Computer Science
University of Manchester
Oxford Road
Manchester M13 9PL
England

email: rhubbold@cs.man.ac.uk

August 1993

Abstract

In this paper we describe a number of issues which are central to the design of a software architecture for a distributed, generic, virtual reality system. These include support for diverse and demanding applications, the management of time to provide high-quality interaction with tightly controlled closed-loop feedback, and the need for continuity of the experience presented to the user. These issues are being addressed in the design of a generic VR system called AVIARY.

1 Introduction

An interesting feature of VR is that it is not based on any fundamentally new technology. Input devices based on direct interaction, such as the light pen, joystick and mouse have been around for a long time. Similarly, displays capable of rendering 3D scenes have also existed for many years, and indeed these things have been routinely coupled in CAD packages, flight and military simulators, and computer graphics research laboratories.

What makes VR a radical approach to HCI is not the technology (although evolutionary progress has eased the task) but its application. It is a subtly new way of using established technology and techniques, but with dramatic implications for HCI.

*Presented at the *First Eurographics Workshop on Virtual Environments*, Barcelona, 7th September 1993

The new feature that VR brings to the HCI scene is the removal of the computer as an object of perception, allowing the user to *interact directly* with the generated environment. Whilst technically similar to flight simulation, this is psychologically radically different as we are no longer looking into a window, but are actually in the environment. Although in its infancy, the potential for solving some of the thornier HCI problems, such as natural interaction with 3D objects, is clear.

Exciting though this all is, we are not there yet. Although the potential has been demonstrated, we are far from having a mature technology capable of dealing with the large scale use of VR for significant applications, with any degree of richness. Some issues are implementation-specific, others – to a far greater degree than in previous interfaces – belong to the realm of psychology and perception.

If we stand back and consider what is necessary to bridge the gap between the existing state of the art and a future mature technology, we begin to see the range of issues that is raised, and the nature of some of the interesting problems that must be addressed.

In the Advanced Interfaces Group at the University of Manchester we have been attempting to clarify and address this area. We are currently implementing a software kernel in a newly established laboratory to take our ideas further, as documented elsewhere [1, 2].

In this paper we wish to raise some problems that we consider important, and which will need to be addressed if VR is to progress. Two particular threads can be distinguished: the importance of addressing the human perceptual mechanisms, and the new perspective that VR brings to existing implementation issues.

1.1 Engaging perceptual mechanisms

In our everyday existence we cope with, and filter out, tremendous amounts of information almost effortlessly and with very little conscious thought. Indeed, if the same information, in all its detail, were to be presented in a form that we had to think about consciously, then we would be overwhelmed quite easily. Spatial awareness, pattern recognition, information filtering, coordination of multiple information streams, are things we take for granted. Rather than look for a solution in AI, part of the VR thesis is that information presented in a suitable way can be processed far more effectively and directly by people.

Traditional interfaces are suited to, and have largely relied on, human cognitive abilities with relatively low bandwidths between operator and machine. Interfaces that engage perceptual skills have potentially far higher bandwidth. Used in this way the machine marshalls and presents information, a task to which it is generally well-suited, and the operator's perceptual mechanisms filter and abstract behaviour of interest – a task to which they are well-suited. This information is then processed cognitively and acted upon. VR offers a means of engaging these perceptual skills far more directly and effectively than is generally offered by traditional screen based interfaces.

One of the primary goals of VR research, then, must be to discover how to engage human perception most effectively. The technological priorities may not be the obvious ones. For example, players soon adapt to quite unsophisticated graphics in some video games, and become deeply engaged in the scenario,

provided certain levels of cues and interactions exist. Conversely, even good photo-realistic images fail to retain attention if there is little that can be done with them, or if the interaction lags are substantial.

The quality of an interface in these terms is notoriously difficult to quantify, or even to describe to others. Research can therefore all too easily become focussed on measurable issues such as frames per second, or number of primitives in the scene, which may ultimately be of only secondary importance.

For example, one issue that we would regard as central to the success of large scale VR systems is that of coherency. We believe it is of great importance that some degree of consistency between different virtual worlds and applications can be maintained if users are to develop skills for inhabiting environments and navigating between them. Total consistency is neither practical or necessary, but what degree of coherency is required for users to perceptually orient themselves in the complex environment?

