- Shared rendering for multiple display devices

  Systems must be capable of driving more than one display device to allow use with stereographic displays. It is beneficial if some of the computation involved in rendering is shared between all display devices.

- Scalability

  Rendering systems should be scalable to allow them to be configured for a wide range of applications.

rendering system cannot know the characteristics of the end users' applications, flexibility and programmability are essential.

The author believes the rendering system should include the following facilities:

- Texture mapping

  As pointed out above, texture mapping allows a reduction in model complexity and provides a large number of visual cues.

- Culling acceleration

  The system should include hardware (or a specialised processor) to perform culling.

- Gouraud shading

  Gouraud shading improves the image quality drastically and is particularly important for 'hiding' edges in tessellated objects.

- An efficient host interface

  The rendering hardware must communicate with the host in an efficient manner. Systems which require the host to send each primitive explicitly (such as the Silicon Graphics VGX) place too much work on the host.

- High level graphics libraries

  The application should communicate with the rendering hardware through a high level graphics library which has knowledge about the graphics hardware. In this way the application can make efficient use of the rendering hardware without the programmer having to perform hardware specific optimisations.

- Correction for optical distortion

  Correction for distortion in the optics of a head mounted display should be included in the rendering or display system.

7.3 Recommendations

Three major architectural factors would appear to be useful in meeting the specific needs of a VR rendering system:

- Image composition

  Image composition allows a system to be scaled up almost without limit, leading to the ability to render complex models while maintaining a high redraw rate and low latency. Image composition architectures tend to be highly flexible, allowing the user to commission a system tuned to his application. Image composition is just one form of parallelism, many of which prove to be useful in rendering systems.

- Logic enhanced memory

  The rendering capabilities of many systems are limited by the bandwidth of the connection between the frame buffer and the last stage of rendering; logic enhanced memory is one of the few methods of avoiding this bottleneck. Its form varies from the relatively complex implementation, incorporating much of the rendering system, used in Pixel-Planes and PixelFlow to the relatively simple enhancements made in FBRAM. Both variations show the effectiveness of the technique; the author believes that in the future, more sections of the rendering system should be integrated with the frame buffer memory.

- Highly programmable pipeline stages

  One of the characteristics of pipelined systems is that each stage is usually highly inflexible; this leads to the fact that pipelines are usually only efficiently used for a very small range of applications. Highly programmable pipeline stages (exemplified by LeoDraw and the Pixel-Planes renderers) offer high levels of flexibility by allowing the system to be optimised for a particular application, or class of applications. Since, in almost all cases, the designer of the

A important observation from this research is that the rendering system is not just the rendering hardware, but is the combination of rendering hardware, software and host system which must work together efficiently to form a useful virtual reality system.

7.2 Future work

Although there has been much work carried out on culling algorithms, the author is not aware of any research on the way in which culling may be implemented in hardware or the effect it has on the predictability of the rendering system.

In a system with no culling algorithm the workload on the rendering system is nearly constant since it must be able to render the entire model description in each frame. Unfortunately culling algorithms tend to be highly unpredictable and thus it is difficult to maintain a constant frame rate when the algorithm may be able to cull large amounts of the model out on one frame but very little on the next.

Research is needed to produce *predictable* culling algorithms. As well as improving culling algorithms, this could produce a tool for the application designer which could allow him to analyse his model and extract worst case complexity information. This tool could also help the designer identify particularly complex parts of his model which could be simplified to improve the redraw rate.

While the author has suggested a culling algorithm which could be implemented in hardware, he also believes that research is needed into hardware implementations of the other algorithms. Traversal of the grid cell structure described in section **5.5.2.1** can be accomplished using a modified form of Pineda's algorithm **[Broo94]**, and the author believes that this is especially suitable for implementation in hardware.

mation to make optimal use of the graphics hardware. The Performer library comes close to meeting these goals.

The requirements of VR applications on the host are too great to be met by one processor and probably always will be, since the demands of VR are growing as fast (if not faster) than the supporting technology. For this reason, the operating system and graphics libraries used must be able to make efficient use of multiprocessing hosts; the combination of IRIX REACT and Performer is one example of a combination that allows efficient use of multiprocessing.

A problem with culling algorithms is that they can create even more load for the host processor. **Chapter 5** presented a novel culling algorithm which is suitable for implementation in hardware or a simple microcoded processor and thus does not increase the loading on the host. Simulations of the algorithm show that, under suitable circumstances, it can be highly effective.

In practice culling algorithms are really simplified hidden surface and clipping algorithms. The final clipping and hidden surface algorithms have strong constraints which ensure that only visible primitives are displayed. In contrast the algorithms used in culling have similar goals but have much lighter constraints; they aim to remove primitives which are not visible (to accelerate later stages), but their only constraint is that they do not remove primitives which *are* potentially visible. When considering the research presented in the thesis, the author feels that culling algorithms should not be regarded as a completely separate issue from clipping and hidden surface removal but that, instead, the whole set of culling, clipping and hidden surface removal algorithms should be treated as a set of algorithms which work together to provide the correct image in the most efficient manner.

provides a sufficiently high quality image for most tasks and for this reason the author believes that the cost of Phong shading is unnecessary for many applications.

