Hardware Enhancements for Raster Graphics

Technical Report 85-015
June, 1985

Andrew S. Glassner and Henry Fuchs

The University of North Carolina at Chapel Hill
Department of Computer Science
New West Hall 035 A
Chapel Hill, N.C. 27514

To appear in Proceedings of the 1985 NATO Advanced Study Institute Conference on Fundamental Algorithms in Computer Graphics, Yorkshire, England, May 1985

Hardware Enhancements for Raster Graphics

Andrew Glassner
Henry Fuchs

Department of Computer Science
University of North Carolina at Chapel Hill
Chapel Hill, North Carolina 27514 USA

Abstract

Great strides have been made in the field of fast, realistic image generation in the last 20 years. A generally acknowledged goal of the field called the Ultimate Display has been guiding the field all that time, and may very well be guiding us for the next 20 years. But with the recent advent of VLSI technology the Ultimate Display may finally be less of a dream and more of an achievable goal.

Most of the recent advances toward the Ultimate Display have been in the fields of image synthesis and generation. The development of these fields has taken place on at least two major fronts in recent years. One research direction has been to place a premium on the realism of the generated images, regardless of the clock time required to create the image. Another major research direction has been to build highly interactive, dynamic systems that must produce successive images within some fixed period of time, usually 1/30 second or less.

This paper examines approaches that have been used to squeeze the maximum realism out of an image generated on a real-time system. Most of these approaches have begun as insights into the nature of the algorithms, permitting a speedup through strategic and tactical improvements. Recently, the powers of multiple processors have been brought to bear on the problem, exploiting parallelism inherent in the image synthesis procedure. Very Large Scale Integration (VLSI) techniques have proven to be a very useful way to develop these multiple-processor systems. VLSI is helping not only with the generation of images but with many other problems involved in the Ultimate Display, such as hand and eye tracking, and a miniaturized head-mounted display. This paper discusses each of these subjects.

This work was supported in part by the Defense Advance Research Project Agency, Contract number DAAG29-83-K-0148 (monitored by U.S. Army Research Office, Research Triangle Park, NC), the National Science Foundation, Grant number ECS-8300970, and the National Institutes of Health (NIH), Grant R01-CA39060.

I. Introduction

The field of computer graphics has grown rapidly in the last several years. One subfield of special interest to many people is real-time graphics. The term real-time is usually applied to systems that must react interactively by updating the image on the screen to reflect changes desired by the user.

In this paper we will restrict our attention to real-time, three-dimensional shaded graphics systems. These systems have been used for many years as flight simulators and trainers. More recently, real-time systems have started helping doctors, architects, engineers, and others who work with complex three-dimensional scenes. As the range of applications grows, so does the range of operations requested by these users. Thus the designer of a real-time graphics system must support an ever-increasing range of operations, always working within the constraint of producing a new image at least 30 times each second. Some of today's applications are so complex that current systems are unable to meet this demand, and users must tolerate slower update rates. But the goal of 30 frames per second remains.

In addition to the increasing range of operations and complexity of data, users are also demanding increased realism in their images. Many techniques have been developed for generating images that appear so realistic they seem to be photographs of natural, complex scenes. But these techniques have generally been developed with little consideration for their real-time applications. The more realistic an image appears the more work must go into its synthesis, and in general this work costs time. Thus the graphics architect must supply even more function in the ever-constant 1/30 of a second. The job of the real-time graphics system architect does not end with the synthesis of the image. The other aspects of the man-machine interface also must be considered and gracefully incorporated into the whole system.

In this paper we will consider how some real-time computer graphics systems have been designed to respond to these demands of increased performance and realism.

In Section II we present the overall goals of real-time computer graphics systems in their most general form. We then identify four key topics important to reaching that goal: image generation, display techniques, tracking techniques, and database interaction. The next four sections examine these key topics in more detail, with special emphasis on past attempts to speed up or improve the results.

Section III examines the key topic of image generation, and begins with a brief history of the development of some key ideas in that field. We then present a summary of the image generation process as we perceive it, and analyze the different approaches that might be taken to speed it up. The rest of the section presents several such approaches, characterized by the terms identified in the analysis.

Section IV involves the key topic of display techniques, and how the image may be best presented to the user. This issue is critically related to the perception of the image and its effectiveness as a conveyor of information to the user.

Section V considers the issue of key topic of tracking. The goal is to make the transfer from user to computer of information describing the user's position and gaze direction as natural and effective as possible.

Section VI concerns the key topic of control and modification of the database. As the user interacts and modifies the data it reacts and changes. As the number of applications of real-time graphics increases this dynamic element is becoming increasingly important.

Section VII offers our acknowledgements.

II. Overall Goals of Real-time Graphics Systems

In 1965 Ivan Sutherland presented a seminal paper describing his vision for the future of computer graphics [Suth 65]. His goal was the creation of an "ultimate display" to present the graphical output of a computer to a user. This display would be so portable that it would travel with the user all the time he or she was using the machine. To present its graphical information it displayed images in front of the user's eyes; thus it seemed natural to mount the device on a helmet on the user's head. The family of techniques that have risen to approximate this goal are thus referred to as "head-mounted displays." The head-mounted display may indeed take the form of a helmet, or miniaturization of video display tubes and optics may give it the form of a lightweight pair of glasses. The display itself need not obscure the user's environment. Indeed, half-silvered mirrors have been used for many years to combine two images; they could certainly be used to combine a synthetic image with a user's environment.

One goal of the head-mounted display vision was that the images created by the computer should be so realistic and natural that they are indistinguishable from actual objects. Thus, the computer is able to present a new environment to the user, one that is a smooth mixture of the simulated and the real.

Another goal in the head-mounted display is the ability to track and respond to the user's actions. For example, let us imagine a surgeon wearing a head-mounted display. The surgeon might be presented three-dimensional data from a CAT or NMR scan of a patient, displayed to give the surgeon the illusion that she is a small visitor inside the patient's body. For example, the surgeon may suspect a tumor inside some body part, so she walks over to the part in question. Unfortunately, she may find that the body part completely surrounds the suspected tumor. The surgeon could reach her hand out, sweep the extra material to the side, and look into the hole she just made. Of course, the surgeon actually didn't push any real thing; the computer saw, recognized, and reacted to the movement by manipulating the database in the correct way to preserve the illusion and help the surgeon do her work.

So in its full form the head mounted display allows the ultimate computer-generated video experience. The wearer can be placed into any visual surroundings with any degree of correlation to the "real world," including none at all! The actions of the user are interpreted as interactions with the real and simulated worlds, and the computer adjusts the simulated world accordingly. Actually, the original "ultimate display" idea went even further, providing inputs to all five senses, not just sight. In particular, the sense of feel would also receive synthetic input from force feedback equipment. Thus, a user could sit in a virtual chair, or climb a virtual tree! Except in passing we won't consider the other senses again in this paper.

To bring about the head-mounted display we have many problems to solve. In this paper we have isolated four critical issues. These consist of algorithms for synthesizing images, techniques for displaying images, methods for tracking the user's gaze and actions, and ways to control the database to reflect the user's actions.

III. Real-Time Realistic Image Generation

The generation of realistic images has been one of the central research issues of computer graphics for many years. The general approach has been to isolate the critical aspects of natural scenes that give them the quality of "reality," and study how synthesis algorithms may include that information in the images they generate. The first pictures made that exhibit this new information are usually made at great expense in computer resources, including rendering time. Only after the effects are well understood are we able to search for faster implementations.

The critical constraint in real-time image generation is the persistence of vision of the human eye. If the eye is presented with successive images too slowly, the human peceptual system will not fuse them into one continuous stream. The required interval between frames varies among observers and conditions; for instance the interval in a dark room is not the same interval in a light room. For a variety of technical and psychophysical reasons the value chosen as the "real-time threshold" is usually 1/30 of a second. Thus the goal of the real-time systems architect is to completely finish the rendering of a picture and get it presented to the viewer in no more than 1/30 of a second.

