Chapter 4

Specialist requirements of VR rendering systems

This chapter discusses ways in which the requirements of VR rendering systems differ from conventional rendering systems.

4.1 Conventional rendering systems

Existing systems can be classified as follows:

- Systems designed to produce high image quality without regard to the time taken to create the image

  These systems are typically used for generating high quality still images or for rendering high quality animations for films and television. While producing these high quality animations, it is not necessary to produce the images at the same rate as which they will be displayed. The time taken to render these images is limited only by the cost of computer time (often on powerful - and thus expensive - super-computing facilities), and the practicality of the time taken to render the whole animation.

- Systems designed for ‘interactive’ use

  The term ‘interactive’ has always meant rendering graphics much faster than the systems described above - how much faster has changed dramatically over the last decade or so. At first an ‘interactive’ system meant that the image could be generated in a few seconds. Interaction in modern systems usually means an update rate fast enough to allow a user to drag an object around the display. The update rate of these systems is often highly unpredictable.

- Restricted systems

  By restricting the viewing position and contents of the scene, it is possible to optimise the rendering software and hardware. This method was used in early systems for molecular display, which provided highly accelerated sphere rendering. The most common use of restricted rendering systems today is in computer entertainment software, which must provide complex graphics on low cost hardware.

The primary goals for most VR rendering applications are to display large amounts of information in an accurate and natural manner. It is also necessary to avoid anything which distracts the user from the task at hand or is annoying to the user.

4.2 High and constant redraw rate

A high and constant redraw rate is one of the most important requirements of a VR rendering system. In **[AiRB90]** it is stated that, below around 6 frames per second (fps), it is difficult for a user to navigate around a house or three dimensional scene without the aid of a two dimensional plan. They state that ease of interaction increases as the frame rate rises from 6 fps to around 20 fps, at which point the increases in ease of interaction, levels off. Redraw rate is more critical in VR applications because the user is relying very heavily on the image. In a non-immersive system the user can always do something else while waiting for an image to redraw. In an immersive VR system the user cannot do anything except look at the image. A high redraw rate also helps to reduce latency which is another important factor in the usability of a VR system and is described in section **4.3**.

Another reason for having a high redraw rate is to reduce *temporal aliasing* problems. The most common example of this, is the effect which causes a wheel to be seen as spinning backwards on film. This happens because the image is not updated often enough. Figure 4.1 shows an object being rotated. The diagram was generated by rotating each object through 10° anticlockwise to generate the next object. If these images were displayed sequentially then the viewer would probably perceive a wheel spinning anticlockwise. However an identical set of images could have been generated by rotating the object 100° anticlockwise or 80° clockwise. Thus an object rotating swiftly in one direction may appear to be rotating in the opposite direction. In a driving simulator an evenly

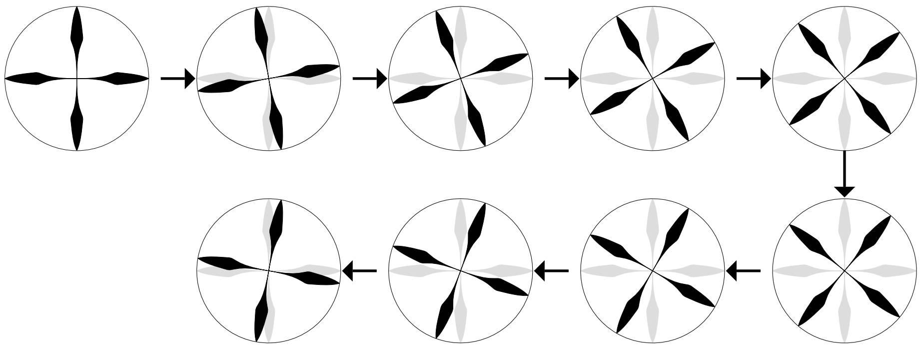

Figure 4.1: Images of a rotating wheel - with potential for temporal aliasing

spaced row of street lights can exhibit similar behaviour - apparently moving in the opposite direction to the car. These effects are extremely distracting but they can be avoided by using a higher refresh rate. The refresh rate must be sufficiently high that the movement between two consecutive frames is only small. An alternative solution to this problem is to introduce motion blur. Motion blur produces effects similar to those seen when a fast moving object is caught on film with a long exposure time.

