An Architecture for Orientation Mapping Post Rendering

Matthew Regan and Ronald Pose

Department of Computer Science,
Monash University,
Wellington Rd,
Clayton, Victoria 3168,
Australia.

Email: regan@bruce.cs.monash.edu.au
rdp@bruce.cs.monash.edu.au

Telephone: +61 3 565 5203
Fax: + 61 3 565 5146

Abstract.

In recent times Virtual Reality has become a growing area within computer graphics with many and varied applications. Many systems suffer from inadequate performance resulting in a lack of realism and even psychological side effects[1]. A large body of research exists to improve the performance of Virtual Reality systems by increasing the rate at which scenes are drawn. This project differs in that the display controller is redesigned so as to cater directly for the needs of a Virtual Reality display system.

By performing complex computation within the display controller as a pixel is being displayed, latency to user head rotation may be directly reduced and latency to translations and stereoscopy may be indirectly reduced. This architecture uses image composition[2] in a unique way to increase the rendered lifetime of an object and reduce rendering loads. In many cases this display controller architecture allows the creation and maintenance of scenes of higher complexity than an equivalent rendering engine without the added display controller hardware.

1.0 Introduction.

This paper introduces and briefly investigates a new computer graphics display controller architecture which performs orientation viewport mapping after a scene has been rendered, as the pixels are being sent to the output device. Delaying the mapping has a drastic effect on perceived latency. A prototype system is currently under construction based on this architecture. This architecture differs from standard display controllers in the nature of the video memory addressing mechanism. Rather than fetching pixels for display from the video memory sequentially, the fetch mechanism bases the pixel fetch location on the pixel's actual screen location (as seen through the wide angle viewing lenses) and the orientation of the user's head. This means the viewport mapping is delayed until after the scene has been rendered.

The new display controller directly reduces latency to user head rotations by fixing user head rotation latency to the refresh period of the display device. Latency to other forms of interaction such as user translations through the virtual scene, head motion parallax and stereoscopy are indirectly reduced through the use of image overlaying.

Image overlaying or image composition [2] is a technique often used by low-cost non-head mounted display graphics systems such as video games to increase the apparent realness of a scene. Different sections of the visible scene are drawn into separate display memories then overlayed to form a final scene (a similar technique is often used when creating cartoons to reduce the drawing required per display frame). When translating through the virtual scene, each display memory may be updated independently, where close objects receive more of the rendering engines time than less frequently changing background objects. Image overlaying generally does not work very well with conventional display controllers in VR systems as user head rotations invalidate all display memories including those containing background images. This mass invalidation occurs because all of the images in the display memories are rendered with a fixed viewport mapping so when the viewport changes due to a user head rotation all of the images have an incorrect viewport mapping. The new display controller does not suffer the same drawback as the rendered scenes in display memory are independent of the viewport mapping orientation, hence they do not become invalid when the viewport mapping changes[1].

2.0 Orientation viewport mapping post rendering.

Display generation usually consists of two processes. The first process called rendering draws images into display memory. The second process sends the image in the display memory to the display device. This process is performed by the display controller. Such a display system is shown in Figure 2.1.

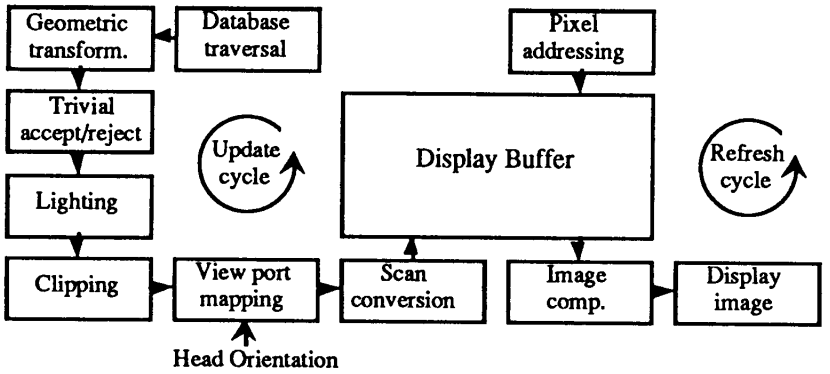


Figure 2.1 A typical display system.