Engaging human perceptual skills and perceptual coherency are, we believe, keys to the success of VR.

1.2 A new solution space for old problems

A second feature that seems generally characteristic of VR issues is presented here in the form of an observation. Many of the problems involved in the implementation of a VR system are those of well-studied areas of computer science. This is most clearly true with computer graphics, parallel processing, discrete event simulation, multimedia and HCI. Whilst problems in these areas have been extensively studied, and certain tasks are known to be difficult, VR brings a sufficient perspective shift to allow us to seek new solutions to these problems. In this way the subtly different requirements of VR may help us to see beyond some of our current computational preoccupations. One example would be the emphasis on photo-realism in many areas of computer graphics.

VR also provides an opportunity to bring together techniques which have been developed for specialised applications, but have not as yet been integrated in a more general environment. Examples include different strategies for model culling and rendering, and synchronisation of simulation and interaction. These examples are discussed later in this paper.

With this background, we will now present some of the central issues with which we are concerned. These arise from our desire to construct a VR ‘operating system’ as a base for real applications. They range over the abstract and the concrete but are all real issues that we must address.

2 Generic or specific support – can they be reconciled for VR?

The goal of our research is the design of a generic VR system capable of supporting a diverse range of large-scale applications within a single, coherent environment.

The need for a general approach is highlighted by considering the wide range of disparate applications

which may benefit from the integration of VR-style interfaces. Three-dimensional CAD packages, for example, may exploit novel VR input techniques for model building. Scientific visualization systems may be greatly enhanced by allowing the user to explore and manipulate complex data representations 'from within'. Other applications, such as air traffic control systems, or surgical training simulators, require a mix of 3D viewing capabilities coupled with unobtrusive means of interacting with real-time environments.

A particularly interesting task, which has always faced the vendors of support environments, is that of providing a system flexible enough to cope with a range of diverse applications, and yet allowing some regularity and efficiency. Unix could be said to have been quite successful in this respect, and there is much of relevance to VR in operating systems concepts. This is clearly an active and pertinent issue in the future of VR.

For a future, mature, general purpose VR system it will not be sufficient merely to provide a single world and an object-oriented environment, requiring users to tailor the class hierarchies for their own applications. Whilst this approach is useful in smaller systems, for more ambitious environments it is rather limited and difficult to manage. The diversity of applications for which VR seems appropriate suggests that a variety of world models are required [1]. It is not possible to provide all features within a single world, not only due to the contradictions that would ensue, but also because of the performance implications of such an approach. This is an interesting challenge to reconcile with the need for coherency of experience in the system as a whole.

2.1 The distinction between application and system

In considering a generic large scale environment of the future, one detailed issue that is raised is the distinction between the system and its applications. Exactly where does the boundary between application and system lie? This is much more clearly an issue if we consider a system with multiple active applications within a single virtual world. There is a direct analogy with operating system/application interfaces which have evolved over the last few decades.

Separating what is application, from what is virtual world is not simply an academic exercise, as clear distinctions will be needed if the users' conceptual model of a complex system is also to be clear. In practice elegant distinctions of this kind tend to be compromised for all sorts of reasons, but there is no reason why a clean conceptual model should not be devised to which the system as a whole should aspire.

3 Perceptual consistency and display performance

Maintaining the illusion of a perceptually consistent world places quite stringent demands upon its implementation. Unless a system can respond rapidly and without distracting artifacts, the advantages of immersive environments may be unattainable. This area is complex, because consistency does not

necessarily imply, for example, real-time behaviour for all tasks. More important is the creation of an environment in which the user remains comfortable and well oriented. In some cases this may demand real-time response, but in others it may be more important to maintain a sense of continuity [3]. We are used to dealing with a real world in which some actions may take a finite time to complete. Thus, when calling a lift, we do not necessarily expect that the lift will arrive instantaneously. On the other hand, the movement of objects attached directly to our hands, or the performance of head tracking, must occur in real-time if we are to maintain the illusion of reality.