The perceived redraw time is not continuously variable, but is a multiple of the time taken to refresh the display device. Thus if the display device is refreshed thirty times a second (corresponding to a little over 33ms) no difference is perceived if the rendering system redraws the display in 15ms or 30ms. Similarly if the rendering system takes 34ms to redraw the image the redraw rate has effectively been halved since one redraw is now taking place after two display refreshes. In the author's experience of using a VR system, refresh rate changes are very noticeable. This is disturbing to the user and therefore must be avoided. Some applications work around this problem by using a fixed frame rate which is lower than the frame rate which could be achieved on the less complex frames **[RoHe94]**.

Some rendering systems are susceptible to the orientation of objects in the scene. Looking at the rendering system in **Chapter 3** the cause of this is easy to find. A triangle which does not occupy many scan lines but occupies many pixels on each scan line can be rendered much more quickly than a triangle which occupies many scan lines but only has a few pixels on each. This is because generating spans takes time and there is a significant start up time on the span filler. A primitive with more spans but the same number of pixels as another primitive will take longer to render. Effects like sensitivity to orientation make predicting the performance of a system in a real application very difficult. If a guaranteed refresh rate is required on a system which has such unpredictable performance, it is necessary to create a scene of much lower complexity than would be required on a system which is not sensitive to orientation. **[Kyri93]** creates a measure of orientation sensitivity (the *sliver ratio*) which is close to unity on systems which are insensitive to orientation, and then measures the performance of a number of rendering systems. The Silicon Graphics Extreme has a sliver ratio very close to one (1.0009) while systems such as the Evans and Sutherland ES3100 have ratios over twelve indicating they are highly sensitive to the orientation of primitives.

Another argument for constant redraw rates is that, if the user was viewing a simulation, any variation in redraw rate may be perceived in a change of the simulated behaviour **[Kyri93]**

The highest redraw rate which is practical on a system is limited by the rate at which the display device is refreshed; there is little point in redrawing the image faster than it is being displayed. Thus higher refresh rates allow higher redraw rates, which in turn allows lower latency.

4.3 Low latency

In the real world if a person moves their head the image that they can see instantly follows the movement. In an HMD based virtual world, the action has to pass through the entire virtual reality system before the user sees the effect. The chain of events is:

- The head movement is detected by tracking system
- The tracking system communicates the new head position and orientation to the host. These two elements are discussed in **Chapter 2**
- The operating system on the host services the data transfer from the tracker, and eventually awakens the appropriate process. This is discussed in **Chapter 6** .
- The VR software reads the tracker data, converts it to a convenient form, and notifies the rendering system of the change.
- At the start of the next frame the rendering system creates a new frame based on the new position and orientation information.
- Finally the image is displayed on the display device in the user's HMD.

All this takes time - usually a few hundred milliseconds. The time from a sensor or input device being moved, up to the time when the display is updated to fully correspond to the change in input, is the latency. Similar effects also occur if the user moves an object in the world. It has been suggested that a latency as low as 60ms can affect the user's performance in the virtual world **[Kala93]**.

One of the biggest problems caused by latency is the fact that an object appears to move about in virtual space even when it is actually stationary. This can be confusing - especially if the user is trying to grab the object or if the object is a representation of the user's hand (which he knows the real position of). Figure 4.2 part A shows the user's head at rest - the intended and perceived positions of the object correspond. In part B of the same figure, the user has turned his head, the rendering system has not yet rendered a corrected image and the object appears to have moved to the left. In part C the system has caught up and the object is now perceived to be in its intended position in the virtual world - the object thus appears to have moved right relative to part B. In **[AzBi94]** an example is given of a system with 100ms latency. If the user's head were to move at 100°/second this results in a 10° error. Such an error corresponds to a 5 inch positional error for an object an arm's length away. Besides affecting the ability of the user to perform his task in the virtual world, it may induce nausea.

Latency is a greater problem in augmented reality systems where the virtual and real worlds are combined. Figure 4.2 includes an object in the real world which is visible to the user. Due to latency the virtual object moves back and forwards around the real object. Since the user has something to compare the behaviour of the virtual object with the effect of latency is much worse.

