

Techniques for Reducing Virtual Reality Latency with Architectural Support and Consideration of Human Factors

Ronald Pose and Matthew Regan

Department of Computer Science,
Monash University,
Clayton, Victoria 3168,
AUSTRALIA
E-mail: {rdp,regan}@cs.monash.edu.au

Abstract. Virtual Reality is an exciting new area of human-computer interaction. Its use depends critically on having extremely low response times of the virtual environment to user interaction. The problem of reducing the lag between the user's head position or orientation changing and the updating of the displayed view has been described as one of the grand challenges of computing. The techniques described here form the basis of a solution for this problem. A radical new approach to the problem is presented and the influence of human factors is also considered.

1 Introduction

Researchers and practitioners have long been grappling with the serious problem of latency in responding to movement of a user in a virtual environment. This problem is most difficult to handle in the visual domain since the sheer quantity of data and processing required to update the view delivered to the user is beyond the capabilities of even high-end computer graphics systems. The problem, while simply described as the lag between a user's head position or orientation changing and the updating of the displayed virtual view to reflect that change, has rather severe consequences, and has eluded solution by many practitioners. A particularly disturbing effect is that of motion sickness which occurs when the brain's expectation of orientation and position is not matched by appropriate sensory input, in this case visual input. To put it simply the view displayed is not what the brain expects so disorientation and perhaps motion sickness can occur.

2 Conventional Approaches to the Problems

One approach to building Virtual Reality systems has been to slow the update rate of the display so as to enable the computer system to cope with the processing load.

This leads to a rather jerky displayed world with latency being at least the time between updates of the display. Experience with motion pictures and with television has shown that the higher the frame rate the better.

An alternative approach is to reduce the processing load by reducing the quality and resolution of the images displayed. While the display update rate is increased and latency somewhat reduced with this approach, the realism of the virtual world is sacrificed leaving one with cartoon-like imagery rather than anything approaching photo realism.

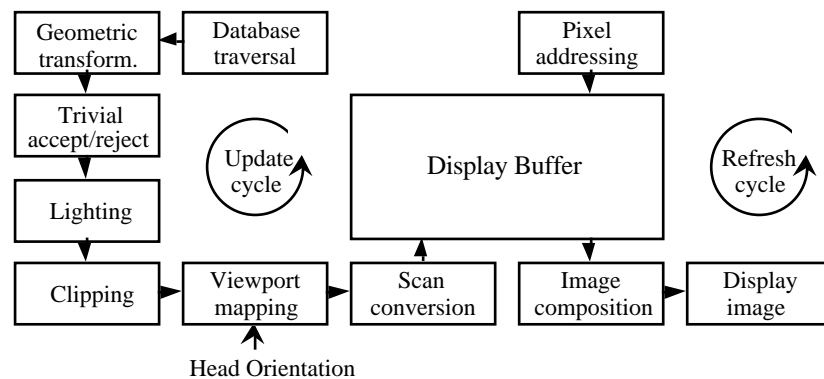


Fig. 1. Conventional Virtual Reality Display Architecture.

The state-of-the-art in conventional Virtual Reality systems rely on using the fastest available graphics processing and rendering equipment as depicted in Figure 1. This tends to be very expensive with the very best systems costing over a million dollars. In effect the problem is tackled indirectly through the use of massive amounts of processing power. Even so it can be observed that the trade-off between latency and image quality still has to be made, and that with reasonably realistic looking scenes the latency is still a significant problem. An extremely successful Virtual Reality computing system is the PixelPlanes system developed at the University of North Carolina [Fuchs et al., 1989]. This uses thousands of processors to give extremely impressive rendering performance and very good latency characteristics, however PixelPlanes is rather expensive technology which most developers could not afford.

3 Re-Examining the Problem

In order to deal with this problem of latency it is helpful to forget for a moment the conventional ways of computer graphics and instead look at the overall environment in which we are working.

The problem as stated earlier is one of mismatch between what is displayed to the user and what should be displayed according to the virtual world model in which

the user is situated. This mismatch usually is due to the latency involved in updating the display to reflect the virtual reality. The easy solution, involving reducing image quality so one can keep up with the virtual world, creates a mismatch of a different kind, but still spoils the illusion of Virtual Reality.

