Fuchs88B

Prepared for

NATO International Advanced Study Institute
Theoretical Foundations of Computer Graphics and CAD
Il Ciocco International Centre
Castelvecchio Pascoli, Lucca, Tuscany, Italy
4 - 17 July 1987

An Introduction to Pixel-planes and other VLSI-intensive Graphics Systems

Henry Fuchs
Department of Computer Science
University of North Carolina
Chapel Hill, NC 27514 USA

Abstract

This paper, an updating of [Fuchs, 1987], reviews some experimental and commercial graphics systems that intensively use VLSI (Very Large Scale Integration) technology. Described in some detail is the current state of one of these systems, our own Pixel-planes. Among the fastest of 3D systems, Pixel-planes renders about 35,000 full-screen, smooth-shaded, Z-buffered triangles per second, about 13,000 Z-buffered interpenetrating spheres per second.

Introduction

Computer graphics has always been an expensive proposition and its users a demanding, unsatisfied lot-- the picture never got onto the screen fast enough, or later, never moved fast enough once it got to the screen, and then, the picture was never sharp enough, never realistic enough. Many users have been tackling problems that could only be solved with the highest performance graphics systems -- the crystallographer trying to understand the structure of a complex protein from noisy data, the radiologist trying to detect a possible tumor amid the clutter of healthy tissue, an architect trying to "walk" the client through the still-unbuilt house with only the images on a video screen. This paper describes the answers of various graphic system designers to this need. Regrettably, because the internal details of commercial systems are often not published and are revealed to outsiders only on a non-disclosure basis, some systems that the authors would like to have selected could not be described in this paper. Since this field moves very rapidly, it should also be noted that this paper was compiled during Fall, 1986.

Historical Background and Overview

Graphics displays have been a part of computers since at least 1950 with the point-plotting CRT on MIT's Whirlwind 1 computer [Everett, 1952]. The random-deflection CRT, driven by a refresh list of point locations, was the basis for virtually all graphics displays through the 1960's. By 1968, at least one system had general 4x4 matrix transformations, clipping, and perspective divide [Sutherland, 1968; Sproull and Sutherland, 1968]. This paved the way for commercial systems with real-time manipulation of 3D wire-frame models with perspective [Evans & Sutherland, 1971]. These capabilities are the ones still found in today's high-performance random-scan vector systems.

The major drawback with all the random-scan vector systems has been their inability to produce realistically-rendered objects and scenes. Although algorithm development for generating continuous-tone renderings increased throughout the 1960's and early 1970's, the only readily-available output devices were film recorders. These took many seconds or minutes to expose a film in front of a CRT, on whose face was scanned out a sequence of hundreds of thousands of individual positions, each with a distinct intensity. Numerous users and system builders dreamed of dynamically interactive continuous-tone image generation, but until the mid-1970's, only very expensive systems, mostly visual subsystems of flight simulators, could afford this capability [Schumacker, Brand, Gilliland, and Sharp, 1969; Watkins, 1970; Schachter, 1981]. Continuous-tone digital images on video monitors were available, however, for image processing applications. These systems displayed the contents of an image buffer memory bank continuously on the video monitor. Unfortunately, these systems were of only limited use for computer graphics because their storage organization typically allowed only very restricted access to the pixel values. The medium was usually a disk track or a cyclical shift register that was only serially accessible. Thus, updating of random pixel values was unacceptably slow. In some systems, for example, it was only possible to change a pixel when its screen refresh time came around, once very 30 milliseconds.

The major turning point in graphics systems development came when RAM's (random-access memory chips) became large enough (4kbits per chip) so that with a few hundred chips an entire video image could be stored [Kajiya, Sutherland, Cheadle, 1974]. (It is interesting to note, however, that certain researchers had implemented such random-access "frame buffers" much before this. Professor T. Kunii of the University of Tokyo relates [Kunii, 1986] that he had bulit one of these with core memory in the late 1960's.) Once random-access frame buffers became affordable, in the late 1970's, they quickly became the system of choice for many users. They produced continuous tone, color images, and, through various techniques with the color translation tables and addressing methods, these systems could produce a variety of useful effects -- zooming, double buffering, simple animation, and moving overlays over a fixed background. Their prevalence has helped merge graphics and image processing. Combining flexible use of raster memory with built-in general processors, these frame buffer systems have steadily taken over areas previously dominated by the random-scan vector systems. A good example of such a flexible raster system is described in [England, 1981]; its successor is currently available as the Adage RDS 3000. The raster systems did, however, have some weaknesses: they did not create or modified their images as fast as random-scan "vector" systems, and their images usually showed the "jaggy" artifacts of crude pixel representation. Overcoming these limitations has been a major goal of many of the designs described in the following sections.