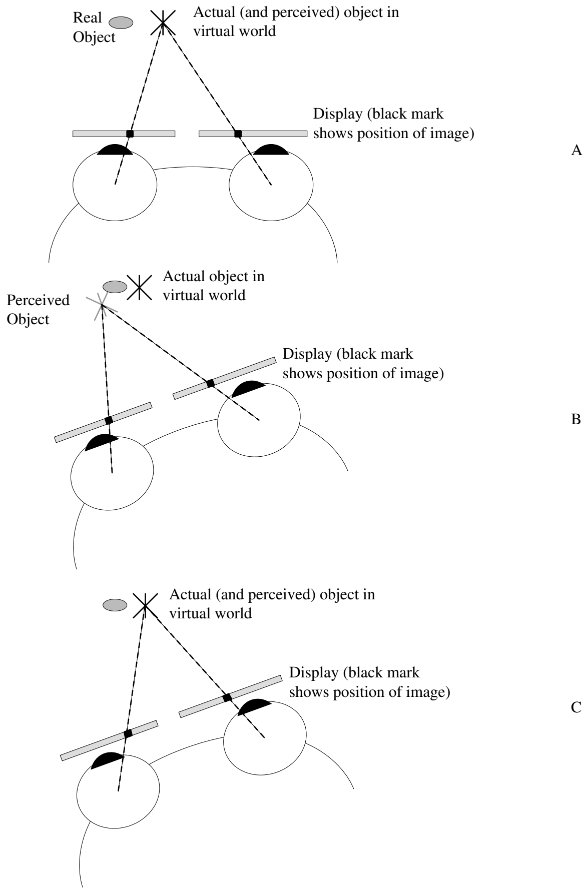

Figure 4.2: The effects of latency in augmented reality

One means of reducing the perceived latency is to predict where the head will be by the time the image is displayed. Some systems use past position and orientation values for prediction, other systems (such as the one presented by Azuma and Bishop in **[AzBi94]**) use gyros and accelerometers to aid in prediction. It is obviously not possible to predict the position of the user's head at any time in the future - the further ahead the prediction is made the more likely it is that errors will be introduced. In **[AzBi94]** it is claimed that the total positional error can be reduced by 5-10 times using prediction aided by gyros and accelerometers.

Regan and Pose propose a system which separates out the orientation of the head from the position sensor and passes it directly to the display system **[RePo93]**. They claim that such a system can reduce the latency due to head rotation down to around 20ms. This system is described in more detail in section **6.1.8**.

Augmented reality systems combine the real world with the virtual world in two different ways. The first method is to use optics to combine the images together. This does not introduce any extra latency. The second method is to use a television camera and combine the image with the computer generated image. Although this may not introduce any extra latency in the images of the virtual world it does in the images of the real world.

So far this discussion has only considered latency due to positional movement. In environments which do not use HMDs latency is still a problem. Considering a flight simulator the time the simulator takes to respond to modifications of the flight controls also constitutes latency. **[CAA89]** states that systems should respond to an abrupt change in pitch, roll and yaw within 150 milliseconds of the time that the aeroplane would actually respond. It also states that the system should not react before the actual aeroplane would react and that any visual changes should not occur until after motion cues have been produced. These requirements are somewhat different to the HMD case. In the flight simulator requirements it is not required that the latency be as low as possible but that it should faithfully mimic the behaviour of the aeroplane. Where an HMD is not being used latency only affects movement of objects and tracked hands etc. - this may prove to be less of a problem than head position latency.

Latency is not just an issue for the display system - delays in updating tactile feedback systems are also undesirable. If a tactile feedback system was being used to help a user ‘catch’ a virtual object, any latency could mean that the object had fallen from his hand before he had the chance to grasp it.

4.4 Consistency with the physical world

The errors described in the section on latency can be regarded as dynamic position tracking errors. **[AzBi94]** discusses both static and dynamic position registration issues in detail. Static registration involves the one-off calibration of the viewing parameters and position of the user in the virtual world. Good calibration ensures that an object of a given length in the virtual world really does appear to be that length and objects at a given distance appear to be correctly placed. Errors in scaling can cause problems if the user tries to reach out for an object. The effects of focusing on the optics of the display system rather than a real object can actually cause objects to scale and appear to be at distances different from their real position. In the case of fighter simulation systems (or much more importantly in head up displays in real aircraft) these scaling errors can cause the pilot to miss his aim.