Let us examine what is happening in the virtual world itself. It could be that some part of the virtual environment is changing or moving in some way. While indeed this does occur, it is also true that in general only a small part of the virtual world is changing at any time, and typically the virtual world contains many static objects or scenes. Thus we can deal with such activity without recomputing everything, and since the objects involved are generally independent of the user, a slight delay in reflecting their changes does not cause the serious problems mentioned above.

The other event, which is much more significant, is the movement of the user within the virtual environment. Any slight movement of the user's head leads to a change in everything the user sees, hence it looks to the user as though the observed environment has moved completely. Thus one can see that it is the relative movement between the user and the objects in the virtual world which is observable to the user, and it is the user's own movement which has the most dramatic effect, since it affects the display of even static parts of the virtual world.

Now we should look more closely at the various types of user movement, to see what the consequences are. Consider the obvious movement, translation, for example walking forward. When translating, objects in the world appear to move relative to you. Close objects appear to move more than distant ones. Very distant objects may not appear to move much at all. Thus for translations we have to concentrate especially on close objects, and can largely ignore the distant background. In typical scenes the majority of the complexity of the scene is in the background, hence the problem appears tractable. We are also fortunate in that people do not tend to translate so very quickly so changes tend to happen at a manageable rate.

Another user movement is that of rotation of the head. Here we have a rather different effect. Even a small rotation immediately causes everything in the observed view to appear to move, even the background and static objects. There is also the added problem that rotations are much faster and more frequent than translations; one tends to look around and observe one's environment by rotating one's head or eyes. It looks as though everything has to be recomputed and re-rendered.

However, let us not forget the idea of stepping back and looking at the overall environment. It seems that the environment is staying relatively static but the user's view of the environment is changing quite rapidly and somewhat unpredictably, mainly due to rotation. The key point here is that the environment itself is not changing dramatically, so one needs to find a way in which the representation of the displayed environment also stays relatively static, and hence can be computed without the serious problems with which we are concerned.

4 A New Model

The ancient Greeks found a similar problem in trying to describe the motions of the planets and stars. They had a concept of planets moving relative to stars and to the Earth, and stars moving relative to the Earth. A model involving *crystal spheres* was developed. In this model the heavenly bodies were *painted* onto various layers of crystal spheres which were centred on the Earth and could move relative to one another. Objects on close spheres move more compared with objects on distant spheres. Various complex models of the movements of heavenly bodies were formulated in terms of these spherical shells surrounding the Earth. We now know that much of the observed motion of the stars is due to the rotation of the Earth, but since the observed motion is relative, a successful model centred on the Earth is possible.

We can use a similar model for a virtual world. Concentric spheres centred on the user would have the objects of the virtual world painted on them. Close objects on inner spheres and distant objects on outer spheres. In essence all possible views from the user's position are already rendered onto the spheres, and it only remains to display appropriate views as seen by the user. A rotation merely involves displaying a different portion of the sphere. A translation will require some updating of the spheres, however outer spheres will change much less than inner spheres, and hence may require little or no updating. Of course changes within the virtual environment will have to be reflected in the spheres, but typically only a subset of the spheres will be involved.

5 Implementation of Viewport Independent Display Memory

We have devised a model in which we render the images of the virtual world onto a surface or surfaces surrounding the user. However it may appear that in so doing we have actually created a larger processing bottleneck, both in generating the images and rendering them, and in selecting the appropriate view to display. On first examination it may appear we have significantly greater rendering overheads such as scan conversion, clipping etc. than a conventional system, however this is rarely the case, and is only found if the scene has polygons evenly spread out in all three dimensional directions. With a viewport independent display memory one must scan convert all polygons received. This is the worst case scenario for a conventional system, but for guaranteed interactive response one must allow for the worst case. Many conventional rendering systems are designed to cope with situations approaching the worst case scenario [Molnar & Fuchs, 1990]. The rendering overheads for a conventional system may be reduced if the user is not looking at a complex part of the scene, however as the system has no control over the user's choice of direction for viewing, it is fair to assume the user is looking at the most polygonally dense section of the world. The viewport mapping is indeed

computationally expensive however this task has been offloaded into relatively simple and cheap dedicated hardware, the *Address Recalculation Pipeline* [Regan & Pose, 1993].

