

# “Reducing Lags in Virtual Reality Systems using Motion-Sensitive Level of Detail”

Martin Reddy  
JCMB 1408, King’s Buildings  
University of Edinburgh, EH9 3JZ  
e-mail: mxr@dcs.ed.ac.uk

## Abstract

Most virtual reality systems suffer from a time lag which can detrimentally affect the user’s performance. This paper describes the major components of this lag and details the factors which contribute towards it. The use of Level of Detail (LOD) is one possible way to reduce the lag. This technique is normally implemented on a distance-based criterion (i.e. lower levels of detail are used when an object is far away); however, I am currently investigating the use of Level of Detail in relation to the velocity of each object in a virtual environment.

## 1 Introduction

This paper represents on-going research in the Computer Science department of the University of Edinburgh, in association with the department of Psychology. It is concerned with addressing the very real problem of lag which is inherent in most virtual reality (VR) systems.

This lag exhibits itself as a noticeable delay between a change being effected on the virtual environment (e.g. by a user’s action) and seeing that change reflected on the display device. It has been shown that humans can detect head tracking lags of greater than 5 msec and that delays of longer than 30 msec can produce effects of motion sickness and hinder operator performance (Frank et al., 1988). It is therefore desirable to investigate ways of reducing these lags. (It should be noted however that we will never completely abolish these lags: we will always want to do more than the computer is capable of.)

Although, to a certain extent, these lags can be reduced by resorting to custom-built hardware, it is the view of the author that if VR is to become a cheap and widespread technology, then it must be made available on low end, off-the-shelf architectures. For this to be possible, work is required into the development of algorithmic solutions to combat the various latencies currently experienced in VR systems.

In this paper, I will describe the work which I am performing towards this goal by attempting to optimise the display based upon the angular velocity and visual content of each object in the virtual environment.

## 2 Components of System Lag

The problem of System Lag can be divided into three fundamental components: **1.** Sensor Delay, **2.** Processing Delay, **3.** Rendering Delay (Atkin, 1993). These are defined as follows.

$$SystemLag = SensorDelay + ProcessingDelay + RenderingDelay. \quad (1)$$

Sensor Delay is the lag which is induced by sampling the real environment (e.g. tracking the user's head or hand) and passing this information to the computer system. I am not concerned with Sensor Delay in this paper.

Processing Delay encapsulates all of the calculations which must be performed on each object in order to update its state. Examples of this are: rotating an object; deforming an object; performing collision detection; updating representations of the user's body parts etc. The following factors can affect the magnitude of the Processing Delay:

- **Complexity of Scene** : The more dynamic objects which are in the environment, the more pronounced the Processing Delay is likely to be.
- **Complexity of Objects** : More complex objects will take longer to process because all calculations must be applied to each atomic part of the object.
- **Degree of Interactivity** : If the user can effect a wide variety of changes on the environment, then more processing is required to monitor and control all of these possibilities.
- **Complexity of Dynamics Rules** : The 'Laws of Physics' for the virtual environment, e.g. incorporating gravity into the model or flight dynamics into an aircraft simulation.

Rendering Delay is the time taken to produce an image of the virtual environment on the display device, based upon a particular viewpoint. Depending upon the algorithm employed, this could involve operations such as: clipping non-visible objects, projecting coordinates, backface removal, depth sorting, accounting for light sources, applying textures etc. Factors which can affect the magnitude of the Rendering Delay include:

- **Number of Visible Objects** : Due to efficient clipping algorithms, the number of visible objects in a scene can have a more radical effect on the Rendering Delay than the total number of objects in the environment (This also implies that the Rendering Delay can vary quite noticeable depending upon the viewpoint).
- **Complexity of Objects** : All of the operations described above must be applied to each atomic part of an object (e.g. each polygon). Therefore the more parts which are present in an object, the more times these operations must be performed, and consequently the larger the delay induced.
- **Shading Model** : Several techniques exist to implement various kinds of light sources and the subsequent shading of objects. Common methods include flat (Lambertian) shading and smooth (Gouraud or Phong) shading, in increasing order of computational cost (the Lambertian model calculates one normal for an entire polygon, whereas Gouraud interpolates the vertex normals and Phong calculates normals for each pixel).