[Deer92] talks in detail about consistency when using frame sequential stereo displays - some of the comments are generally applicable. One important observation is that although many systems set the viewpoint to be the centre of the eye this does not correspond to the first nodal point of the lens system (which is the correct location of the viewpoint). Since this position is not at the centre of the eye it moves as the eye rotates. **[Deer92]** recommend the use of eye tracking systems to provide a more accurate viewpoint position. The space between the two eyes varies from user to user and must also be correct to produce an accurate rendition of the virtual world. The same paper also discusses methods for reducing errors due to the fact that the CRT is not flat. Deering claims that removing these inaccuracies in positioning improve the perception of depth in the system greatly.

Another consistency issue which arises only in augmented reality systems is ensuring that a virtual object which is moved behind a physical object becomes hidden. In **[BFO92]** ultrasound scans are combined with live image of a patient. The aim of this system is to allow a doctor to see inside the patient - this however was one of the main difficulties. Initial attempts produced the feeling that the ultrasound images were pasted on top of the patient. A later version created a virtual ‘pit’ in the patient in which the ultrasound images were displayed. Objects in the real world (such as the top of the ultrasound probe) would become hidden behind the virtual imagery even though they are actually outside the body. The solution suggested in **[BFO92]** is to use range finding to gain information about depth in the real world - this information could then be used by the display hardware to determine if the real world view or the generated imagery should be displayed at a particular pixel.

4.5 Multiple image generation

In most VR applications it is necessary to generate images from more than one viewpoint either for a stereoscopic display or to drive multiple projectors in wide field of view projection systems. One priority is to ensure that the display devices are all updated at the same time. This requires synchronisation between multiple rendering pipelines and in particular between the different display controllers.

In addition it would be useful if some of the rendering effort could be shared between the views. Referring back to Figure 3.1 the model traversal, tessellation, and modelling transformation stages are independent of the viewpoint and so can be shared among the rendering pipeline of all the views. The trivial accept and reject stage is viewpoint dependent - but if the viewing parameters are close to each other (as they are in stereo rendering) it is possible to generalise the trivial accept and reject stage so that it may also be shared among the views. The back face culling used in the trivial accept and reject stage will probably cull almost the same set of faces for both views in a stereo system; because of this it is possible to share the work between the two views; albeit with a small number of extra polygons being rendered on each view.

Since the viewing frusta for the two views in a stereoscopic display are similar, the view frustum culling can be shared between views by culling to a volume enclosing both frusta.

All stages from lighting onwards are viewpoint dependent and therefore can not be shared between views. Figure 4.3 shows a combined pipeline for rendering a stereoscopic view.