The latency is determined by how long it takes to select the required portion of the encapsulating surface and write the image to the display device. This is handled by the Address Recalculation Pipeline which runs at the speed of the display device and introduces negligible latency. The new improved approach is depicted in Figure 2 and a comparison of the latency components with those of conventional systems is shown in Figure 3.

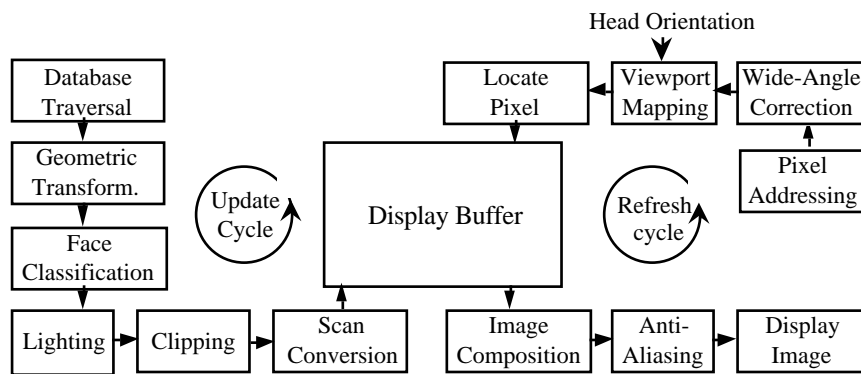


Fig. 2. Delayed Viewport Mapping.

The selection of the appropriate region of the sphere to display can be modelled as projecting rays from the eye onto the surface of the sphere, so defining the area to be seen. Alternatively one can view the process as one of changing coordinate systems from real-world to spherical world. This indeed requires a great deal of computation for each pixel displayed and would be infeasible in software. However the approach is ideal for a pipelined hardware implementation, which while using fairly fast hardware has a fairly simple structure and can be built economically [Regan & Pose, 1993].

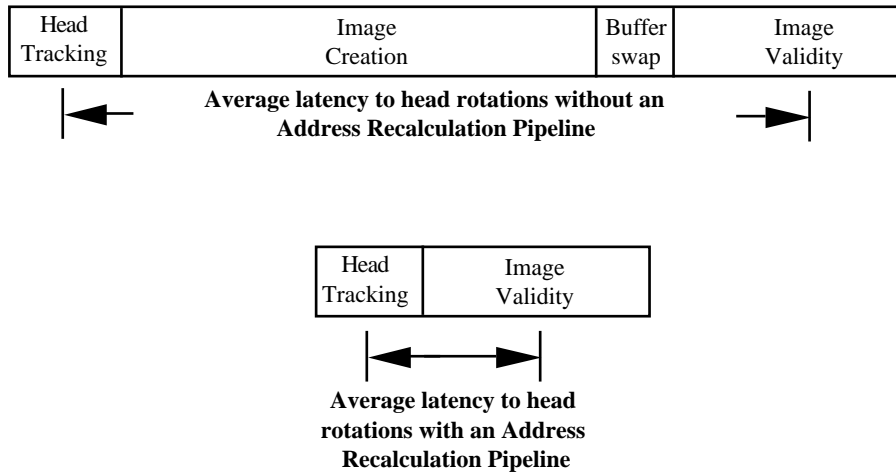


Fig. 3. Comparison of rotational latency components.

We did investigate a hardware design based on a spherical system but found some annoying characteristics. First, in mapping a sphere onto a memory array one tends to waste a lot of memory, or else have a very complex mapping scheme. Second, the apparent sizes of pixels in a sphere vary greatly as one moves from equator to the poles. Third, the coordinate transformations involve trigonometric functions which are moderately expensive to implement. By far the dominant issue is the first, in that we are using much more display memory than a conventional graphics system, and since we are not accessing it in a nice regular pattern, we are forced to use quite fast memory, hence we do not want to waste any of it.