Thus, a key issue for any immersive VR system is that of the 'closed loop' feedback time for tracking the user's movements – for both head and limbs [4]. Most low-cost VR systems are woefully inadequate in this respect, which is the primary reason that many VR demonstrations portray very simple models. Rapid feedback during navigation is central to the tight coupling of users' actions to the system's response. There are actually two separate factors which affect the closed loop response time: the rate at which pictures can be animated, and the rate at which inputs from the human participants can be decoded and acted upon.

Fast animation. The simplest way to characterise the speed of animation is to count the number of frames per second required to create the illusion of continuous smooth motion, as users move through a world, or objects in the world move and change. This varies between individuals, but is generally at least 20. In practice, frame rates of less than 60 Hz may produce ghosting effects, whereby an observer sees multiple images of objects due to discrete changes in their positions. A classic example of this is to watch the telegraph poles in some of Silicon Graphics' simulations of driving a vehicle around a landscape. At less than 20 Hz the scenes appear as a series of separate frames, and the illusion of smooth motion is lost. At faster rates, such as 30 Hz, the frames merge to provide a smooth sequence, but the telegraph poles appear to be duplicated! Only at rates approaching 60 Hz does the motion appear completely smooth.

Sensor tracking. It is essential that inputs can be processed sufficiently rapidly to avoid delays between actions by users and the system responding accordingly. In slow systems it is possible to get completely out of synchronisation, so that the displayed image of a user's waving hand, for example, is completely out of phase with its true position. We will return to this point again in section 4.

Some researchers take the view that solving these problems is only a question of building faster hardware. However, if we are to build portable VR systems which work well with a range of applications, and on equipment which is cost effective – which means cheap, if VR is to be widely used – then we must examine the problem from a more fundamental viewpoint. We also need to take account of the characteristics of 'real' applications, which are often quite complex – support of these must be a central goal of our research.

3.1 Animation rates for high-performance workstations

To understand better the issue of scene complexity and its effect on animation, we have considered the rendering speeds of current high-performance 3D workstations, such as the Silicon Graphics VGXT and

Reality Engine [5], and the Evans and Sutherland Freedom Graphics Series for Sun workstations [6]. The usual way manufacturers quote rendering performance is in terms of some number of notional triangles, quadrilaterals (or triangle or quadrilateral strips), with various shading options (flat, Gouraud, Phong etc.). Impressive rates, in the order of 1 million or more triangles per second can be achieved in theory. But for real applications, with all of the constraints which apply when designing and writing code – and here we exempt very special cases such as military simulators – rates in the order of 10% to 20% of the peak performance are more realistic. If we assume a speed of 200,000 triangles per second (i.e. 20%), and a minimum required frame rate of 20 Hz, then we must limit our model to at most 10,000 triangles.

Generous though this may sound, in many applications it is inadequate. In one application we are studying, 3D CAD models contain as many as 50,000 primitives [7]. These primitives include shapes such as planes, boxes, cylinders, spheres, and cones. Some of these generate many polygons, taking us well beyond the 10,000 triangle limit for a 20 Hz display rate. Unfortunately, although alternative algorithms exist for direct rendering of primitives such as quadrics (e.g. ray-casting), these are not supported in current display hardware, so that everything must be reduced to polygons. Curved surfaces, and even mathematically simple shapes such as spheres, eat up polygons at an alarming rate, dramatically reducing the complexity of models which can be displayed at adequate frame rates.

3.2 Adaptive rendering

One approach to this problem is to use adaptive rendering techniques, so that only those parts of the model within the field of view are rendered, and only those close to the observer are shown in detail. However, solutions to this are frequently specific to particular applications and to particular hardware platforms [8]. This issue is closely related to the problem of hidden surface removal, and experience with this suggests that the use of application knowledge is a major factor in solving it effectively. For example:

- In an architectural walk-through, we can use knowledge of a building's layout to cull parts of the model which are known not to be visible from the current viewing position. Thus, if we know that the walls are solid, we need only display objects visible within the room, and perhaps a further subset visible through open doors or windows [9].
- In flight simulators it is common to employ a pre-processing step, based on binary space partition (BSP) trees and backface culling, in order to determine in a view-independent way those objects which are potentially visible from different viewpoints. Then, at runtime, only the subset of potentially visible objects requires processing, selected according to the real viewpoint [10, 11]. Additionally, levels of detail – usually applied as texture maps – can be made dependent on the distance of the observer from each surface. Mipmaps offer one way to introduce greater detail as the observer moves closer to a surface [12, 8].
- With finite element models, we can use connectivity information in the element data structures to quickly determine a subset of potentially visible element faces.