A confluence of several people's varying interests in graphics, text processing, and flexible personal computers led to the development in 1973 at the Xerox Palo Alto Research Center of the ALTO personal computer [Kay, 1977; Thacker, McCreight, Lampson, Sproull, Boggs, 1979]. Its basic design was a high-resolution B/W display (1 bit/pixel), dedicated processor, mouse for x,y input, and a high-speed interface to a local area network. It has been copied widely into successively newer technologies [Bechtolsheim and Baskett, 1980] and has spawned the new industry of personal, professional workstations. Although these systems are often used simply for text display with multiple fonts styles, their flexibility allows the rapid drawing of lines and some simple 2D textures. The ALTO also pioneered the use of multiple, overlapping windows of arbitrary size, inside each of which could be any combination of text, graphics, and images. To facilitate rapid movement of text about the screen and between screen and off-screen memory, the machine included a BitBlt (for Bit-Aligned Block Transfer) instruction. Variations and generalizations of this instruction have been widely adopted, often under different names (Raster-Ops, Pix-BLT).

Each of these three kinds of graphics systems has been optimized for a different mix of tasks: a) the high-performance 3-D interactive system, b) the color frame-buffer display, and c) the personal workstation. While these three kinds of systems have blended together over the years, each has certain strengths that its designers have tried to bolster:

a) for the high-performance 3-D system:
- fast geometric transformation and clipping of primitives (typically, lines and polygons)

- fast lighting calculations
- fast rendering of primitives

b) for color frame buffers:
- many bits per pixel and flexible ways of using them for multiple frames, background, overlays, etc.
- fast, general processor for executing a variety of algorithms

c) for general workstation display:
- high spatial resolution to display as many windows with as much information as possible
- fast text generation of multiple fonts, sizes and styles
- fast movement of text about the screen and to/from off-screen memory

As we shall see in the next section, with the increasing similarity between the three kinds of systems, manufacturers have started to offer systems that attempt to satisfy two and occasionally all three of these areas with a single system.

Taxonomy

Although the field is still too vague to arrange a definitive taxonomy, it may be useful to arrange the components around the simplified functional organization sketched in Figure 1. It should be noted that a number of the referenced systems implement particular functions, such as rendering, with a totally different method than that indicated in the figure. They are listed in the figure to show the extent of their function in comparison to other systems. The brackets indicate the extent of each defined component; It is understood that a working system constructed from any of the referenced machines would include all the functional units from data base to video monitor.

A VLSI Sampler

The systems reviewed here are roughly in chronological order, but the order is occasionally modified for clarity of presentation.

Graphics Display Controllers. Among the earliest custom chips for graphics were the various video timing chips that generated the complex synchronization and blanking signals needed to drive a video display. These also controlled the timing of the access to the frame buffer memories for refreshing the screen. As silicon structures shrank in size and more transistors could be integrated into the same chip, the video chips increasingly took on tasks involved with the access to the frame-buffer memories for image generation. The first such chip that was generally available and could also handle low- level image generation, with help from some processor, was the NEC PD7220 GDC. It made possible the construction of frame buffers with much lower parts count and thus much lower cost. For example, an early system to use this chip, the Vectrix Corp. (Greensboro, NC) VX 128, contained a 480 x 672 pixel by 3 bits/pixel frame buffer, an internal 16-bit processor (Intel 8088), serial and parallel ports, separate package and power supply, and cost $2,000 in early 1983. The chip launched a flurry of under-$10,000 frame buffers and stimulated numerous "improved" versions from other chip vendors, some of which are described further below.