Essentially the basic model will work for any surface surrounding the user. A sphere is intuitively obvious in that everything is equidistant, however one can go to the other extreme and look at a tetrahedron. What we eventually chose was a cube. It has very nice properties in that no memory is wasted since its surfaces are the same shape as memory chips. The coordinate transformations are also much simpler in that they are linear and can be implemented essentially with a matrix multiplication. Pixel sizes do not vary as much as for a sphere. While there may be some concern about strange effects occurring in corners and edges these have been shown not to be significant.

A prototype display memory system based on the cube has been implemented and functions well. We call this method using viewport independent rendering, *Delayed Viewport Mapping*, and the hardware realization, an *Address Recalculation Pipeline* [Regan & Pose, 1993].

6 Image Composition

A technique which has been shown to work well in accelerating computer graphics systems is image composition [Molnar, 1991]. Instead of treating the scene as a single entity, one can break it down into component objects and combine them at the end to produce the scene. One can render these component objects separately, and if one has multiple renderers available one can get parallelism in the rendering process. The big gain is that objects which do not change can be left, and not rendered again if you want to change the scene. Thus one can achieve better than linear speed-up. Unfortunately this does not help much in Virtual Reality applications since, as we have seen, a simple rotation will cause every object in a scene to change.

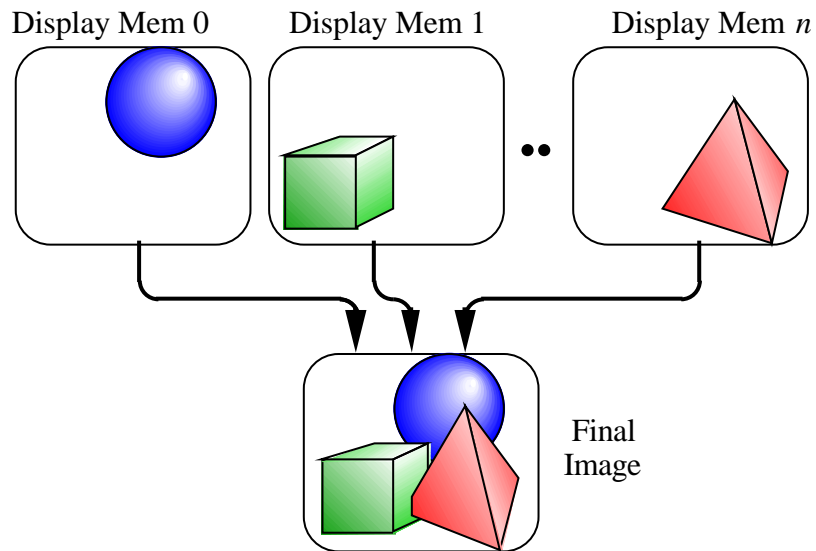


Fig. 4. Image Composition.

We can get the benefits of image composition by employing our viewport independent approach in Virtual Reality applications. Since the surface encapsulating the user does not change when the user rotates, the objects already rendered can remain, and once again we get better than linear speed-up with multiple renderers. Conceptually we have gone back to the *crystal sphere* model, and our image composition involves combining images of objects painted on various concentric cubes. We use the Z-values of pixels to enable us to choose the closest pixel to the user as the one to be displayed. Hence we do not force objects onto particular cubes in relation to their apparent depth, but instead allow the hardware to resolve dynamically which pixels to display. We also provide separate rendering engines for each cube. Thus it makes sense to distribute objects over cubes in such a way that those that need to be updated at similar times are grouped together. In this way the

rendering capacity is not wasted on redundantly re-rendering objects which have not changed. A technique called *Prioritized Rendering* has been developed to handle display updates efficiently [Regan & Pose, 1994].

The technique of image composition is depicted in Figure 4. Image composition occurs as pixels are fetched from the display memories to be displayed on the output device. All of the pixels from the display memories which correspond to the same screen location are compared to find the closest pixel, which is then displayed.