In general, to increase the quality of the image requires an increase in the amount of computer power used to create the image. In a single processor system this means a higher-quality picture (more realistic, more detailed, or superior in some respect to another picture) requires more time. Multiple-processor systems offer the promise of better pictures without an increase in time by applying their computing power in parallel.

In the next part of this section we will briefly review some highlights in the history of computer image generation. This history is extremely biased; no attempt was made to give a comprehensive review of the field. Rather, we simply mention those contributions which will be relevant later in this paper.

Early History

When Sutherland proposed the "Ultimate Display" in 1965 no one had yet rendered a shaded image with hidden surfaces removed (or at least if he did, he didn't publish it widely). The core of computer graphics was vector screens and light pens; realism was a dream.

Two years earlier in 1963 Roberts described a method for removing the hidden lines from a vector image [Robe 63]. It wasn't until four years after Sutherland's paper that the first two real-time hidden surface techniques emerged, in rapid succession. In 1969 Schumaker et al. described the General Electric real-time shaded graphics system [Schu 69]. In 1970 Watkins described his real-time scan-line algorithm for visible surface generation [Watk 70]. His algorithm was later incorporated into the real-time flight simulator produced by the Evans & Sutherland Computer Corporation. The GE and the E&S systems were the first two real-time shaded graphics displays. They were able to produce an apparently continuous stream of 3D shaded images from a changing viewpoint.

These systems gave high performance, but they had price tags to match. It was clear that small systems could not afford the extensive hardware routinely employed by these large-scale simulators. Efforts continued at finding ways to find more efficient methods for rendered image generation. In 1972 Newell et al. published a paper which suggested a very elegant technique called the "painter's algorithm" [Newe 72]. Polygons were simply drawn on the screen one after the other, starting with the polygon farthest away and working forward toward the user. Unfortunately, there are a myriad of special cases that the unadorned painter's algorithm doesn't handle. The authors suggested an hierarchy of techniques to be applied in order, starting with the bare algorithm and increasing in complexity and required computation. Thus, the simplest cases went fastest, and extra time was only spent when needed.

In 1974 Sutherland et al. published a classic paper comparing ten published hidden surface algorithms [Suth 74]. In this paper they describe the common and differentiating features of the various algorithms, and offer analyses of their performances. They also mention in passing that a large array of memory can be used to store the depth at each sample point in an image. This is the germ of the idea of the z-buffer, which appeared again in the description of a frame buffer [Kaji 75].

Later History

In this subsection we describe some important advances that either have not yet been implemented in specialized hardware, or are currently being tested in different implementations. There have been several persistent problems in computer-generated images that confront the designer of a graphics system. Some of these problems are algorithmic and due to insufficient knowledge or study (such as a simulation of mixing liquids), while others are inherent in the medium (such as sampling artifacts).

In 1977 Weiler and Atherton presented what is essentially an inverse painter's algorithm [Weil 77], commonly known as the "cookie cutter" algorithm. The algorithm simply draws polygons sorted from nearest to farthest, testing each new polygon against those already in the pixel. Those parts of the polygon that are already covered in the pixel are simply removed prior to adding that polygon in. This algorithm has a great advantage in that the area contributions of all fragments within a pixel sum to unity. We can use this information to assign a pixel's color in such a way that it correctly combines the contribution of each object in the pixel, a process known as "anti-aliasing."

In 1980 Whitted presented an extension of the ray-tracing algorithm originally presented by Appel [Appe 67]. Whitteds's algorithms incorporated shadows, reflections, and transparency [Whit 80]. The techniques based on ray tracing yield pictures of extremely high realism, but at a correspondingly high computational cost. The subject of ray tracing has received a great deal of attention recently, including a description of distributed ray tracing by Cook et al. [Cook 84]. This formulation combines solutions to many traditionally hard problems in graphics into one unified and elegant model.

A paper presented in 1984 by Carpenter described an extension of the z-buffer called the A-buffer [Carp 84]. The A-buffer is similar to several other techniques in that it collects subpixel fragments which are combined to determine a final shade for the pixel. The A-buffer has the advantage that fragments can arrive in any order; final color resolution is performed when all visible surfaces have been scan converted, shaded, and entered into the A-buffer.

Lighting and Shading

Calculating the proper shade at each pixel once the visible surface is known was once thought to be simple. However, techniques which produce increasingly realistic images require increasingly more computer time to perform this operation.

A variety of techniques exist for computing shading. These also follow the pattern that more realism costs more computing time. Three shading models are most commonly used in real-time systems today. The models can be applied to any kind of surface element, but for clarity of comparison they will all be discussed with reference to polygons. We will also note that these models are used to determine the illumination of a surface, not necessarily the color of the surface at any point.

The Lambert (or faceted) model simply assigns a single color to an entire polygon based on the angle the polygon's surface normal makes with the light source. The Lambert model is the simplest lighting model used. Next in complexity and expense is the Gouraud (or smooth-shading) model, which computes the correct shading for the polygon at each vertex and blends the shades across the face of the polygon. Most expensive in time and complexity is the Phong (or normal-interpolating) model. This model computes the shade for each point on the polygon as a function of the location of the point and the average normal for the composite surface at each vertex of the polygon. The Phong model requires a square root for each pixel in the final display; this is a very costly operation.

Summary of Real-time Image Generation

We have given our view of the generic real-time image generation process in the model in Figure 1. The process of generating a frame begins anew from the top for each object, creating the images of objects one by one to build up the entire picture. Objects are stored in the database, and are retreived at the appropriate time by the database accessor. The object is then subjected to a series of transformations, possibly including rotation, scaling, skewing, and other geometric operations. The object is then projected to the screen by a process called scan conversion. This basically consists of determining which pixels might need to be altered to add this object to the final image. After this step each pixel is considered independently.

Each pixel which might be affected by this object must determine whether the object is truly visible, or if it is obscured by other objects at that pixel. This determination of visible surfaces is followed by a shading step. Each pixel which does include some of the object must determine the correct coloring for the section of the object within the pixel. This step is referred to as .shading, although it may also include operations such as texturing.

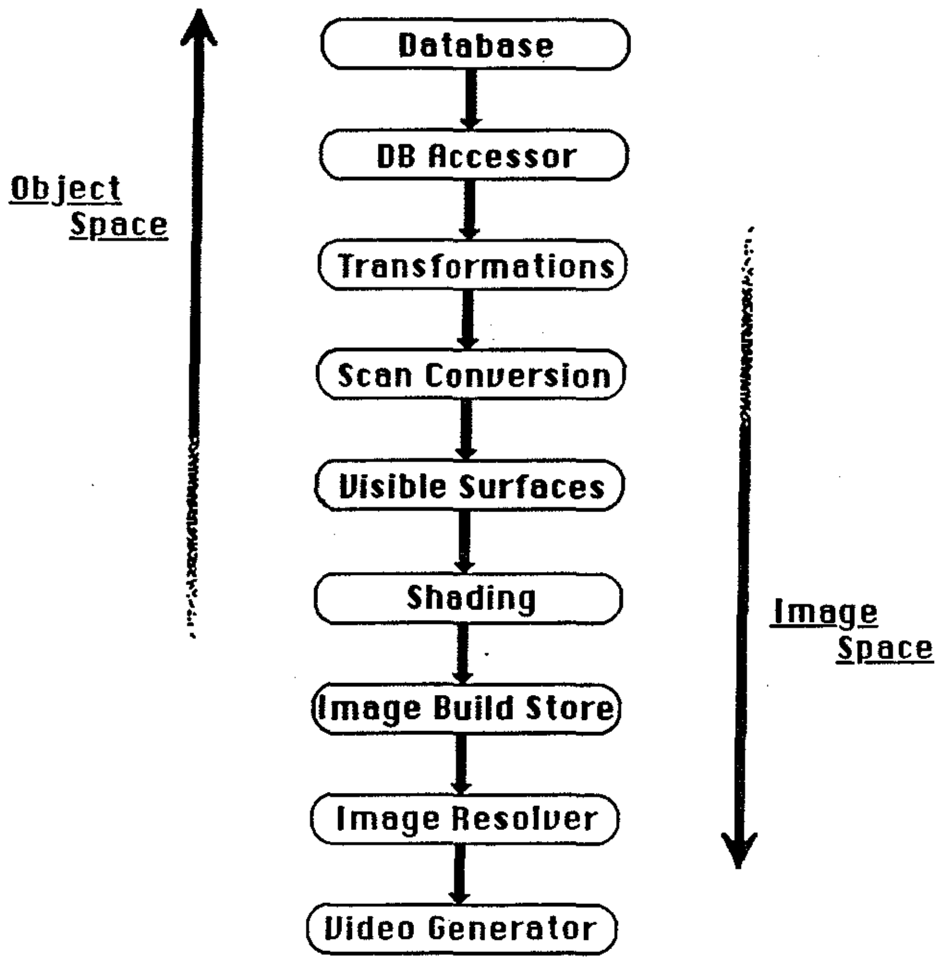
When a piece of object has been colored it is sent to the image build store, where this fragment of the object is combined with all other current and previous object fragements. The image resolver then combines these fragments in some way to produce a composite picture. The picture is actually displayed by the video generator, which may employ algorithms for display correction of the image, including color adjustment and compensation for a particular observer's color biases.