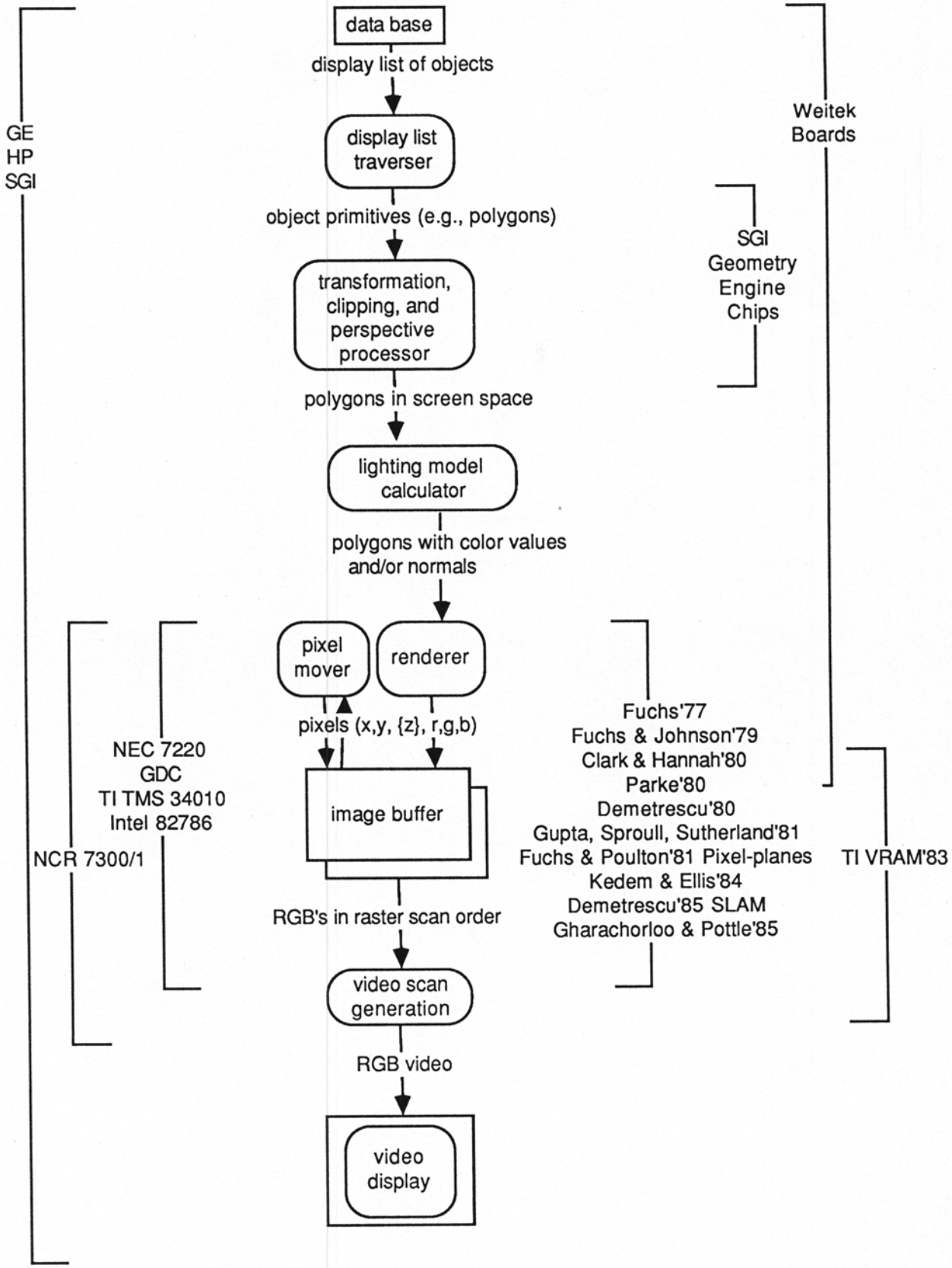

Figure 1: Generic Organization of 2D/3D Raster Systems

SGI's Geometry Engine. This chip, which performs arithmetic operations on 4-element vectors, was introduced in [Clark, 1980] and more fully described in [Clark, 1982]. It may have been the first custom VLSI chip for the 3D graphics market. This chip is the building block of a geometric transformation and clipping engine that is composed of a pipeline of a dozen of these chips. Each chip is configured to execute one part of the classic 3D processing pipeline as outlined in [Sutherland, 1968; Sproull and Sutherland, 1968; and Sutherland and Hodgman, 1974]. The chip is basically a processor with four identical arithmetic units controlled by hardwired microcode that implements the various steps of matrix operations, clipping, and perspective divide. Each of the four units handles one of the elements of the 4-element vector in the commonly used homogeneous coordinate representation for handling 3D projective transformations. The arithmetic units handle 32-bit floating point representation, but due to silicon space limitations, many operations, such as multiply, take many clock cycles to perform. Nevertheless, with four arithmetic units per chip, and with a dozen chips in the pipeline, a 3D point is transformed in 15 microseconds. Improvements, made since 1982 in the commercial version of this chip from Silicon Graphics Inc., have enhanced its throughput considerably.