In each of these cases, we can use application-specific knowledge to provide an efficient solution, but these solutions cannot necessarily be applied in other situations. For example, in the case of an airport or terrain model, only a small part of the model will change between frames. This may not be the case for other applications, and the BSP pre-processing step is very time-consuming for models of any complexity. A general VR system must attempt to provide a framework for solving this problem in a flexible manner.

3.3 The influence of perceptual issues on rendering

Any solution which involves the selective display of information must strive to maintain a perceptually smooth transition between levels of detail. Objects which pop on and off a display can be highly distracting. Picture complexity is often viewpoint-dependent, so that turning one's head can produce very sudden changes in the quantity of data to be processed graphically. The characteristics of different hardware platforms may vary widely in this respect. If displays cannot deliver the required performance then methods are required which degrade gracefully. Even if display performance continues to improve dramatically, a solution for this problem will still be needed for the low-cost end of the market.

Factors other than mere speed are also important here. Many displays can draw 3D vectors faster than filled polygons, suggesting that during manipulation, or navigation, a wireframe mode may be used. Unfortunately, switching modes in this way can be unsettling for the user, and a wireframe view often results in considerable clutter for models of any complexity. Extensive testing will be required to establish the most effective ways of culling a model to achieve acceptable frame rates allied to consistent data presentation. Research into such techniques has been pursued for many years in the flight simulator market, but its application to a much wider range of tasks is still needed.

It is worth observing that it is not obvious what cues require optimising in order to create a convincing world. Anyone who has played a video game in an arcade will know that fast system response is a key element in engaging the player, and this can overcome limitations in the way information is presented. However, in other areas more faithful rendering of the scene may be required, an example being a surgical training system. It is clear that the *quantity* of information which can be displayed depends on the speed of the display system, but the choice of *what* to display will be application-dependent, and not necessarily based solely on distance from the observer.

4 Temporal consistency

The issues of frame rates and device input lag are examples of a more fundamental problem which a general-purpose VR system must solve, namely, the provision of a consistent framework for managing time in an environment where the goal is, at least, to effectively simulate real-time processing.

The determination of input sensor locations, the simulation of virtual world events and the generation of visual and other sensory representations all require finite amounts of time. As a result, input handling,

simulation and rendering all occur within effectively different time-frames. This creates two problems: the skew in action and observation can be perceptually disturbing to the user [13], and the lack of a global frame of reference may impair the integrity of the world simulation.

One method which is currently used to tackle the cognitive skew problem is to use predictive algorithms to calculate the location and motion of input sensors at some small temporal offset in the future in order to bring sensor time in-step with simulation or rendering completion time [14]. This has been shown to be more acceptable to the human perceptual system, although the shortfalls of this method can become apparent when the user is drawn to the temporal discrepancies (by hand-clapping in the real and virtual world, for example). The only completely effective solution to this problem is to keep accumulated latencies below the minimum which can be perceived (around 25 msec for interactive tasks [15]), and this should become less of an issue as VR technologies improve.

The rational handling of time within a simulated environment is a more difficult issue. It is a problem which has been addressed many times in the computing field, in the context of electronic circuit simulations, event profiling in distributed systems, physical modelling, and so on. In many respects, the problem of temporal management in a VR system is more demanding, as solutions must be found in real-time. As a result, many of the computationally expensive solutions which are applied in other areas are inappropriate. However, some aspects of VR modelling may allow certain degrees of freedom with respect to event synchronisation, unlike related disciplines such as electronic simulation, which must adhere to strict rules of causality. Identifying where in the system such tolerances are allowed is an interesting research area.

A closely related issue is the maintenance of deterministic behaviour. Virtual interactions should behave consistently, whether they are carried out in real time, in slow motion, or on a platform incapable of the fine grained simulation of a more powerful system. The problems of synchronism may also have a fundamental effect on the meaning of user interactions. An example of this is a command such as ‘delete this’, where the command is uttered into a speech recognition system and the ‘this’ is indicated by pointing or touching an object in the virtual world.