Figure 2.1 clearly shows two cycles, the update cycle and the refresh cycle. An update cycle has occurred when all required primitives in a scene have been drawn into display memory, where as a refresh cycle has occurred when an image in the display memory has been sent to the display device. Conventional VR display systems bind the latency to user head rotations to the period of the update

[1]The only exception being translations caused by the eyes being offset from the centre of rotation of the head. This means stereoscopy is not independent of orientation and as such require some re-rendering. A mechanism has been included to minimise the re-rendering for stereoscopy and the amount of re-rendering required is often small.

cycle. While this has the obvious advantage of not requiring any specialised hardware, latency tends to suffer as the update cycle period is usually lengthy, unpredictable and dependent on scene complexity.

If we modify the architecture of the display controller it becomes possible to bind rotational latency to the refresh cycle period which tends to be short, fixed in length and independent of scene complexity. In figure 2.1 the users head orientation was introduced in the update cycle, while figure 2.2 depicts a display architecture in which the latency to user head orientation is bound to the refresh cycle period.

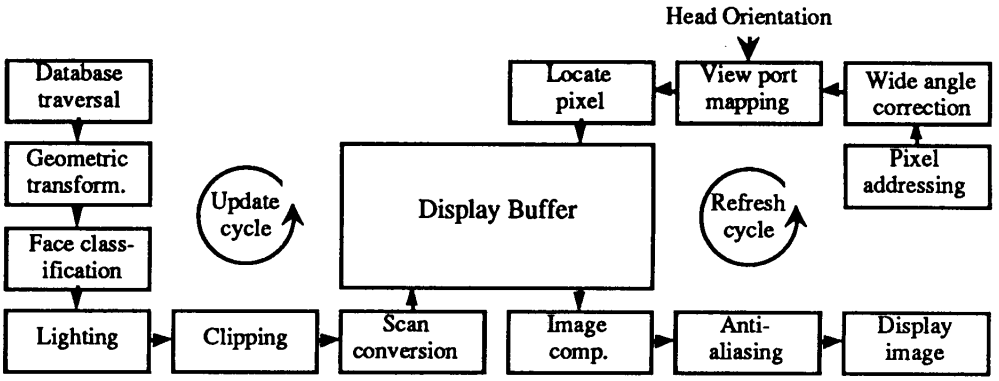


Figure 2.2 A model of the new display system.

Although many projects focus on improving latency by reducing the period of the update cycle [3], or providing faster access to the display [4], few projects alter the architecture of the display controller.

Viewport mapping is performed post rendering so the image in display memory must be as independent of user head orientation as possible. This means the image must completely encapsulate the users head. Within this architecture, the surface of a cube centred at the user's head was the chosen topology for storing images within the display memory. Figure 2.3 shows an image as it is stored in display memory as well as two arbitrary views generated at different orientations.

2.1 Display Memory.

The display memory must have enough information to create a scene from any given orientation. A requirement of the display memory is that it must completely encapsulate the user's head. The first most apparent solution is to use a spherical encapsulation surface with unit radius where lines of longitude and latitude mapped to the rows and columns of the display memory. While this approach works it has some drawbacks. First, expensive Digital Signal Processing devices are required for vector conversion, secondly pixel sizes vary greatly depending upon the pixel's position on the surface of the sphere, resulting in inefficient usage of video memory.

The topology chosen to encapsulate the users head within this architecture is the surface of a cuboid with side length of two units, centred at the origin with the faces perpendicular to the major axes of the world co-ordinate system. This means the display memory is divided into six faces where each face contains a two dimensional grid of pixels. When folded correctly these faces form a complete surface around the user's head.

A run-time advantage of selecting the surface of a cube over the surface of a sphere is that the scan conversion process for the cube requires minor modifications to existing scan conversion/shading algorithms while scan conversion for the spherical surface requires major, computationally expensive changes to common algorithms.