This process can be viewed as a classic pipelined algorithm, where each box is simultaneously working on the results of the previous box. If each step in the pipeline takes the same amount of time the system can be clocked synchronously and thus achieve maximum efficiency.

We suggest that there are two main techniques for speeding up this pipeline. The first technique is based on classic parallelism; the application of many identical processors to solve a given task. To achieve this we isolate a step in the pipeline and design a processor to execute it. We then amass a set of these processors under control of a supervisory processor and use this organization as a pipeline step.
In essence, we simply assign many identical processors to a single task. The other way to speed up the pipeline is to consolidate pipeline steps into a single chip. We can then take advantage of the extreme ratio of on-chip to off-chip signalling times and increase the throughput between those steps of the pipeline. This approach is especially fruitful when used with high-bandwidth pipeline elements.

Relative to the pipeline boxes of Figure 1 we can consider the first approach as an extension of a given box in the horizontal direction, and the latter approach as a combination of boxes vertically. Thus the column of processors in Figure 1 may become either wider or shorter.

In the classic paper comparing hidden surface algorithms, Sutherland et al. [Suth 74] distinguish between those algorithms that work in image space (i.e. the screen) and those that work in object space (i.e. the 3D world where we define objects). In a similar vein, speedup techniques can be viewed as operating in either image space, object space, or both. An examination of Figure 2 will also show an interesting correlation: object-space speedups are implemented with pipelining, while image-space speedups are implemented with parallel processing.



The Image Generation Pipeline

Figure 1

The Image Generation Pipeline

Technique	References	Pipelining	Splitting in Object Space	Splitting in Image Space	Classical Parallelism
8-by-8 display	Fuchs '77 Fuchs and Johnson '79 Clark and Hannah '80 Gupta et al. '81 Demetrescu '85			✓	✓
Distributed Z-Buffer	Parke '80			✓	✓
Pixel-planes	Fuchs '81 Poulton '85			✓	✓
Sub-volume Ray-Tracing	Dippé & Swenson '84	✓	✓		
Processor per Polygon	Cohen & Demetrescu '80 Weinberg '81	✓	✓	✓	✓
Constructive Solid Geometry Machine	Kedem & Ellis '84	✓	✓	✓	✓
E&S Transformation Hardware		✓			
Geometry Engine	Clark '82	✓			
Weitek Rendering Pipeline	Weitek '84	✓			
TI 4161 Row-Scan Memory Chip	TI '83				
NEC 7220 Raster Fill and Scan Chip	NEC '83				

A Characterization of Hardware Speedup Techniques

Figure 2

A Characterization of Hardware Speedup Techniques

Hardware Speedup Summary

In this subsection we will look at a variety of speedup techniques that have been reported in the literature. Figure 2 summarizes the characteristics of each technique in terms of object vs. image space, and parallelism vs. pipelining.

There are two general approaches to hardware speedup: custom hardware designed to replace a general processor, and special-purpose chips designed to replace common aggregates of smaller chips. Most of this discussion will be concerned with the former, though we will consider special-purpose chips at the end.

Image-Space Approaches

A powerful, general-purpose processor at every pixel would be ideal. Unfortunately, it would also be expensive to make and maintain, hot to run, and very difficult to justify. But the processor-per-pixel idea is so seductive that various approximations to it have been tried.

One generic approach was described in various forms in [Fuch 77], [Fuch 79], [Clar 80], and [Gupt 81]. This approach is usually referred to as the 8-by-8 display. In general, the 8-by-8 display consists of a square matrix of processors, 8 processors on a side, for 64 processors in all. An example of one of the 8-by-8 architectures is given in Figure 3. The other common feature of all but the most recent 8-by-8 proposals is a picture memory buffer interlaced in both x and y by a factor of 8. Thus, each of the 64 processors can speak to a unique pixel simultaneously. The 8-by-8 displays can therefore offer a maximum theoretical speedup of a factor of 64 above a single-processor framebuffer. The most recent 8-by-8 display proposal [Deme 85] differs from the others in that each processor addresses not a single pixel, but an entire scan line. Thus, its relative power is much greater than 64 over a single processor; it is in fact the product of 64 and the width of a scan line.

Rather than taking our 8-by-8 array and assigning each processor to one pixel, we might want to take an array of some other size and assign each processor some small piece of the screen. We can then send descriptions of our surface elements to all the processors at once and let them take care of the job of checking whether or not they should be placed in the display.

Another good solution to determining which objects are visible is called the z-buffer. The z-buffer algorithm is a very common, very effective brute force solution to the hidden surface problem. With just one processor sitting on top of the z-buffer we must wait for that processor for every pixel of our object. In 1980, Parke published a comparison of several schemes for distributing control of a z-buffer among several processors [Park 80]. It was found that if many processors are each given a small piece of the z-buffer to monitor and control some significant time savings can be achieved.