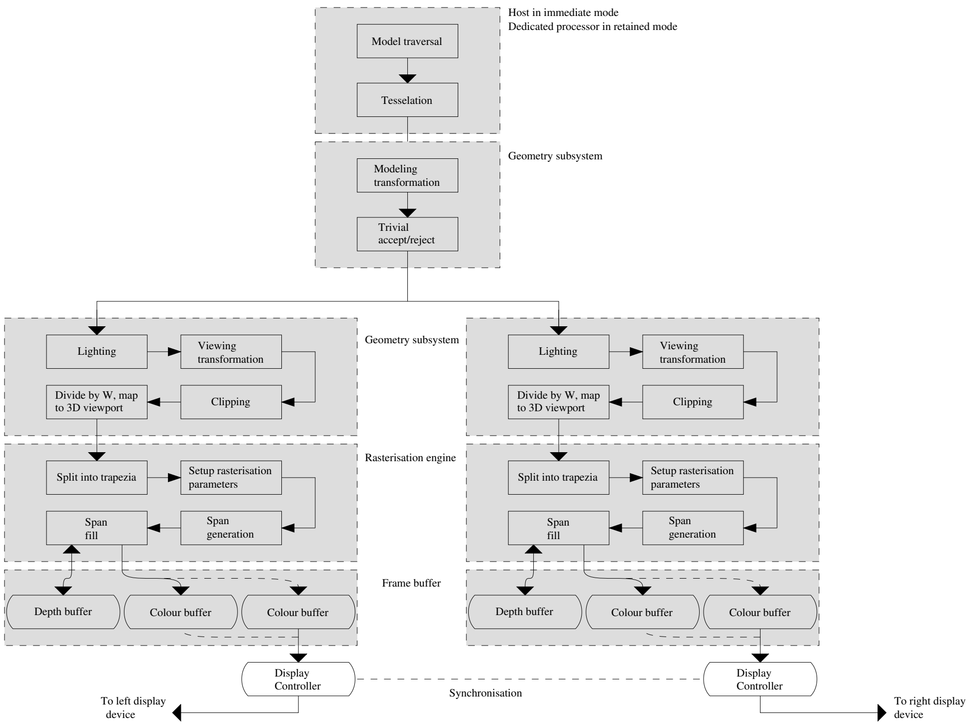

Figure 4.3: A stereoscopic viewing pipeline

4.6 Correction for distortion in the display device

A common source of distortion is in head mounted displays which stretch the image produced by small display devices to produce a wider field of view. One common set of optics used is the 'LEEP (Large Expanse, Extra Perspective)' system **[Kala93]**. This was originally designed to be used with conventional photography with a matching set of lenses on the camera. This did not cause distortion because the images were distorted first in the camera, and then the optics in the display device removed the distortion. When used to display computer generated images it is necessary to distort the images before sending them to the display device. Most rendering systems do not have the ability to produce this distortion because it is computationally expensive. One of the few systems which supports real time correction for wide angle viewing lenses is the system by Matthew Regan and Ronald Pose **[RePo93, RePo94]**. That system uses a large (3 MByte) look-up table to hold a description of the lens distortion which is then used to modify the stream of addresses to the video memory at display time. No run time penalty is incurred by this compensation in the Regan and Pose system.

4.7 Support for dynamic models

In most VR applications the viewpoint changes regularly (usually on every frame); this is especially true in systems based around HMDs. Many researchers have investigated ways of optimising rendering for applications where the model changes relatively rarely and the view point moves frequently. Most of these systems involve preprocessing the model to generate information which can be used to help reduce the number of primitives sent to the rendering pipeline at each frame. Section **5.5.2** describes some of these algorithms in detail.

Many of the preprocessing techniques used to speed up rapid viewpoint changes, rely on a static scene description. These preprocessing techniques (and also the retained mode of operation), incur an overhead when the scene needs to be updated. In applications which change the model frequently, this overhead may be unacceptable.

4.8 Freedom of viewing parameters and model geometry

As pointed out at the start of the chapter, entertainment software often restricts the viewing positions and scene geometry to speed up the rendering system. One example of this is that some recent games are able to provide texture mapping by restricting the head orientation. This greatly simplifies and speeds up the texture mapping process. Many systems restrict the geometry of scenes to make hidden surface removal simpler. A common example is not to allow intersecting objects. If this is done hidden surface removal may be accomplished by plotting objects in back to front order and performing back face cull on each object individually[1]. Other systems do not allow an arbitrary three dimensional scene, for example some systems represent the scene as a two dimensional map, with one item at each location in the map. This item may represent a wall or other object but this is still more restricted than a general three dimensional scene description.

In a conventional rendering application, the viewing position is specified by the application based on commands from the user - in this way the application can restrict viewing position. In an HMD based VR system the viewing parameters comes directly from the user - the application does not get an opportunity to restrict them.

1. Providing that no cyclic pattern of overlap exists.

Some applications may be able to use a restricted scene descriptions - but for a general VR system (which this thesis is considering) arbitrary restrictions are likely to cut out a large range of applications.

4.9 Flexibility

A point raised in section **1.2**, is that the way in which the different types of applications use the rendering system varies greatly. Ideally a VR rendering system has got to be flexible enough to handle many different types of rendering requirement. Some typical flexibility issues include:

- How often the model changes

  In an architectural walk-through, the model is likely to remain almost unchanged for long periods of time. In the visualisation of a time varying simulation the whole data set may change every frame.

- Size of primitives

  Some applications may use large numbers of small primitives, others a few large primitives. The way in which the rendering pipeline is utilised is completely different in the two applications. Even within one application the size of primitives may vary greatly. Considering an architectural walk-through again, when walking down a corridor, the scene would typically consist of a few large polygons representing the walls and some smaller polygons representing doors and door handles etc. On entering a room it may be possible to see a large number of small objects. Within one frame a few large polygons may be used for the floor and walls and then a large number of small polygons may be used for objects in the room.