Texture mapping appears to be an invaluable tool for both reducing model complexity and for producing visual cues which (among other benefits) helps the user detect movement. These reasons convince the author that texture mapping is highly desirable in a rendering system's hardware.

In the author's experience, shadows provide a highly effective means of enhancing the user's perception of the scene. Many complex and confusing scenes become easily understandable with the addition of shadows. However most applications require that only one or two objects in a scene cast shadows and in this case the shadows can be simulated by the addition of extra polygons to the scene description. For this reason the author does not recommend the use of additional hardware (or extensive support in software) for shadow generation in the rendering stage. Note that a problem that can occur in this situation is that the addition of one shadow can make the user aware of the lack of shadows in the rest of the scene.

Virtual Reality applications vary greatly in their use of the rendering system, between applications, between frames, and even within one frame. Without flexibility it is unlikely that a rendering system would be effective for more than one or two applications. Highly programmable pipeline stages (such as the *LeoDraw* geometry system used in *Leo* described in section **6.1.2** and the Pixel-Planes rendering hardware described in section **6.1.7.1**) are one way of providing this flexibility. Some systems use general purpose microprocessors for flexibility, however in general this approach can not deliver the required performance since the processor is not optimised for graphics applications and does not provide support for parallel operation.

A good graphics library must use knowledge about the graphics hardware and allow the programmer to pass information about the application; it should then use all this infor-

as Unix) are plagued by disturbing pauses and interruptions. The rendering hardware can be made more predictable by the inclusion of large FIFOs to reduce the variation in rendering time due to the order of rendering. When producing parallel rasterisation hardware care must be taken to accelerate long narrow primitives as well as short wide primitives, failure to do this results in an unpredictable rendering capacity.

Most systems do not specifically address the issue of latency, however a conventional Z buffer based system has a low latency compared with one which requires primitives to be sorted before rendering. Systems which require sorting (such as PixelFlow) often reduce their latency by using each stage of the rendering pipeline for only a fraction of the frame time which is extremely inefficient. The only system to address latency effectively is the Regan and Pose address recalculate pipeline; however the effectiveness of this system is limited to reducing latency due to head rotation, not translation. This technique will reduce in usefulness as the working range of position trackers increases allowing the user to move around a large space. This system also does not address latency from other input devices, such as position sensors on the user's hands. It is worth noting that the rendering system is just one source of latency in the entire system; the latency due to the input devices and host computer may well be higher than the latency of the rendering system. All these latencies are added together in the latency of the entire system.

The comparison given in Figure 3.10 and Figure 3.11 between flat and Gouraud shading, has convinced the author that Gouraud shading is desirable in many applications; even though it requires significant extra processing. This comparisons shows that the edges visible on a flat shaded curved surface are extremely noticeable when the object is moving; which is likely to reduce the user's ability to use the VR system effectively. The author's experience of using the Division dVs system and systems developed in the Advanced Interfaces Group which use Gouraud shading, indicate that Gouraud shading

- Correction for distortion in the display device

  The optics in many head sets distort the image. The rendering system should be able to compensate for this.

- Support for dynamic models

  The rendering system should not restrict the user from modifying the model.

- Freedom of viewing parameters and model geometry

  The rendering system should not restrict the user's viewpoint or the geometry of the model.

- Flexibility

  Flexibility is essential in efficiently supporting a wide range of applications.

- Image quality and realism sufficient to perform the task

  The rendering system should produce images of high enough quality to enable the user to perform his task efficiently. It is not necessary to produce the most realistic images possible unless anything less would effect the user's work.

In the author's experience of using the Division dVs system and systems developed in the Advanced Interfaces Group, high and consistent redraw rates, low latency and sufficient image quality are the most important factors in producing a usable system.

High redraw rates are now possible, even on complex models, using massively parallel rendering systems (such as PixelFlow). These systems are however expensive and thus not suitable for many applications.

Graphics libraries (such as Performer) designed specifically for visual simulation tasks, help produce a constant frame rate through the use of level-of-detail switching and stress management. A constant frame rate also requires predictable behaviour from the operating system executing on the host. Systems using conventional operating systems (such

Chapter 7

Conclusions and Future work

This chapter summarises the work described in this thesis and suggests some areas of research which would be useful future work.

7.1 Conclusions

Chapter 4 described the requirements of a rendering system that is to be used in a Virtual Reality system. These are:

- A high and constant redraw rate

  A high redraw rate is essential in attaining a low latency, smooth movement and no temporal aliasing. Variable redraw rates are disturbing to the user.

- A low latency

  High latencies can cause stationary objects to appear to move around in the virtual world and can also cause nausea.

- Images consistent with the real world

  In augmented reality systems objects in the virtual world must appear in the correct place when overlaid on the real world. In a normal Virtual Reality system, it is important that images are generated from the correct viewpoints to ensure that the viewer's perception of depth is correct.

- The ability to generate multiple images

  Rendering systems must be capable of generating multiple images simultaneously to support stereoscopic display devices.