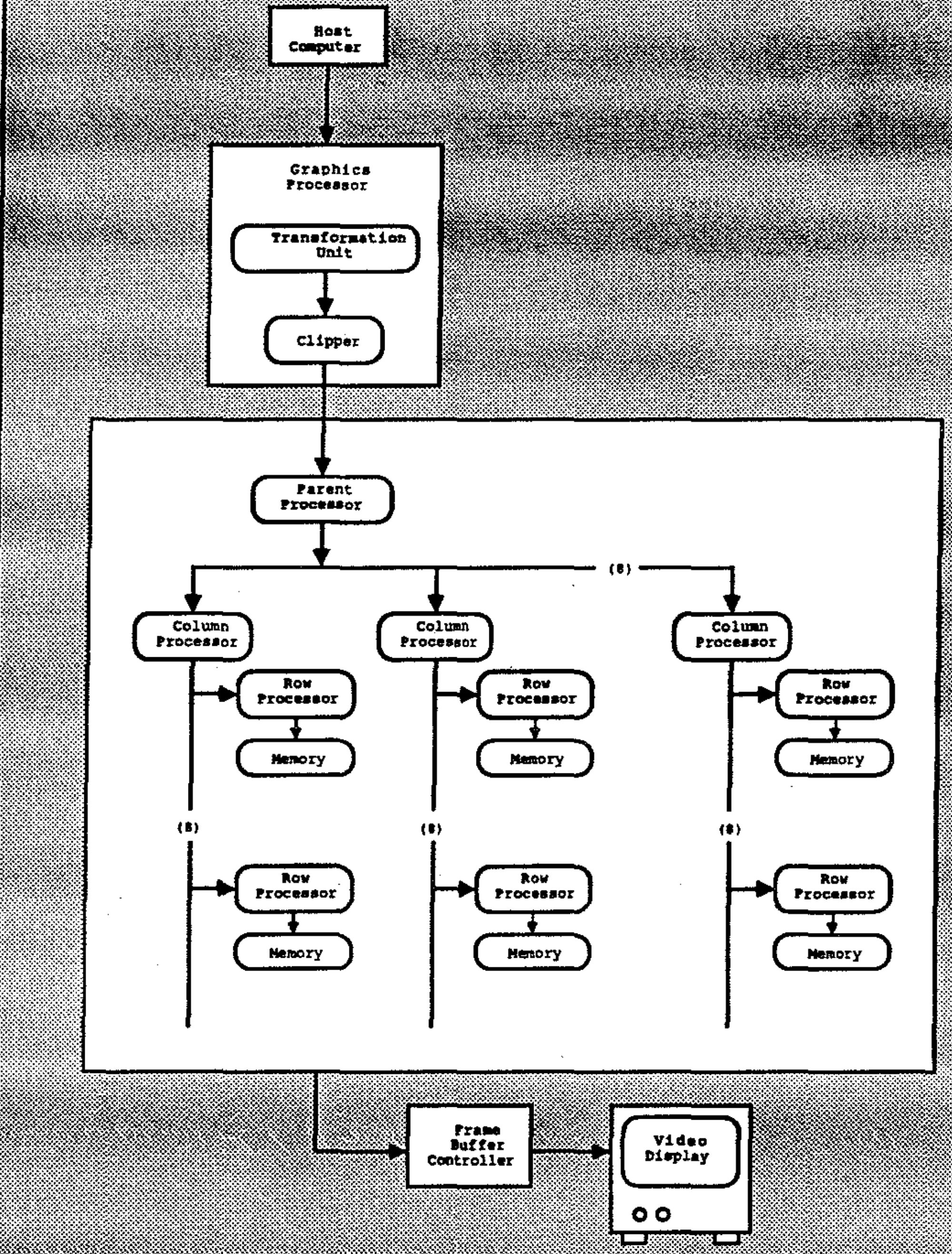


Figure 3

Clark and Hannah 8 by 8 Display Architecture

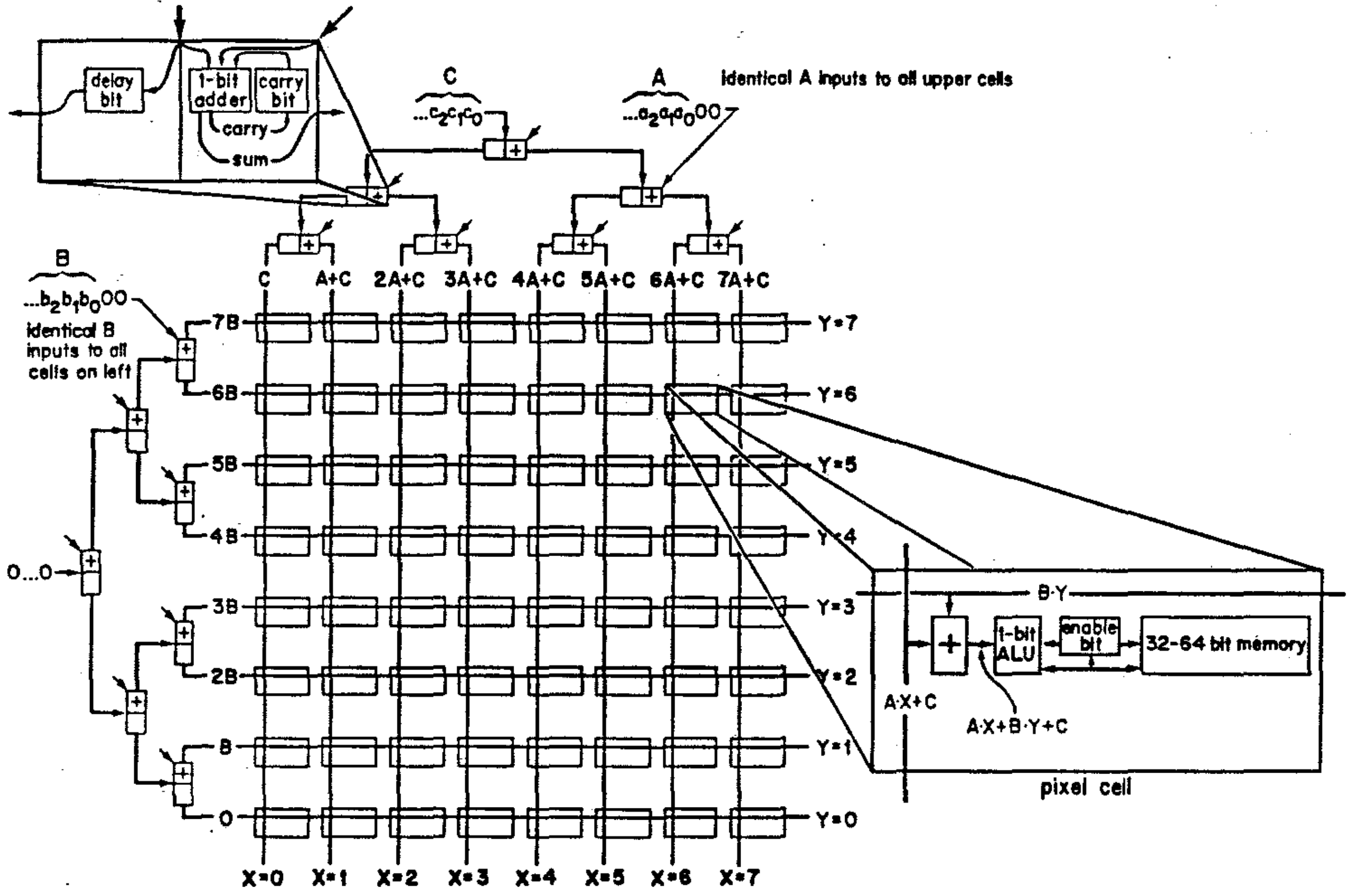
The ultimate image space machine, as we mentioned above, is a powerful processor at each pixel. The 8-by-8 display compromises on the number of processors by providing a powerful processor for each pixel in a subregion of the screen. The other compromise is to provide a processor at every pixel but vastly reduce the power of that processor. This latter approach is used in the Pixel-planes system [Fuch 81], [Poul 85], [Fuch 85], which is shown in Figure 4.

In Pixel-planes, a unique architecture is exploited to perform scan conversion of a polygon in parallel over a large segment of the screen. Each Pixel-plane chip addresses a subset of the screen and provides a very small, very limited processor at each pixel. Each of these processors is supported with a large, dedicated memory (72 bits in the fourth-generation chip). A tree structure is imposed on this chip providing the solution to the plane equation of a polygon at all points on the screen simultaneously. Each edge of the polygon is sent to the chip, and each processor determines which side of the edge the pixel it controls is on. Processors that are on the side of the edge that is "outside" the polygon effectively turn themselves off until the first edge of the next polygon. Processors whose pixels are inside the polygon then compare the pixel's current z value with the z value of the polygon at that pixel. If the polygon is farther away than the z value stored at the pixel the processor turns itself off until the first edge of the next object. Only processors still on after all edges have been checked actually end up contributing to the final image for that polygon. These processors then update their z value to represent the new object. At the conclusion of all the edges of a polygon the processors are reset and the process begins anew.

Object-Space Approaches

When rendering an image on a fixed time budget you must decide how much time you're willing to pay for various attributes of the final image. If realism is highly desired then you'll have to get the time for the processing from somewhere. If you want Phong shading, for example, you'll have to take a square root at every pixel. You can build the fastest square root extractor ever, but you still have to wait for it and no amount of parallel processing will speed that up. You can buy some time in late pipeline stages (like shading) by speeding up early pipeline stages (like visible surfaces). This is exactly the benefit of object-space speedup.