7 Prioritized Rendering

Objects in the virtual world will appear to change at different rates. The apparent movement of the object due to a rotation of the user is handled automatically by the *Address Recalculation Pipeline* which implements *Delayed Viewport Mapping*. Other changes in the way the object is perceived are due to the object's own animation and due to translational movement of the user in the virtual world. It is possible to classify further the kinds of changes in the appearance of the object into those involving its displayed size and those involving its displayed position. Objects will appear larger when one is closer to them and smaller when one is farther away. Relative side-to-side movement will cause an apparent change in the position in which the object is displayed. Of course any animation of the object itself can lead to both these effects. The effects of user translation can be quantified using simple geometric analysis [Regan & Pose, 1994]. Using this approach one can calculate the amount of time for which the current displayed image of the object will remain valid, and hence determine the required rate of update of the object's displayed image.

Prioritized Rendering takes advantage of the fact that not all objects need to be updated at the same rate. By sorting the objects into a priority order based on their required update rates, one can render the most important changes in object appearance first, thus guaranteeing that if perchance there is insufficient rendering capacity, one has achieved the best possible result. Even in such an unfortunate case the latency of response to head orientation changes is not affected, since that is handled by the *Address Recalculation Pipeline*.

This however is not the main advantage of *Prioritized Rendering*. Its most dramatic property is that the overall rendering required has been reduced significantly. We have shown that an order of magnitude reduction in required rendering is not unreasonable due to this approach [Regan & Pose, 1994]. This occurs because the average required update rate for all objects in the virtual world is much less than the display update rate. By balancing the rendering load among multiple renderers updating multiple display memories at various update rates and composing the final image for display, the combination of *Prioritized Rendering*, image composition, and the *Delayed Viewport Mapping* provided by the *Address Recalculation Pipeline* can outperform conventional Virtual Reality implementations.

8 Implications for Building Virtual Reality Applications

Mechanisms have been described which enable the efficient implementation of Virtual Reality applications without the serious tradeoff between latency and image resolution which has plagued most existing systems. However the mechanisms will not necessarily be exploited to their fullest unless one designs the Virtual Reality applications with them in mind.

The conventional approach has been to run a tight loop involving the tracking equipment which monitors the user's position and orientation, the virtual world model, and the rendering of views of that model into the display device. What tended to suffer was frame rate, and hence latency, when the computer system could not keep up with the processing load. In such systems the frame rate would drop as the changes to the scene increased. More sophisticated systems would drop the accuracy of the virtual world model to minimize changes which require re-rendering. Other systems would reduce the accuracy of their rendering to try to maintain a minimum frame rate. All such systems degrade rapidly when stressed by changes in the view of the virtual environment.

In our system, latency to rotation is independent of the virtual world; it is a function of our hardware and is effectively limited by the tracking equipment. Translational latency is reduced indirectly in that distant objects in the background will not be observed to move and need not be re-rendered. Changes in the environment itself have still to be handled, but by grouping them according to their observed speeds one can minimize the number of cubes which need to be updated. In the same way, changes due to translations and due to preserving stereoscopy can be grouped according to the required update rates, and allocated to the display memories appropriately.

We do not claim that our system can display perfect renditions of a fast changing view of a virtual world without latency. Certainly the extremely important and difficult case of rotational latency is effectively solved, however translations and actual virtual environmental changes can conceivably overload the system. The important thing to note is that such problems would result in out of date representations of less important parts of the virtual world as seen by the user. It would not affect the smooth motion by lowering the frame rate and would not let the image quality suffer. Such an overload condition can only occur during very rapid translation or when there are fast moving objects in the virtual environment. In such cases the ability of the human eye to perceive much detail is limited since fast moving objects tend to blur. Hence it is unlikely that such overload would even be noticed by the users unless the system was grossly under powered.

In order to exploit the system one must change the paradigm used in conventional Virtual Reality systems. One must, in effect, ignore the rendering and instead concentrate on a much higher level, that of objects in the virtual environment. Mechanisms are already in place to handle rendering of objects in viewport

independent display memories, but the management of these objects and the allocation of them to particular display memories is very important. One must arrange for objects which are the focus of attention to be rendered on time at all costs. Similarly one must identify background features and fairly static objects and group them together as objects that rarely need to be re-rendered.