The term display memory is used in preference to frame buffer, as the display memory does not contain a one to one mapping of what is viewed on the display device.

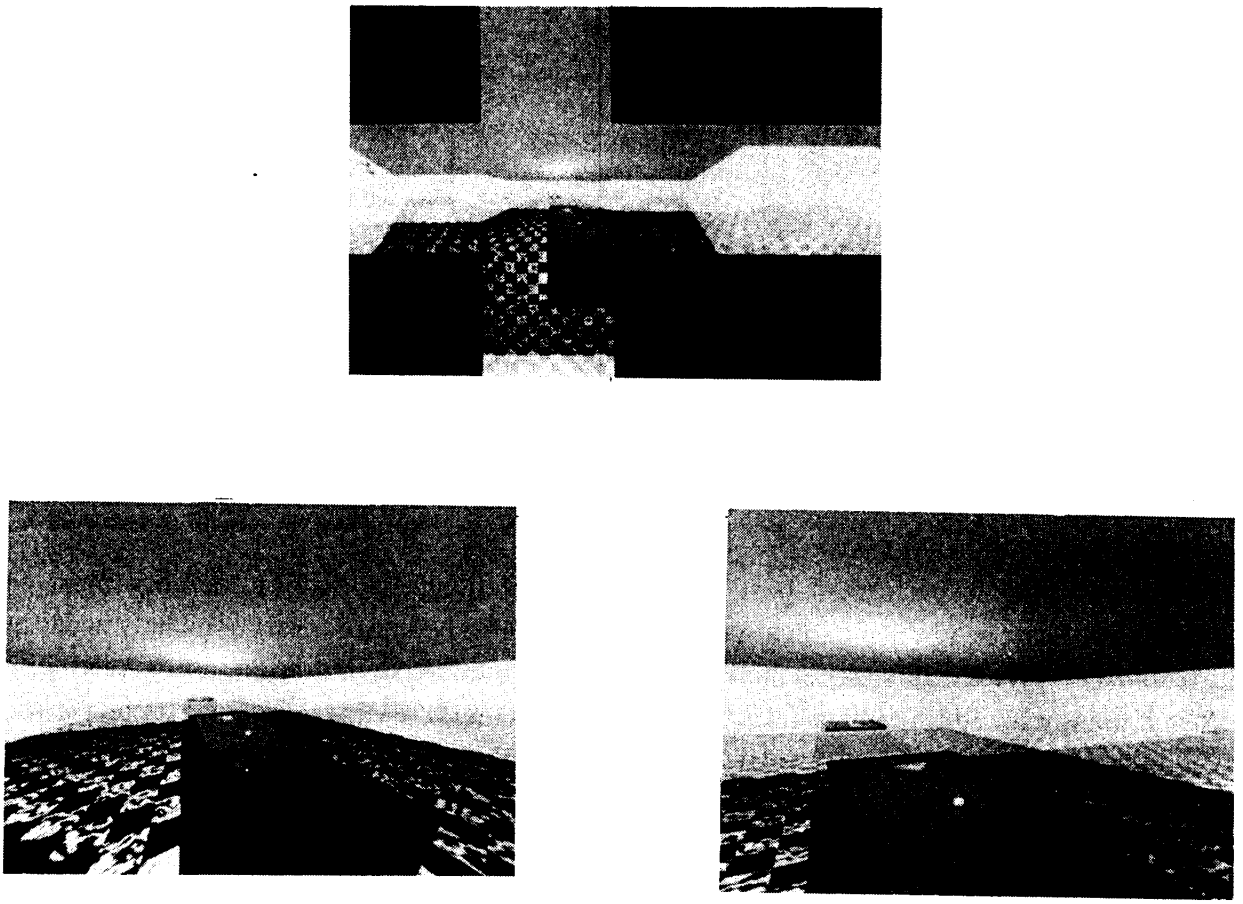
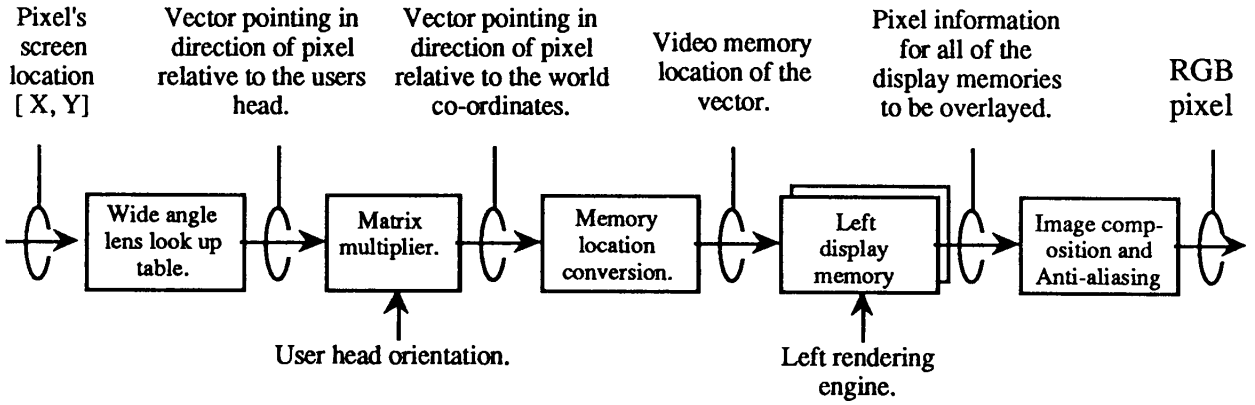