A good example of pure object-space speedup is the parallel ray tracer described by Dippe & Swensen [Dipp 84]. The three-dimensional space is subdivided into a large number of tetrahedra, each of which holds some number of objects. When a ray is sent into the scene, it passes from volume to volume until the ray-object intersection completes. Each processor is assigned a tetrahedral volume and a ray and is asked to perform the ray-tracing algorithm in that volume. When the rays are intersected and returned the visible surface problem has been solved. In this case the Transformation and Visible Surface steps have been combined in the ray-tracing processor (the ray-tracing algorithm side-steps the scan conversion step).



Conceptual design of an 8 x 8 pixel PIXEL-PLANES image-buffer memory chip.
Scan conversion, hidden-surface elimination and color-rendering commands are translated outside the memory system into A,B,C coefficients and associated PIXEL-PLANES commands.

Figure 4

Pixel Planes Organization

Combined Approaches

Several researchers have combined image and object space approaches in the same system to realize the benefits of both worlds.

In 1980 Cohen & Demetrescu presented a system which can be characterized as having a processor for each polygon along a given scan line [Cohe 80]. Each processor scan converts its polygon for a given scan line, as shown in Figure 5. The processor then examines the scan line that has been built up so far and adjusts it where necessary to include the object for which that processor has responsibility. The adjusted scan line is then output from the processor. A chain of these chips arranged in a pipeline outputs the nearest z and associated shade for each pixel in scanline order.

In 1981 Weinberg suggested adding coverage information to each message sent by the scan converting processors [Wein 81]. This information helped the combining processor compute an average coloring for the pixel and thus facilitated anti-aliasing of the image.

In 1984 Kedem presented a description of a Constructive Solid Geometry machine [Kede 84]. The machine stored objects in the leaf nodes of a processor tree, as shown in Figure 6. The objects were combined according to the rules of CSG at node processors to find the final result at a pixel. The distribution of objects into leaf nodes is an object-space division; the CSG nodes manipulate rays which corresponded to pixels, placing those processors in image space.

Support Hardware

In 1968 the Evans&Sutherland Computer Corporation completed the LDS-I, a 3D vector pipeline. The LDS-I had a hardware 4-by-4 matrix multiply unit which performed geometric transformations.

The Geometry Engine described by Clark in 1982 [Clar 82] is a general-purpose mathematics chip for graphics applications, particularly transformations. The IRIS system marketed by Silicon Graphics uses 12 of these chips to compute the geometrical and perspective transformations on 3D image data.

The Weitek WTE6000 Tiling Engine (1983) is a tiling and shading pipeline for high-speed image generation. It's basically a very specialized architecture using very fast mathematics chips.

The Texas Instruments 4161 memory chip (1983) is a dual-ported RAM designed for frame-buffer applications. It supplies two ports for chip addressing, allowing the CPU and refresh circuitry to operate independently. It also provides a 256-bit serial shift register which can be clocked off-chip to easily support transmission of chip data