The principles of synchronisation, latencies, and local time frames within a VR system must therefore be understood in order to devise coherent methodologies for time management which can deliver a conceptually consistent behaviour to the system users. Some groundwork in this area has already been undertaken in the multimedia field, in which time-augmented Petri nets are used to assess synchronisation problems in the delivery of multimedia presentations [16].

Many existing VR systems are somewhat restrictive in their support for temporal management, and simply coordinate events with respect to some specific aspect of the VR platform, such as frame delivery rates. We believe a better approach is to consider time as a completely independent resource, effectively using a time server to manage coordination, synchronisation and the rate of flow of time itself within the virtual world. This is a more general solution, since it allows greater flexibility in the handling of virtual time (allowing multiple rates of flow within the system, for example), and separates world behaviour from specific performance-related metrics imposed by the underlying hardware.

5 The AVIARY system

AVIARY is the generic VR support environment being developed by our group in Manchester [1, 2]. This system will allow us to explore many of the design issues addressed within this paper. The AVIARY system supports:

A hierarchical world model constructed using object-oriented methods. This permits the tailoring of new worlds by software re-use, and also provides a framework for the design of consistent techniques for interaction and presentation across a range of applications.

Multiple, concurrently active applications with well-defined interfaces to the virtual world manager, and between applications.

Platform independence. A prototype version of the system has been developed and executes on a 64-processor, transputer-based parallel machine, or alternatively on one or more networked Sun workstations. Ports to other platforms are currently in progress (see below).

Distributed processing. AVIARY was designed from the outset to operate in parallel and distributed environments. We believe the exploitation of concurrency to be an important research issue for a number of reasons. Firstly, virtual environments are inherently concurrent in nature, requiring the simultaneous processing of many different input, output, simulation and sensory rendering tasks. Secondly, such an approach allows us to tackle large-scale applications which would be considered unmanageable on uniprocessor systems. Another important factor is that the implementation issues of distributed synchronisation are closely tied to the problems of temporal management within virtual worlds; a research area which we are already addressing.

Distributed graphics processing. The need for high-performance means that dedicated graphics hardware is required. In order to minimise communication in a distributed environment we use retained mode graphics, with picture information held in a local display list (or structure) store. For example, the prototype AVIARY system uses PHIGS [17] for the graphics. This allows small changes, such as altering a viewpoint, to be communicated very efficiently, but runs counter to the direct mode graphics used by many workstations. To maintain platform independence, an internal graphics protocol is employed so that the system is not dependent on a specific graphics system.

We are currently establishing a new VR laboratory, equipped with high-performance workstations and a complement of VR input and output devices including head-mounted display, 3D input sensors, audio I/O, and speech recognition hardware. We are fortunate in having access to a 64-processor Kendall Square Research KSR1 – a parallel, virtual shared-memory computer [18]. This machine has 2 gigabytes of memory, an inter-processor communication speed of 1 gigabyte per second, and a theoretical peak computation rate of 2.4 gigaflops. This raw power will allow us to explore computationally demanding applications such as scientific simulation and visualization, and medical imaging. Work in these areas is already progressing on our KSR1. The AVIARY environment will enable us to use the KSR1 for simulation and virtual world modelling, with input processing and rendering tasks being off-loaded to

specialised equipment in the VR laboratory. The system as a whole will permit us to further develop the AVIARY prototype in its support for a range of 'real world' applications.

A low-cost, high-performance graphics subsystem is being designed and built by some of our engineering colleagues [1]. This is intended to provide a cost-effective method for driving several headsets, permitting experiments with multiple users. It will also provide an opportunity to test the behaviour of AVIARY with with lower performance displays than the dedicated graphics workstations.

6 Conclusions

In this paper we have tried to show that building a generic VR system, which can support a broad range of applications, requires much more than software and hardware technology. Even today's most expensive displays are quite limited in the amount of information they can portray, at frame rates fast enough to create the illusion of a 'realistic', continuous world.

One of the major strengths of immersive environments is their capacity to engage their users' perceptual skills in solving problems. This involves a complex set of issues: it is the way that users perceive their environment which should determine the crucial performance measures for an implementation, and we should not blindly assume that just building faster hardware will provide the solutions to the problems. In any case, for many users ultra-fast hardware is simply too expensive.