Figure 2.3 An image stored in display memory and two arbitrary views.

3.0 The Address Recalculation Pipeline (ARP).

The mechanism for allowing orientation viewport mapping after the scene has been rendered is provided by the address recalculation pipeline. Traditionally the display controllers performs relatively simple computations, however the operations performed by the controller may be made more complex provided the computation may be pipelined. For current generation medium resolution displays the pipeline hardware on the controller must have a throughput of one pixel every ≈40 nanoseconds. The system currently under construction at Monash University has a pipeline stage latency of 40 nanoseconds.

The purpose of the address recalculation pipeline is to provide non-sequential video memory addressing mechanisms which automatically correct for wide angle viewing lenses and user head orientation. The pipeline orientates the address generation process within display controller with the orientation of the user's head as the pixels are being fetched from video memory. Due to the nature of the computation, the pipeline has been dubbed the Address Recalculation Pipeline (ARP).



Left display pipeline.

Figure 3.1 One half of the Address Recalculation Pipeline.

Figure 3.1 gives a block diagram of the ARP for the left display device. The first input to the pipeline is the screen location of the current pixel being fetched. A look up table converts this screen location into a three dimensional cartesian vector pointing in the direction at which the pixel is seen relative to the user's head. This vector is then multiplied by a matrix containing user head orientation information which results in another three dimensional vector pointing in the direction at which the pixel is seen relative to the world co-ordinate system. The vector is then converted into a two dimensional video memory location. The pixels at the resultant memory location are fetched from the multiple display memories and composed before being anti-aliased. Finally the RGB pixel to be displayed is produced. Both the left and right displays have an address recalculation pipeline.

The first stage of the ARP must convert a pixel's screen location into a unit cartesian vector pointing in the direction at which the pixel is seen by the user relative to the head mounted display. Due to the complexity of such a transformation (especially when unusual wide angle lenses are used [5,6]) a look up table is used for this stage. Many wide angle viewing lenses preserve a standard viewport mapping, however if a special mapping is required for higher fields of view, the look up table may be loaded with a new lens mapping and the lens distortion may be compensated for without run-time penalty. Figure 3.2 shows the effect of different lens mappings on the one scene. Having such flexibility in the configuration of the wide angle viewing lenses make the system largely independent of manufacturer's standards.