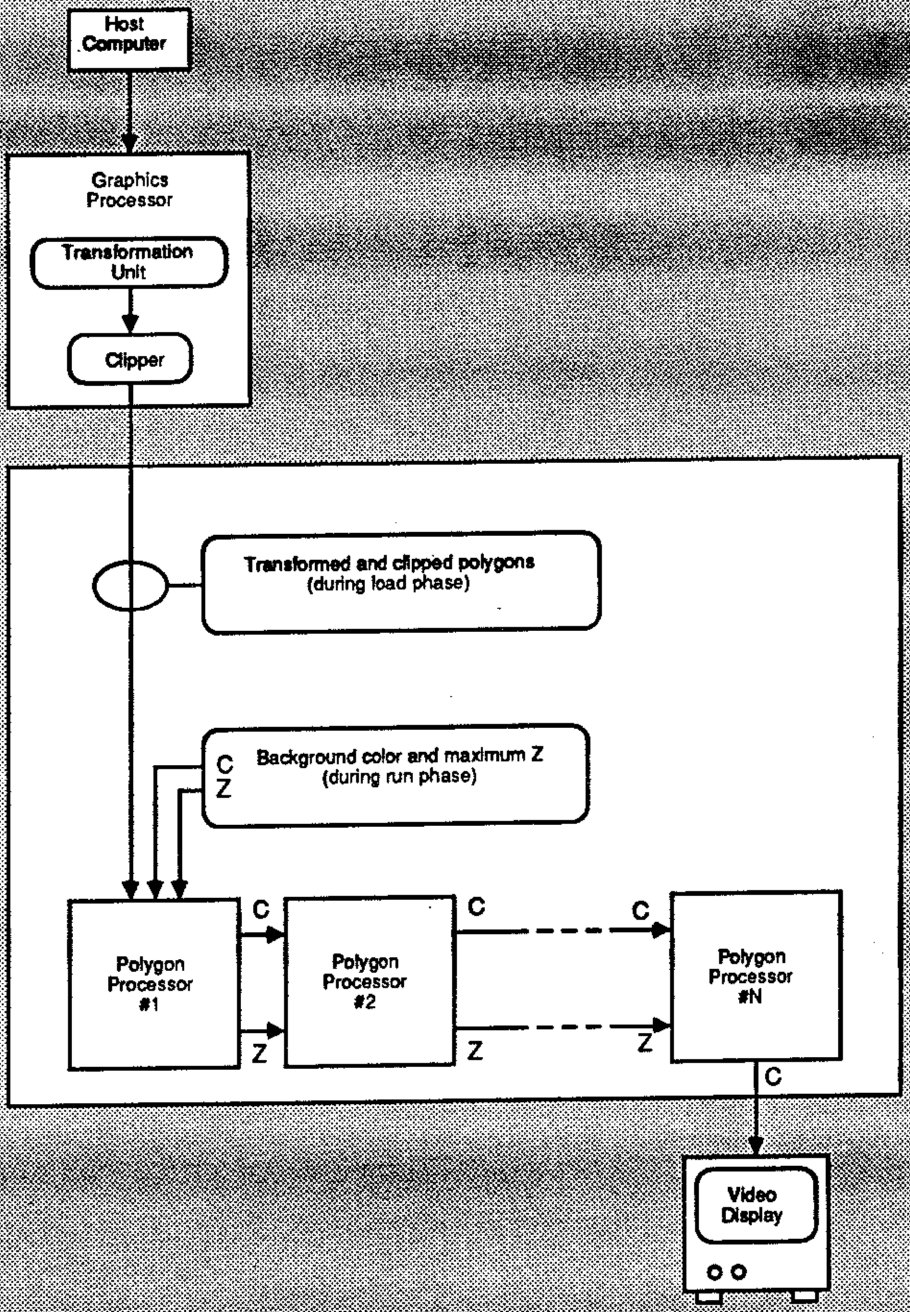


Figure 5

Cohen and Demetrescu Pipeline Architecture

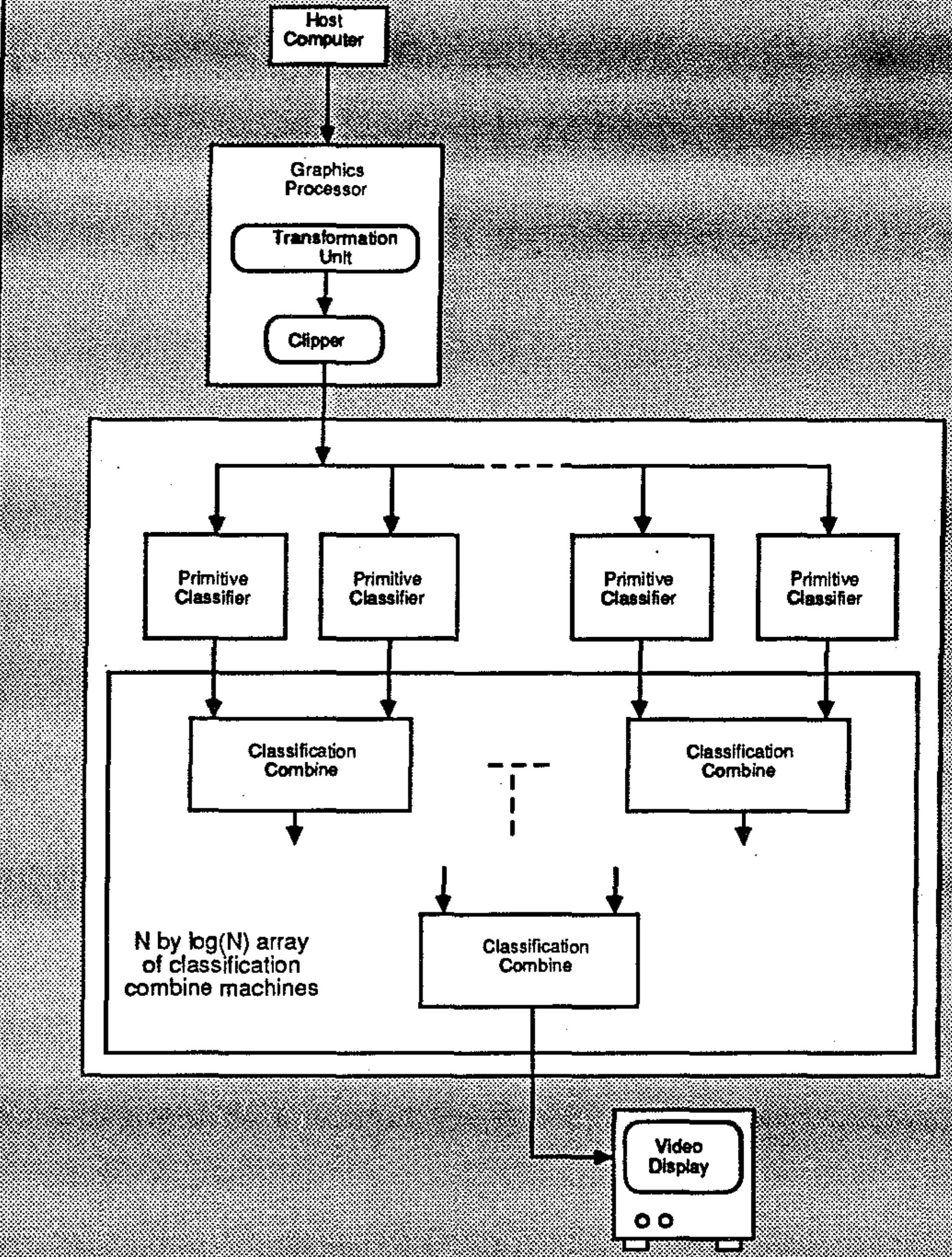


Figure 6

Kedem and Ellis CSG Machine Architecture

to raster generating hardware.

The NEC 7220 (1983) also provides several high-level functions. Intended as a general-purpose frame-buffer controller, it is able to scan-convert polygons into a frame buffer, and provides functions for drawing lines, arcs, and other graphic primitives. The chip also serves as a video generator, which supports zooming, panning, windowing, and light-pen tracking.

IV. Display Techniques

So far we have concerned ourselves with the formation of an image inside the computer. When the image rendering pipeline is completed, we have a two-dimensional matrix of colors expressed in some color system. Our next job is to make this information accessible to the user. The classic computer output devices are the punched card, line printer, and cathode ray tube. These devices are all peripheral to the computer itself, and are usually used in a deliberate way. That is, a terminal is usually thought of as a means for controlling a computer, not the computer itself. We believe that the most natural and efficient forms of communication do not involve this extra translation step, however widespread it may be. We would like our input and output to be perceived as directly associated with the machine. In this chapter we consider output devices for presenting images to users.

Dream Displays

If we were to dream of any devices for output, our dreams would be of the ultimate head-mounted display. Here the word "ultimate" is used advisedly. Probably the ultimate display in purely technical terms would be direct cortical stimulation, in which computer-generated stimuli could not be distinguished from sense-reported stimuli. We feel that there are important social and ethical issues to be addressed in such a technique, and prefer to dream about devices that can be easily removed by the wearer at any time. In that context, we envision the head-mounted display to look like a set of lightweight, attractive eyeglasses. The display would contain no high voltages, frequencies, or radiators, and would not be physically coupled to any object in the room except the wearer.

In one configuration the lenses of the glasses would be several layers deep. The lenses would be high-resolution color matrices dot-addressable by layer, where "transparent" is an available color. The lenses would have fast switching times, on the order of 33mS (to achieve frame rates of 1/30 second). The lenses would also have the ability to generate light as well as absorb and reflect it, so they could work in dark rooms. The images would consititute a stereo pair, enhancing the stereopsis by placing synthesized objects at different virtual foci through some system of optics. We would like to point out that the head-mounted display is not simply a matter of placing synthetic images in front of the user. There is a problem of hidden surfaces that has been in computer graphics since the start, and it returns in this context as well.

Imagine that you are holding a coil of rope in your hand. Your hand is stretched out in front of you, the coil is grasped in your fist and is resting on the inside of your elbow. Clearly you cannot see your elbow through the opaque rope; the rope obscures your elbow. Now think about where your hand is grasping the rope. Your fingers are coiled around the rope, so you can see your fingers but not the rope. Now let us return to the head-mounted display. You are looking at the world through half-silvered lenses that are displaying computer-synthesized images along with the outside world. The rope will be a computer-generated object. Your hands are real; they are the outside world. Now consider what you actually see. Since the images on your half-silvered lenses are merged everywhere with the real world scene, you have to somehow compensate for the hidden surfaces we mentioned just above.

How do we obscure the rope by the hand? We need to know where the hand is in relation to the rope and in relation to the viewer. Anywhere we decide the rope is behind the hand we just do not draw in the rope. Because the image of the rope suddenly stops at the edge of your fingers and starts again at the other edge, and the stereo cues make you think it is just about where your fingers are, you will probably decide that your hand is obscuring the rope. Now consider the rope sitting on your elbow; you see the rope and not your elbow. But how do we stop you from seeing your elbow? As we've described things so far there's no way; the synthetic image is merged with the real image at every point. The only way to make the elbow invisible is to opaque those points on your glasses through which you see your elbow. Now you can't see your elbow, but you still see the image of the rope; thus you may decide that the rope is obscuring your elbow.

The conclusion we can draw from this is that the glasses need to be more than just transparent glass or even colored glass; they need to be light-generating on the side near your eye and controllably transparent on the side away from your eye.

The glasses would come with a set of lightweight headphones to connect up to computer-controlled sound synthesis equipment, to provide yet another dimension of sensory illusion. The head-mounted display would become a universal tool for interacting with one's environment. In the office, a virtual desk would extend in front of the wearer, with an arbitrary number of papers and books; anything can be brought to the "real" desk where all interactions are monitored and recorded by the computer. An object changes as it is manipulated, and effects of that change, if any, propagate throughout the simulated environment. Thus, taking a book from the bottom of a stack may cause the remaining books to topple, but a spoken word or gesture will get them to re-stack themselves with no trouble.

Surgeons could be supplied with projections of the patient upon whom they are considering surgery. They can plan and even practice the surgery, and watch computer simulations of the results in a greatly accelerated time frame. When real surgery is being performed any information the surgeon desires, from a magnification of the work being done to x-ray views of the patient, can be supplied instantly. The artistic and entertainment possibilites are open to the imagination. Giant networks of people interacting with each other's virtual images all over the globe could gather to see cultural events or synthesize their own. With the distributed home computer network now growing the future of such a display could change nature every few years.

Today's Displays

Recent work by Shiffman and Parker [Schif 84] and Vuilleumier et al [Vuil 84] have shown that it's possible to construct very small high resolution displays using IC fabrication technology for both the driving electronics as well as for the display substrate itself. Such work encourages us in the belief that displays can be constructed with sufficient size, speed, and resolution for useful head-mounted displays.