Human beings are remarkably adaptable. An attraction of virtual environments is that we can suspend the 'real' world. Not only can we change our size – from the galactic to the microscopic – we can alter time, so that events move faster or slower. In effect, we can create slow-motion worlds, and replay events, so that we can study them in detail, and we can speed up time, as when carrying out simulations of phenomena such as the weather. The concept of time within a VR system is central to managing these problems, providing a framework for resolving the different time-bases involved in tracking user inputs, synchronising displayed images, and interfacing to simulations of various phenomena.

From a computer science perspective, the inherently concurrent nature of virtual worlds, and the issues of synchronisation and management of time, make parallel and distributed solutions to these problems an interesting topic for research.

References

[1] A.J. West, T.L.J. Howard, R.J. Hubbold, A.D. Murta, D.N. Snowdon, and D.A. Butler. AVIARY – A Generic Virtual Reality Interface for Real Applications. In R.A. Earnshaw, M.A. Gigante, and H. Jones, editors, *Virtual Reality Systems*, chapter 15, pages 213–236. Academic Press, March 1993.

[2] David N. Snowdon, Adrian J. West, and Toby L. J. Howard. Towards the Next Generation of Human-Computer Interface. In *Informatique'93: Interface to Real & Virtual Worlds*, pages 399–408, March 1993.

[3] George G. Robertson, Jock D. Mackinlay, and Stuart K. Card. Cone trees: Animated 3D visualizations of hierarchical information. *Communications of the ACM*, 34(2):189–194, February 1991.

[4] Roy S. Kalawsky. *The Science of Virtual Reality and Virtual Environments*. Addison Wesley, 1993.

[5] Silicon Graphics Inc., 2011 N. Shoreline Boulevard, Mountain View, CA 94043, U.S.A. *Workstation Product Documentation*.

[6] Evans and Sutherland Design Systems Division, 580 Arapeen Drive, Salt Lake City, Utah 84108, U.S.A. *Freedom Graphics System*.

[7] CADCentre Ltd, High Cross, Madingley Road, Cambridge CB3 0HB, U.K. *Plant Design Management System*.

[8] Silicon Graphics Inc., 2011 N. Shoreline Boulevard, Mountain View, CA 94043, U.S.A. *Iris Performer reference manual*.

[9] John M. Airey, John H. Rohlf, and Frederick P. Brooks Jr. Towards image realism with interactive update rates in complex virtual building environments. *ACM Computer Graphics*, 24(2):41–50, March 1990.

[10] I.E. Sutherland, R.F. Sproull, and R.A. Schumacker. A characterisation of ten hidden-surface algorithms. *ACM Computing Surveys*, 6(1):1–55, March 1974.

[11] H. Fuchs, Z.M. Kedem, and B.F. Naylor. On visible surface generation by a priori tree structures. *ACM Computer Graphics*, 14(3):124–133, July 1980.

[12] Lance Williams. Pyramidal parametrics. *ACM Computer Graphics*, 17(3):1–11, July 1983.

[13] R. Held. Correlation and decorrelation between visual displays and motor output. In *Motion sickness, visual displays, and armoured vehicle design*. Aberdeen Proving Ground, Maryland, Ballistic Research Laboratory, 1990.

[14] Martin Friedmann, Thad Starner, and Alex Pentland. Device synchronization using an optimal linear filter. In R.A. Earnshaw, M.A. Gigante, and H. Jones, editors, *Virtual Reality Systems*, chapter 9, pages 119–132. Academic Press, March 1993.

[15] S.K. Card, T.P. Moran, and A. Newell. *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates, Publishers, Hillsdale, New Jersey, 1983.

[16] T. Little. Managing time in multimedia. In J. Rosenberg, editor, *SIGGRAPH '91 Panel Proceedings*. ACM, 1991.

[17] T.L.J. Howard, W.T. Hewitt, R.J. Hubbold, and K.M. Wyrwas. *A Practical Introduction to PHIGS and PHIGS PLUS*. Addison-Wesley, Wokingham, England, 1991.

[18] Steven Frank, Henry Burkhardt, and James Rothnie. The KSR1: bridging the gap between shared memory and MPPs. In *Proceedings of Compcon93*, pages 285–294, February 1993.