Indeed one can consider the required approach to be somewhat object-oriented, and one should try to concentrate on gross object behaviour rather than fine object detail. Given enough processing power the detail will be handled. In those cases where processing power is a limiting factor, it is likely that the view is changing so rapidly that fine details would not be missed anyway.

9 Human Factors in Applying the Techniques

While the techniques we have developed incorporating *Prioritized Rendering*, *Delayed Viewport Mapping* and image composition, have enabled us to produce a Virtual Reality display system with latency characteristics far superior to conventional approaches, and which enable us to cut dramatically the computationally expensive rendering requirements of such systems, a great deal of further improvement in performance can be obtained by considering human factors concerning the relationship between the user and the virtual world.

Prioritized Rendering can use the expected validity time of the object's displayed image to determine an appropriate update rate. If the validity time is very long, that implies that the object's displayed image is changing very slowly. Often that means that the object's change of appearance is not the focus of the user's attention, and hence slight delays in updating that object will not be noticed. Of course in such cases there is not much rendering of the object going on anyway, but it may be useful in the extreme case of an overloaded system to know that very slowly changing things can be deferred in favour of more quickly changing objects, which are more likely to be the focus of the user's attention.

At the other extreme, objects with very short expected validity times indicate very rapid changes in the object itself or its relative position with respect to the user. While such objects may indeed be the focus of attention, if they are changing or apparently moving at great speeds, then we can rely on the human visual system not being able to perceive accurately the details of fast moving objects. Factors such as motion blur come into play. We can thus take advantage of the lack of accurate perception of fast changing images, to allow a reduction of rendering in this case. Certainly in this case the rendering load of the object will be significant since the object has a high update rate. Human factors here should allow us to reduce the update rate if there is a shortage of rendering capacity, without a significant perceived degradation in performance. It is certainly expected that one could reduce the fastest rate of update to about 25-30 Hz., motion picture and television update rates, if necessary, without user perceptible degradation. Of course update rates greater than

the update rate of the display device cannot have any observable effects and should be avoided.

When the system is pushed to its limits in terms of computational capacity available for rendering of objects, there is the possibility of taking advantage of the user's focus of attention to allow selective degradation of image resolution on peripheral or background objects. This is possible due to having multiple display memories with individual renderers which may not only operate at different update rates but in extreme situations may be operated at different resolutions.

One may also consider the physical nature of the user's body. Consider the human factors of head rotation. While very fast rotation about the neck is possible, the degree of rotation is limited. It is not possible to rotate your head completely. To look behind you requires a body rotation which is much slower than a head rotation. Also sudden reversal of rotation is limited by human physical factors, as is vertical rotation. It thus follows that the update rates for parts of the virtual world which cannot be viewed immediately can be reduced, hence reducing the rendering load. In other words the priority of carrying out those updates can be lower. Simulations we have done indicate that considering the maximum head rotation speed can lead to a factor of 2-3 reduction in object rendering load. This is in addition to the order of magnitude reduction due to *Prioritized Rendering*.

It is also relevant that humans cannot tolerate excessive acceleration. Vehicles too cannot change direction very rapidly, so it is possible to use low priority updates for objects behind you and to the side, concentrating on what is ahead of you in the direction of travel. Too much perceived acceleration in a virtual environment induces the disconcerting effect known as vection [Hettinger & Riccio, 1991].

Apart from merely travelling within the virtual world acting as an observer, users will want to manipulate objects within the virtual world and have other intentional interactions. Such manipulations may be achieved through gesture or more directly through handling of objects via glove-like devices. When performing such manipulation, the user is generally concentrating on the task at hand, and not moving much within the virtual world. It is therefore possible for the priority of updates of the objects being manipulated and the objects representing the manipulator, to be increased significantly above those of the rest of the visible scene, ensuring prompt visual feedback. Because in such cases it is known what the focus of the user's attention is, in the extreme case of renderer overload it is possible to concentrate on the important aspects of the displayed scene.