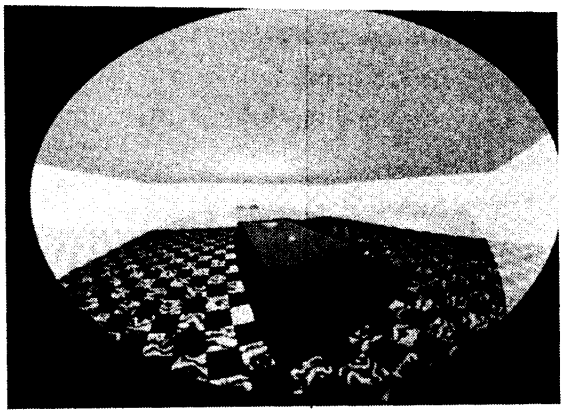


Figure 3.2 Lens distortion mapping in the wide angle viewing lens look-up table.
The matrix multiplier stage consist of two matrix multipliers (one per eye) and are used to orientate the user's head orientation to the world orientation of the display memory. The precise computation is depicted in Figure 3.3.

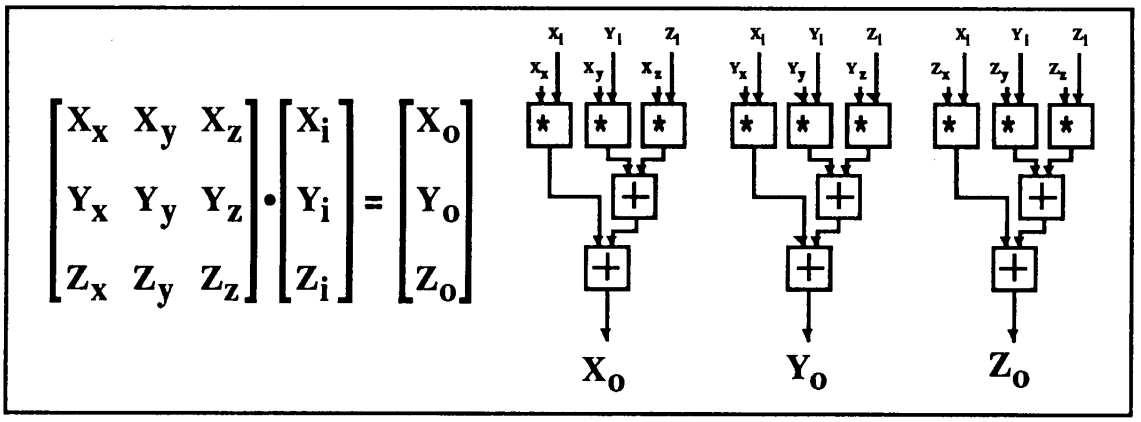


Figure 3.3 One of the matrix multipliers.

The unit vector $[\ X_i\ \ Y_i\ \ Z_i\ ]^T$ points in the direction at which the pixel is seen by the user relative to the head mounted display. The matrix on the left hand side of the multiplication in Figure 3.3 contains three orthogonal unit vectors orientated in the direction of the user's head relative to the world coordinate system and is supplied directly from the user head tracking equipment at the start of each refresh frame. Once the orientation matrix is multiplied by the pixel direction vector a resulting output vector is produced. This output vector $[\ X_o\ Y_o\ Z_o]^T$ is a unit vector pointing in the direction at which the pixel is seen by the user relative to the world coordinate system.

The vector conversion stage within the pipeline must convert a 3D unit vector into a display memory location ie. a point on the surface of a cube. This conversion involves computing the point at which the ray intersects with the surface of a cube. Aligning the cube to the axis of the co-ordinate system such that each face of the cube has one of its X, Y or Z co-ordinates fixed at ±1 means that the point of intersection may be computed with a division followed by a set of range checks. For example if y/x and z/x are both in the range (-1.0 1.0) the ray may intersect with only two of the six faces. The sign of x then determines which of the two faces the ray intersects, with the point of intersection being (y/x, z/x) on the face. The vector converter is depicted in figure 3.4.