Today's Problems

The displays mentioned above are all relatively new technology that will certainly have to go through several design iterations until all the bugs are worked through, if indeed all the bugs can be solved. These new displays also suffer from some common problems. They are not very powerful light sources, so either the light must be used very carefully or a light "repeater" will have to be used between the display and the eye. Additionally, they are all physically very small. To generate a display large enough to operate as a head-mounted display their image must be magnified in some way.

V. Tracking

The goal of tracking is to make the computer responsible for maintaining information describing where the user is looking and what he or she is doing with respect to the database. For example, in the head-mounted display paradigm we require that the computer know where we are looking. This is the opposite philosophy from sitting in front of a display tube with joysticks and sliders and manually entering your position and gaze; we want the computer to keep track and do the work itself.

There are two major reasons we want to track the user: to provide the correct images to the head-mounted display, and to modify the database correctly to respond to the user's movements. For the purpose of accurate viewpoint determination, we need to know at a minimum where the user's head is located and in what direction the eyes are looking. To dynamically modify the database when interacting with the user, we can progress from gross physical motions to subtle cues depending on our expertise, goals, and funding.

A minimum tracking system would keep track of the user's position and gaze, and the tip of a wand held in the user's hands. We suggest that the first improvement to make to this system would be to track the user's hands. Gestures constitute a large part of our everyday communication, and our hands are a major mechanism of interaction with our surroundings.

The next two subsections will investigate the two major approaches to tracking: the machine tracking the user, and the user tracking himself.

Machine Tracking User

The head-mounted display requires that we know where the user is located and in what direction the user is looking. Database interaction requires that we know how the user is interacting with the world. We will first look at a device that enables tracking of the user's head and gaze, and then we will look at two devices that allow the user to interact with the database.

Viewpoint via Mechanical Coupling: The Sword of Damocles

In 1968 Sutherland published an account of the first interactive head-mounted display [Suth 68] (also described in [Vick 70]). The means of tracking the user was via a device formally named the "Head Position Sensor." This was a shaft of nested cylindrical tubes that ran between two pivots: one on the ceiling of the room and one on the top of a helmet-like contraption the user wore on his or her head. The computer determined where the head was located and the angle at which it was oriented by measuring the overall length from ceiling to head and the values of the angles at the two pivots.

Ivan Sutherland is said to have referred to this device as "The Sword of Damocles," in reference to the overhead contraption. The optics in the headpiece projected the synthetic image to a focus point apparently several feet in front of the viewer.

Interaction via Visual Coupling: The Wand

The first device to be used with the Utah head-mounted display was a wand. This was simply a stick with a light at the end, which was driven by some light-camera synchronization circuitry. The blinking light was sensed by cameras mounted on the walls of the room, which were also controlled by the synchronization circuits. Although the implementation of such a scheme is simple, it suffers from some severe problems. First, the light must always be visible by the cameras. Thus, the system builder must either buy many cameras and mount them throughout the room, or severly modify where the wand may be held and where the user may stand. Another problem is that since the light must be differentiable from the image of the room, it must be sufficiently bright to be easily distinguishable. This implies either that the room be dark (which may be troublesome to the user), or that the light be very bright (and possibly irritating). There are also other drawbacks. So although the problems with this scheme are manifold, it does represent a first step towards virtual interaction.

Interaction via Mechanical Coupling: The Arms

In 1976 Kilpatrick reported on a system called GROPE [Kilp 76], built at the University of North Carolina at Chapel Hill. GROPE consisted of two mechanical arms, a master arm grip and slave arm tongs. The system was able to perceive the location of the user's hand by measurement of the geometry of the arm mechanics, much like Utah's Sword of Damocles. The UNC system was able to provide force feedback through the arm, to simulate the actual resistance objects in the database would present to changes in their inertia.

Kilpatrick reports on a series of experiments he ran with a set of subjects. The subjects were placed in front of a computer-controlled vector CRT which presented them with an image of the database. Included in this image was a robot arm similar to the master arm. The subject then placed his hand into a grip on the end of the master arm and interacted with the database, using the force feedback in the arm and the image in the vector tube as cues.

This system also had its drawbacks. Most important was the potential harm to the user in case of system failure. Large, powerful metal arms driven by computer-controlled motors may be dangerous or even lethal in the hands of hardware or software error. Also, the force feedback in the master arm may go too far and hurt the user. Although many safety devices and interlocks were built into the system, there is always the possibility that something could go wrong.

Viewpoint via Visual Coupling: Gaze Following

In 1981 Bolt described a system which reacted to the perceived location of the user's gaze. The user was constrained to sit in a chair located in the center of a large room. A TV camera sensitive to infrared light was zoomed into a close-up of the user's eyes. Mounted fairly high up in the room was an infrared light source. The light was unobtrusive to the user, yet the reflections of the light on the user's eye provided sufficient cues to determine the gaze angle to within 1 degree. The combination of the gaze angles of the two eyes yielded the three-dimensional point upon which the user's eyes were focused. As in all the other schemes mentioned so far, the user was restricted in how far he or she could wander in the room. This system also limited the range of gazepoints to within 10 to 30 degrees off of some reference axis. However, this system was superior to the others above in terms of the unobtrusiveness of the measuring device.

User Tracking Self

An alternative solution to the tracking problem is to reverse the location of the responsible hardware from the above schemes. This statement implies that tracking responsibility should move away from the computer and back to the user, which seems to defeat the entire notion of tracking. The approach is not to make the user himself responsible, but to attach some hardware to the user which will make itself responsible for accurate tracking of the user.

There is a significant implication in this statement. In the previous schemes the user was restricted in some critical ways because of the nature in which measurements were being taken of him or her. But if the user is measuring his own position, then there should be no restrictions. Thus, when a user tracks himself, many of the range restrictions on the parameters of interest are relaxed or eliminated.

Decoupling: The Self-Tracker

In 1984 Bishop presented information on a custom VLSI chip he called the Self-Tracker [Bish 84a], [Bish 84b]. The chip consisted of a one-dimensional array of photosensors, control circuitry, and a processor. The chip performed tracking by analyzing successive images falling on its 1-dimensional photosensor array. By mounting several of the chips in a cluster at angles to each other, Bishop was able to show that his system could determine its own movement between successive sample intervals. By maintaining a small sampling interval the Self-Tracker was able to hold numerical problems in check and effectively track its own motion - translation and rotation - in virtually any environment.

The Self-Tracker as presented did not completely solve all the problems associated with a device tracking itself, but it did solve many of them and it was a major conceptual and technical step forward. By mounting a Self-Tracker cluster on a user's head, the front sensors would be looking very nearly where the eyes would be looking. The only communication needed out of the cluster would be successive transformation information.

VI. Interactive Databases

When the head-mounted display paradigm comes of age we will have solved many of the problems mentioned so far throughout this paper. But the freedom and versatility of the head-mounted display will cause a new pressure to arise in the user community: interactive databases. The study of databases is a well-established field, and complex databases are common in sophisticated systems such as flight and space simulators. But although there may be a great deal of interaction between a user and a database in such a system, there has so far been little work on graphics databases that support user-user, user-object, and object-object interaction.

When a user in a head-mounted environment interacts with some piece of data, he or she will naturally expect the rest of the environment to conform to the change. For example, consider a user wearing a head-mounted display which places her on a dirty field, standing next to a marbles game. The user pitches a virtual marble forward into the ring, and watches as the marble strikes other marbles, which in turn strike other marbles, and so on until all the marbles roll to a halt. If one of the marbles happens to fly through the air and hit the windshield of a nearby car, the windshield may shatter and the marble pass through into the car, possibly rolling around a bit before setting down. The bits of windshield glass would then be scattered on the ground, and may affect future shots in the marbles game. This contrived example only indicates the difficulties that a self-interacting database will have to solve. The issues of a shattering windshield and real-time, interacting marbles are beyond the capabilites of almost all graphics database managers today.