These human factors can certainly be used to improve the effective performance of Virtual Reality implementations using our techniques. While much experimentation is required to determine accurately the degree of benefit of these factors, and especially to allow some more quantitative measures which can be employed directly in *Prioritized Rendering*, some of the general principles are fairly clear, and others will no doubt become apparent.

10 Influence on Future Virtual Reality Applications

The technology of the *Address Recalculation Pipeline* employing the method of *Delayed Viewport Mapping* combined with *Prioritized Rendering* is clearly superior to existing Virtual Reality implementation techniques. It is also much more cost effective than existing systems using extremely powerful and expensive computer graphics equipment. With its clear advantages in latency, reduced rendering costs and graceful overload management, we expect that these techniques will become dominant for Virtual Reality display systems.

With the improved performance available using these techniques it is expected that future Virtual Reality applications will be optimized for this technology. Furthermore, the virtual worlds created for such systems will be characterized by complex scenes with highly realistic backgrounds and large numbers of independent objects. Conventional Virtual Reality systems suffer from unacceptable performance degradation with this kind of virtual world.

The technology also is directly applicable to telerobotic applications such as those found in controlling remote robotic manipulators in space or under sea. It is also ideal for Augmented Reality applications where virtual world images are superimposed on the real world. There is also much scope for our techniques to be used in future interactive television systems, and elsewhere in the entertainment industry.

Conclusion

By changing dramatically the way one views the virtual world, one can take advantage of techniques such as *Delayed Viewport Mapping*, image composition and *Prioritized Rendering* to increase the performance of Virtual Reality computer systems. A novel but affordable hardware architecture, *Address Recalculation Pipeline*, was employed to implement an extremely low latency Virtual Reality display system. In itself this can eliminate rotational latency, however to exploit fully the technology, the structure of Virtual Reality applications needs to be changed.

In this paper an outline of the new display architecture was presented. This set the scene for an outline of how to structure Virtual Reality applications so as to take advantage of this architecture, and the method of *Prioritized Rendering* which can exploit it effectively. The techniques described can enable Virtual Reality applications to be run on economical hardware, and in fact outperform very expensive conventional graphics computers.

The possibilities for yet further improvement to be obtained by considering the human factors concerning the interaction between the user of the system and the virtual world were also examined. It would appear that these human factors can allow further substantial improvements in performance.

Acknowledgements

Matthew Regan acknowledges the support of an Australian Postgraduate Research Award. The prototype hardware was funded by an Australian Research Council small research grant.

References

- [Fuchs et al., 1989] Fuchs, Henry et al. (1989) Pixel-Planes 5: A Heterogeneous Multiprocessor Graphics System Using Processor Enhanced Memories. Proceedings of SIGGRAPH 89. In *Computer Graphics*, Annual Conference Series 1989. pp. 79-88.
- [Hettinger & Riccio, 1991] Hettinger, Lawrence and Riccio, Gary (1991) Visually Induced Motion Sickness in Virtual Reality Systems: Implications for training and Mission Rehearsal, Presented at a DoD sponsored Interagency Tech Simulation, 1-3 Oct, 1991.
- [Molnar & Fuchs, 1990] Molnar, Steven and Fuchs, Henry. (1990) Advanced Raster Graphics Architectures. Chapter 18, *Computer Graphics*, Foley and van Dam, pp. 872-873.
- [Molnar, 1991] Molnar, Steven, (1991) Image Composition Architectures for Real-Time image Generation. Ph.D. dissertation, University of North Carolina, 1991.
- [Regan & Pose, 1993] Regan, Matthew and Pose, Ronald (1993) An Interactive Graphics Display Architecture. Proceedings of IEEE Virtual Reality Annual International Symposium. (18-22 September 1993, Seattle USA), pp. 293-299.
- [Regan & Pose, 1994] Regan, Matthew and Pose, Ronald (1994) Priority Rendering with a Virtual Reality Address Recalculation Pipeline. Proceedings of SIGGRAPH 94. In *Computer Graphics*, Annual Conference Series 1994. pp. 155-162.