Once a display memory location has been computed data may be read from the display memory. As the image composition technique is to be employed within the prototype, multiple display memories are fetched simultaneously. The pixel's Z-buffer component from each of the display memories are compared with the closest pixel being displayed.

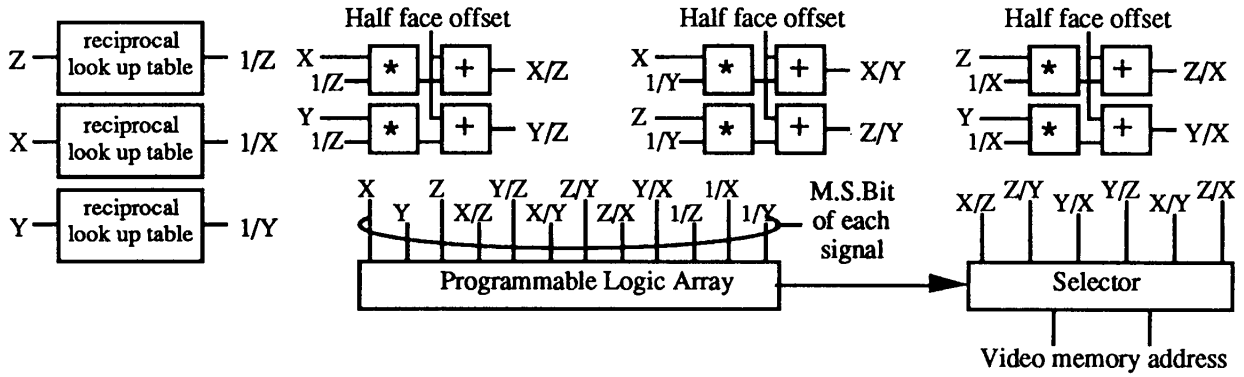


Figure 3.4 Block diagram of one of the vector converters (some pipeline delay registers have been removed for clarity).

Anti-aliasing final display images is quite important in this architecture due to edge effects caused by double sampling (first in the original rendering then again in the viewport mapping). So as well as the typical staircased effect for edges running at an angle, edges which are horizontal or vertical also break up. The breaking up of horizontal of vertical edges seem to be more noticeable than the staircasing of edges running at angle. The solution chosen for this architecture is to fetch a block of four adjacent pixels simultaneously then apply some smoothing filter. The nature of the filter is determined by the redundant bits in the computation of intersection. That is, the position of intersection between the ray cast and the surface of the cube is known to sub-pixel accuracy, so the sub-pixel information is used to apply a filter. The prototype system is to use a linearly interpolating filter which is relatively simple to implement and has adequate low pass filtering properties.

4.0 Image Composition and Fast Translations.

Image composition [2] is a technique whereby multiple display memories are fetched simultaneously with the same pixel location and the pixel with the smallest value in its Z-buffer is displayed. The Z-value comparison is usually performed by a tree comparator. In this application, different objects may be rendered into different display memories. These objects may then be re-rendered *independently* as required. In some circumstances a background scene may require no re-rendering (in such a case the background may be rendered using a high quality rendering technique such as ray-tracing). In actual practice the performance improvement obtained by image composition depends on how evenly objects may be clustered by their required update rate for a finite number of display memories. Unfortunately our project budget restricts the prototype system to three display memories per eye (A total of six display memories).

The mechanism by which image composition improves translation is often misunderstood. Rendering to six faces of a cube instead of one display plane increases the load on the scan conversion hardware. This load may however be offset by the reduced rendering requirement.