It is our opinion that graphics databases are going to soon be modelled with intelligent, communicating objects, very much like objects in the language Smalltalk. We suggest that distributed intelligence is the best tool computer science has right now for coping with the incredible complexity of environments on the order of natural environments.

Happily, these techniques are the object of great study by many people in computer science today, who are looking to them for applications from artificial intelligence to parallel processing. The architectures and techniques for managing large numbers of intelligent, communicating objects will be critical to the success of a head-mounted display environment.

VII. Acknowledgements

Several of the figures in this paper were adapted from similar figures created by Greg Abram for publication in [Abra 84]. We gratefully acknowledge his permission to use them in this paper. The authors extend their thanks to Jeff Hultquist for his many contributions to this project, including Figures 1 and 2, help on wording critical passages, a very thorough critical review, a tolerant ear, and an intolerant blue pencil. Thanks also go to Bobette Eckland, who found many textual errors in the original manuscript and offered thoughtful editorial advice.

X. References

[Abra 84]	"VLSI Architectures for Computer Graphics", G. Abram, H. Fuchs, Proceedings of the NATO Advanced Study Institue on Microarchitecture of VLSI Computers, July 1984
[Appe 67]	"The Notion of Quantitative Invisibility and the Machine Rendering of Solids", A. Appel, Proceedings of the ACM National Conference, Washington DC, 1967
[Bish 84a]	"The Self-Tracker : A Smart Optical Sensor on Silicon", Gary Bishop and Henry Fuchs, 1984 Conference on Advanced Research in VLSI, M.I.T.
[Bish 84b]	"Self-Tracker : A Smart Optical Sensor on Silicon", Gary Bishop, Ph.D. thesis, UNC-CH, 1984
[Bolt 81]	"Gaze-Orchestrated Dynamic Windows", Richard A. Bolt, Siggraph Volume 15 Number 3, August 1981
[Carp 84]	"The A-buffer, an Antialiased Hidden Surface Method", Loren Carpenter, Siggraph Volume 18 Number 3, July 1984
[Clar 80]	"Distributed Processing in a High-Performance Smart Image Memory", J. Clark, M. Hannah, VLSI Design, Volume 1 Number 3, 4th Quarter, 1980
[Clark 82]	"The Geometry Engine: A VLSI Geometry System for Graphics", James Clark, Siggraph Volume 16 Number 3, July 1982
[Cohe 80]	Presentation at Siggraph 80, D. Cohen, S. Demetrescu
[Cook 84]	"Distributed Ray Tracing" R. Cook, T. Porter, L. Carpenter, Siggraph Volume 18 Number 3, July 1984
[Deme 85]	"High Speed Image Rasterization Using Scan Line Access Memories", Stefan Demetrescu, 1985 Chapel Hill Conference on VLSI Proceedings, Computer Science Press

[Dipp 84] "An Adaptive Subdivision Algorithm and Parallel Architecture for Realistic Image Synthesis", Mark Dippe and John Swenson, Siggraph Volume 18 Number 3, July 1984

[Fium 83] "A Parallel Scan Conversion Algorithm with Anti-Aliasing, for a General-Purpose Ultracomputer", E. Fiume, A. Fournier, L. Rudolph, Siggraph 83, Volume 17, Number 3 July 1983

[Fuch 77] "Distributing A Visible Surface Algorithm Over Multiple Processors", H. Fuchs, Proc. 1977 Annual ACM Annual Conference, October 1977

[Fuch 79] "An Expandable Multiprocessor Architecture for Video Graphics", H. Fuchs, B. Johnson, Proc. of 6th Annual ACM-IEEE Symposium on Computer Architectures, April 1979

[Fuch 81] "PIXEL-PLANES: A VLSI-Oriented Design for a Raster Graphics Engine", H. Fuchs, J. Poulton, VLSI Design, Volume 2, Number 3, 3rd Quarter, 1981

[Fuch 85] "Fast Spheres, Shadows, Textures, Transparencies, and Image Enhancements in Pixel Planes", H. Fuchs, J. Goldfeather, J. Hultquist, S. Spach, J. Austin, J. Eyles, J. Poulton, to appear in Siggraph Volume 19, Just 1985

[Fuss 82] "A VLSI-Oriented Architecture For Real-Time Raster Display of Shaded Polygons", D. Fussell, Graphics Interface '83, May 1982

[Gupt 81] "A VLSI Architecture for Updating Raster Scan Displays", S. Gupta, R. Sproull, I. Sutherland, Siggraph 81, Volume 15, Number 3, August 1981

[Kaji 75] "A Random-Access Video Frame Buffer", J. Kajiya, I. Sutherland, E. Cheadle, Proc. IEEE Conference on Computer Graphics, Pattern Recognition and Data Structure, May 1975

[Kede 84] "Computer Structures for Curve-Solid Classification in Geometric Modelling", Gershon Kedem and John Ellis, Rochester Institute of Technology TR 137, May 1984

[Kilp 76] "The Use of A Kinesthetic Supplement in An Interactive Graphics System", Paul Jerome Kilpatrick, Ph.D. thesis, UNC-CH, 1976

[Newe 72] "A New Approach to the Shaded Picture Problem", M. Newell, R. Newell, T. Sancha, Proc. ACM National Conference, 1972

[Park 80] "Simulation and Expected Performance Analysis of Multiple Processor Z-Buffer Systems", F. Parke, Siggraph 80, Volume 14 Number 3, July 1980

[Poul 85] "PIXEL-PLANES: Building A VLSI-Based Raster Graphics System", J. Poulton, H. Fuchs, J. Austin, J. Eyles, J. Heinecke, C. Hsieh, J. Goldfeather, J. Hultquist, S. Spach, to appear in 1985 Chapel Hill Conference on VLSI Proceedings, Computer Science Press

[Robe 63] "Machine Perception of Three-Dimensional Solids", L. Roberts, MIT Lincoln Lab TR 315, May 1963

[Schu 69] "Study for Applying Computer Generated Images to Simulation", R. Schumacker, B. Brand, M. Gilliland, W. Sharp, AFHRL-TR-69-14, Air Force Human Resources Lab, Wright-Patterson AFB, OH September 1969

[Shif 84] "An Electrophoretic Image Display With Internal NMOS Address Logic and Display Drivers", R.R. Shiffman and R.H. Parker Proceedings of the Society forInformation Display, 25(2), 105-152

[Suth 65] "The Ultimate Display", I. Sutherland, Proceedings of 1965 IFIP Conference

[Suth 68] "A Head-mounted Three Dimensional Display", I. Sutherland, FJCC 1968, Washington DC, 1968

[Suth 74] "A Characterization of Ten Hidden-Surface Algorithms", Ivan Sutherland, Robert Sproull, Robert Schumaker, Computing Surveys, Volume 6 Number 1, March 1974

[Vick 70] "Head Mounted Display", Donald L. Vickers. utechnic magazine, Volume 10, Number 4, May 1970

[Vuil 84] "Novel Electromechanical Microshutter Display Device", R. Vuillemier, A. Perret, F. Porret, P. Weiss, Proceedings of the 1984 Eurodisplay Conference, Society for Information Display, July 1984, Paris.

[Watk 70] "A Real Time Visible Surface Algorithm", G. Watkins, Ph.D. dissertation, TR 70-101, University of Utah, June 1970

[Weil 77] "Hidden Surface Removal Using Polygon Area Sorting",
K. Weiler, P. Atherton, Siggraph 77, Volume 11 Number 2
Summer 1977

[Wein 81] "Parallel Processing Image Synthesis and Anti-Aliasing",
Richard Weinberg, Siggraph Volume 15 Number 3,
August 1981

[Whit 80] "An Improved Illumination Model for Shaded Display",
T. Whitted, CACM Volume 23 Number 6, June 1980