## 3 Level of Detail

### 3.1 The Use of LOD in Computer Graphics

All objects in a virtual environment are composed of a collection of surfaces (e.g. polygons, bézier/spline patches etc.). Based upon this construction, it is usually possible to create another representation of an object which has a different Level of Detail (LOD). This can be achieved by creating a model which has less surfaces in it, but still retains the general form of the original object. By doing this, both the Processing Delay and the Rendering Delay can be reduced

(because there are less surfaces to process and display, respectively). However, the object will appear more crude and less detailed, i.e. LOD is an image quality/performance trade off.

The most common use for LOD so far has been to relate the level of detail of an object to its distance from the observer. Due to the tessellation of most computer displays, small objects can only be displayed to a certain degree of accuracy and detail. Therefore any areas of high detail in an object will not be visible when the object is very small (e.g. at a distance). So in this situation, it is pointless for the graphics system to consider these components. It therefore makes sense to substitute a lower LOD model when the object is beyond a certain distance in order to improve the efficiency of the system without unduly affecting the fidelity of the display. This technique has been used successfully in numerous flight and car simulators (Kemeny, 1993).

## 3.2 Generating Level of Detail

Attempting to produce a different LOD model by hand is a very tedious and approximate process. It is therefore desirable for the computer to generate lower LOD models automatically from an original object (i.e. reduce the polygonal complexity of an object). There are two principal techniques which can be employed to perform this task. The first is to attempt to remove certain polygons, usually by merging a number of polygons into a single polygon which approximates the previous grouping (Hamann, 1994; Holloway, 1991). This method requires two issues to be resolved: **1.** the method for selecting the polygons to be reduced (e.g. area or curvature), and **2.** the algorithm to perform the desired reduction.

An alternative approach is to use adaptive subdivision (DeHaemer Jr. and Zyda, 1991). This process begins with an initial approximation to a polygon mesh and then refines this approximation by recursively subdividing it at the point where it varies most from the original mesh. By adjusting the algorithm's tolerance, the approximation can be made as accurate as required.

# 4 Motion-Sensitive Level of Detail

## 4.1 An Overview

My research interest lies in relating LOD to an object's relative motion. The basic premise is that the human visual system cannot resolve as much detail in a moving object as in a stationary object. It should therefore be possible to degrade the image quality of an object based upon its apparent motion. This would reduce both the Processing and Rendering Delays and hence increase the frame rate during periods of motion.

In a totally static environment, the user will not notice any system lags because the scene remains constant. It is only when objects are moving, or when the user alters the viewpoint, that these latencies become apparent. It is therefore during these periods of motion, more than any other time, that we require rapid feedback to improve interactivity. By relating the level of detail of an object to its motion, it would be possible to utilise high detail models when the object is stationary so that the user is presented with a high resolution image, but to use lower detail models during periods of motion in order to increase the interactivity of the system.

This can be taken a step further in an immersive VR system because whenever the user moves their head, the whole environment is effectively moving (in relation to the user's viewpoint). Therefore almost all objects can be degraded in order to improve the frame rate during exaggerated head movements<sup>1</sup>. This is the time when most users report a feeling of disorientation:

---

<sup>1</sup>Note that not all objects will necessarily be degraded when the user moves their head because, for example, the user may be tracking a moving object which should therefore remain in high detail

when they move their heads rapidly, and the system does not keep up with their movements. But again, as we turn our heads, our vision blurs (in relation to the speed of our movement) and so we can take advantage of this by reducing the quality of the scene during these periods in order to increase the frame rate.

This idea has been suggested in the past and a couple of researchers have even developed systems based upon this concept (Funkhouser and Séquin, 1993). However, these have normally been implemented in a rather *ad hoc* fashion with little consideration given to the characteristics of the human visual system and how it functions. It has also tended to be a facility which appears to be ‘tacked onto the end’ of another research programme. However, I believe that this issue deserves a more thorough and formal investigation in its own right.

## 4.2 Perceptual Issues

The goal of this work is to investigate the minimum amount of visual information to display, based upon the characteristics of the human motion perception system. This motion system can be modelled by a measure called temporal frequency, defined as follows:

$$\textit{TemporalFrequency} = \textit{SpatialFrequency} \times \textit{AngularVelocity} \quad (2)$$

The angular velocity component is simply the speed at which an object moves across the retina. The spatial frequency term is defined as the rate of change of intensity over field of view (measured in cycles per degree). Therefore images composed of many areas of sharply contrasting intensity will contain mostly high spatial frequency components; whereas images with relatively smoother transitions will generally contain low spatial frequencies (Humphreys and Bruce, 1991). This implies that a scene which has a large number of sharp, well-defined edges will be characterised by higher spatial frequencies than a scene with smooth, rounded features.