When translating through a scene, the rate at which an object update must occur is dependent on several factors such as the distance of the object, the relative velocity of the object, the size of the object, the angle between the path of the user to the object and the relative velocity of the object and the display resolution of the system. The net result is that many objects within a scene require updating less than a few times a second (they only change position or size enough to warrant re-rendering a few times a second). When compared to a conventional system where each object must be redrawn every display frame the saving may be immense, resulting in more realistic scenes being realised from the same rendering hardware. Latency to translation is often less noticeable than latency to rotations as many translations are the result of a user gesture such as pointing a finger or pushing an accelerator rather than actual physical motion. Providing an adequate update rate for translation however is important to provide a sense of fluid motion. We are currently examining many update models so different objects may be re-rendered to different display memories in order to reduce the overall load on the rendering engine. Clustering algorithms are being examined to tie objects with a similar update rate to the same display memory.

5.0 Expected Latency.

Latency to head rotations (except stereoscopy) is fixed to the display update rate. This means the eye may track an object as the head rotates with little detectable latency. A display update rate of 60 Hz combined with a head tracking update rate of 120 Hz results in an average rotational latency of approximately 20 ms. Forward prediction of head location using a linear or higher order models [7] may reduce this latency even further. As the viewport mapping process does not have to pass through the double buffer display switching (as it independent of rendering), the time at which the raster passes through the centre of an image is precisely known. This in turn improving the quality of the head predication process.

Providing a binocular stereoscopic view of the scene may require some objects to be re-rendered. Head rotations cause stereoscopy to diminish as the viewport moves from its original orientation (the orientation for which the stereoscopy was rendered), however due to the nominal intraocular distance of 6.3 cm for humans, only relatively close objects translate significantly, so image composition allows these objects to be re-rendered independently. Added to the fact that the human visual system takes several tens of milliseconds to perceive a stereo scene, and apparently is not sensitive to small temporal errors [8]. The latency involved in re-rendering for stereoscopy is important but not critical to system performance.

6.0 Conclusion.

This paper outlines a technique for redesigning a display controller within a traditional graphics display system. By performing complicated computations within the display controller the system may be optimised for the needs of a Virtual Reality display system. Wide angle viewing lens effects and response to user head rotations may be compensated for within the display controller, avoiding the rendering bottleneck. The architecture allows image overlaying to provide more realistic scenes and a mechanism for focussing the rendering engines time on sections of the scene which change the most frequently. The implicit rendering overload strategy is more desirable than many current

systems. When rendering loads are high, the apparent fluidity of the animations suffer rather than latency to user gestures.

A prototype system is currently under construction and is expected to be finished by December 1993. The prototype is being built with off the shelf technology and is expected to cost under US$ 20,000.

Acknowledgements.

Mr. Matthew Regan acknowledges the support provided by an Australian Postgraduate Research Award (APRA). Dr. Ronald Pose acknowledges the support of an Australian Research Council (ARC) grant to assist in the implementation of prototype display hardware.

References.

[1] Deyo, R., and D. Ingebretson, "Notes on Real-Time Vehicle Simulation," in implementing and interacting with Real-Time Microworlds, Course Notes 29 for SIGGRAPH 89, Boston, MA,
August 1989.

[2] Molnar, S., "Image Composition Architectures for Real-Time image Generation" Ph.D dissertation, 1991.

[3] Fuchs, H., et al. "Pixel-Planes 5: A Heterogeneous Multiprocessor Graphics System Using Processor Enhanced Memories," SIGGRAPH 89, 79-88.

[4] Whitton, M. C., "Memory Design for Raster Graphics Displays." IEEE Computer Graphics and Applications, Vol. 4, No. 3, March 1984, pp. 48-65

[5] Robinett, W., and Roland, J., "A Computational Model for the Stereoscopic Optics for Head Mounted Display". In Presence 1, (winter 1992), 45-62.

[6] Deering, M., "High Resolution Virtual Reality", SIGGRAPH 92, 195-202.

[7] Friedmann, M., Starner, T., and Pentland, A., "Device Synchronisation Using an Optimal Linear Filter". In Proceedings of the ACM Symposium on Interactive 3D graphics (Cambridge, Massachusetts, March 29 - April 1, 1992), 57-62.

[8] Lipton, L. "Temporal Artifacts in Field-Sequential Stereoscopic Displays. Proceedings of SID 91 (Anaheim, California, May 6-10, 1991)., 834-835.