- Use of hierarchy

  Scientific visualisation applications may display one highly complex object while many other applications might create a deep hierarchy. The demands on the geometry subsystem are quite different even if the number of primitives is otherwise the same.

- Type of primitives

  Some applications may be naturally polygonal, others (such as the CADCentre system mentioned in Chapter 2) use large numbers of curved objects. It could be wasteful to incorporate extra hardware to perform curved surface interpolation in a general VR system if many applications are not going to use it.

In some ways the trend to use pipeline parallelism with large numbers of specialised blocks reduces flexibility. It is not unusual to come across applications which use one section of the pipeline very heavily. Using (multiple) conventional microprocessors to perform rendering does not suffer from this problem, however it is rarely possible to extract high enough performance from such a system.

Most workstations being used today in research establishments, are in themselves fairly flexible. This flexibility does however demand that for a given application parts of a system will typically be over-specified. Manufacturers often offer a range of graphics subsystems allowing users to customise their workstation depending on their application area and resources. This situation is often acceptable in research environments where a workstation is often used for many different applications. It is conceivable that in the future a VR system may be installed to perform just one task - for example a VR medical visualisation system. If the application was widely accepted and installed in many hundreds of locations it might be worth producing a rendering system specially optimised for that application.

In the general workstation market the workstation itself must be highly flexible. In the situation where specialist systems are being produced a flexible *architecture* is needed which can easily be customised to particular applications without creating a completely new design.

4.10 Image quality and realism

Given many millions of pounds to spend it would be possible to construct very highly realistic images in real time using techniques such as ray tracing and radiosity. In practice, with more modest sums of money it is necessary to ask how realistic the images that are generated have to be. Only a few applications (mainly limited to some design applications and top of the range flight and driving simulators) require very high levels of realism. Many types of scientific visualisation and information retrieval applications do not require realism - they require the ability to display large amounts of information clearly and accurately.

Many ways of enhancing the quality of the generated image exist - for instance better quality shading, shadowing and texture mapping. In many architectures there exists a trade off between scene complexity (in terms of the number of primitives rendered per frame), the refresh rate, cost and enhancements. These enhancements are only worth considering if the particular application is likely to be able to perform better with them.

In some applications shadows can help in perceiving spatial relationships in an image **[WaFG92]** - the size of the shadow and its distance from the object which cast it tells the user how far the object is from the surface the shadow is cast on. Unfortunately calculating shadows is expensive and many systems do not support automatic shadow generation.

In applications where objects are seen from a large distance, texture mapping can be used to replace the large number of polygons normally required to represent the detail on an object. This technique is often used to place windows on large buildings or to act as a crude replacement for a tree (or an entire forest). Texture mapping can also be used to simulate printed surfaces such as signs.

Enhancements such as texturing and shadowing (which are described in more detail in **Chapter 5**) may be more effective than producing fully realistic images. **[SaTa90]** state that "*Comprehensibility is mainly created through suitable enhancement rather than by accurately simulating optical phenomena*". In that paper (entitled 'Comprehensible rendering of 3-D shapes') it is suggested that edges between surfaces are actually emphasised using line drawings on top of conventional images - these techniques can produce images which clearly show the shape of the object yet look realistic. Dooley and Cohen in **[DoCo90]** argue that normal hidden surface removal loses too much information - and that it would be useful if hidden surfaces were somehow partially visible. They create a system which produces images similar to technical drawings, where hidden edges are drawn dashed. Different hidden surfaces are displayed using different line patterns and thicknesses chosen to show the edges importance. The examples shown in that paper are not realistic but they show the detail of complex shapes in a clear and easy to understand manner. It is important to realise the aim of most applications is to display information clearly and *not* to produce the nicest looking image.

It is also important not to have annoying or distracting artifacts which could reduce the user's ability to use the system. Some of these have already been covered (such as temporal aliasing); other forms of rendering artifacts are covered in **Appendix A** .

4.11 Suitability of three dimensional displays

In some situations a series of two dimensional images may be easier to understand than a three dimensional model. An example is that, without translucency, a three dimensional representation of a medical scan only allows part of the data to be displayed.

Transparency can allow the display of features deeper inside the model but it may be confusing. In some cases it may be more effective to show a simple two dimensional slice through the object.