For any specific velocity, there is only a certain range of spatial frequencies which are visible – anything out with this range is simply invisible to the human eye. The general relationship is that at low velocities, only high spatial frequencies are visible and conversely, at high velocities, only low spatial frequencies are visible. The implication of this phenomenon for computer graphics is that we can remove high spatial frequency components of an object when it is in motion and the user’s perception of the scene will not be affected (Nakayama, 1990).

However, there are two main problems associated with calculating the spatial frequencies of a computer-generated image. These are that it is both viewpoint dependent and field of view (FOV) dependent. The latter restriction is not a complication in an immersive VR system as the FOV of the head-mounted display will be known. However, accounting for viewpoint dependency is more problematic, particularly in real-time. The following section details my approach.

## 5 General System Design

Work is being undertaken at the University of Edinburgh to implement a virtual reality system which utilises motion-sensitive LOD in an attempt to improve performance. The current equipment being used includes a Pentium based platform, augmented with the *Reality 3* graphics system. Immersive simulations are realised through a VPL *Eyephones LX* head-mounted display and a Polhemus *Fastrak* tracking system.

The software system being developed consists of two major components. These are as follows:

- **The Pre-processor** : This encapsulates all of the operations which must be performed

*à priori*. Spatial frequency is a viewpoint dependent measure; however it would be too computationally expensive to calculate in real-time (the standard technique would require a Fourier analysis on each frame). It has therefore been decided to gather general spatial characteristics beforehand for each object, and then employ a heuristic algorithm to estimate the spatial frequency components of an object in real-time.

The pre-processor stage will also govern the task of automatically generating various LOD models from an original object. This will be done via an appropriate polygon reduction scheme.

- **The Scheduler** : This is the part of the VR software which implements the motion-sensitive system. It will monitor the velocities of all objects (relative to the user's viewpoint) and estimate the spatial frequency content of these objects in real-time. It will then modulate the LOD of each object based upon these measures.

## 6 Conclusion

The above system is still at a very embryonic stage and so no figures on the effectiveness of this approach can be presented at the current time. However, it is hoped that by basing the system on models of visual perception, a significant improvement in performance will be experienced during periods of motion, with relatively little loss of perceivable image quality.

## References

- Atkin, P. J. (1993). Parallel Processing for Virtual Reality. In *Parallel Processing for Graphics and Scientific Visualization*. University of Edinburgh.
- Frank, L. H., Casali, J. G., and Wierwille, W. W. (1988). Effects of Visual Display and Motion System Delays on Operator Performance and Uneasiness in a Driving Simulator. *Human Factors*, 30(2):201-217.
- Funkhouser, T. A. and Séquin, C. H. (1993). Adaptive Display Algorithm for Interactive Frame Rates During Visualization of Complex Virtual Environments. *Computer Graphics Proceedings, Annual Conference Series*, pages 247-254.
- Hamann, B. (1994). A Data Reduction Scheme for Triangulated Surfaces. *Computer Aided Geometric Design*, 11(2):197-214.
- Holloway, R. L. (1991). Viper: A Quasi-Real-Time Virtual-Worlds Application. UNC Technical Report No. TR-92-004, Dept. of Computer Science, University of North Carolina, Chapel Hill, NC.
- Humphreys, G. W. and Bruce, V. (1991). *Visual Cognition: computational, experimental & neuropsychological perspectives*. Hove Lawrence Erlbaum Associates. ISBN 0-863-77125-4.
- Kemeny, A. (1993). A Cooperative Driving Simulator. In *Proceedings of the International Training Equipment Conference and Exhibition*, pages 67-71, London, UK.
- DeHaemer Jr., M. J. and Zyda, M. J. (1991). Simplification of Objects Rendered by Polygonal Approximations. *Computer & Graphics*, 15(2):175-184. Pergamon Press, UK.
- Nakayama, K. (1990). Properties of early motion processing: Implications for the sensing of egomotion. In R. Warren and A.H. Wertheim, editors, *The Perception and Control of Self Motion*, pages 69-80. Lawrence Erlbaum, Hillsdale, NJ.