Transformations and Rendering with General Purpose Arithmetic Chips. The classic alternative to implementing a geometric transformation unit with custom chips is to implement it with general-purpose standard arithmetic units. [Weitek, 1986] describes the recommendations of one of the most popular suppliers of these fast arithmetic chips. Users of these general chips expect that their faster inherent speed will allow them to use fewer units than those of the slower custom variety from Silicon Graphics. In addition, the general-purpose chips allow coding of a wider variety of algorithms into the same unit. For example, the board-level products from Weitek not only perform geometric transformation, but also do rendering of planar and curved primitives. However, since the units have to transfer the pixels into an external frame buffer for screen refresh, their application to dynamically interactive systems has so far been limited. They do best on objects and scenes that take several seconds to transform and render.

Dividing Frame Buffer Among Multiple Processors. Several designs have aimed to speed up the rendering of primitives into the frame buffer by dividing the frame buffer among multiple, usually identical processors. The frame buffer can be divided in an interleaved fashion so that every eighth pixel in a row and every eighth pixel in a column is assigned to the same memory. In this way a geometric primitive such as a polygon or line segment is sure to fall into many of the frame-buffer sections and thus sure to be processed by many of the processing elements [Fuchs, 1977]. [Fuchs and Johnson, 1979] describe an implementation of this scheme using 8-bit microprocessors for the processing elements. [Clark and Hannah, 1980] describe a variation of this scheme with an additional layer of eight processors, each responsible for all the processors in every eighth row. These higher-level processors could perform the set-up calculations so that start-up costs for rendering a primitive are reduced.

[Gupta, Sproull, and Sutherland, 1981] describe a system with a similarly interleaved 64-piece frame buffer with processors optimized for data moving for workstation displays. In this system, each processor is connected to its 8 neighbors (up, down, left, right, and four diagonals) plus 3 others far away. Movement of a rectangular region can be accomplished by reading a "footprint" of 64 pixels into the processors, transferring the pixels among the processors and having each processor write its newly acquired pixel back into its memory. A similar system is described in [Sproull, Sutherland, Thompson, and Minter, 1983]. An alternative subdivision of the screen is suggested in [Parke, 1980]. Here the screen is successively subdivided in half. A memory portion and a rendering processor are stored at each leaf of a binary tree whose internal nodes each contain a "splitting" processor. A primitive travels down the tree, being steered to the region in which it is located and split if it straddles more than one region. The hope is that only a few of the primitives will end up at each root, and thus the total rendering load will be shared evenly among all the rendering processors. Various interleaved and successive subdivision schemes are analyzed in [Parks, 1982].

Processor-per-Polygon Pipeline. [Demetrescu, 1980] describes a novel architecture for rapid rendering of many polygons using a distributed Z buffer algorithm. The system consists of a pipeline of processing chips, each responsible for a single polygon. Each chip has one input and one output port. It receives a packet of color and Z values through the input port and sends either the same packet or a new packet to the output port. The packets move through the pipeline in raster-scan order so that each chip

knows the X,Y address of the current pixel. To determine which packet a processor sends out, it first determines whether its polygon covers the current pixel at all--most of the time it will not. If it does, the processor compares its polygon's Z value with the Z value it received from its input and outputs the color and Z value of whichever has the smaller Z. Thus, the polygon that is output is the nearest one among all the ones in the pipeline. If the set-up values can be loaded into all the polygons fast enough, and if each processor can complete its task relating to a pixel in one pixel's video refresh time, then the system might run in real-time. The chip was designed and built, but not debugged [Demetrescu, 1986].

An extension to handle anti-aliasing is proposed in [Weinberg, 1981]. In this scheme, the pipeline would not only pass color and Z values for a pixel, it would pass a sequence of polygon parts that would be visible within the pixel area. If a polygon processor found its polygon to be partially visible in the current pixel, it would add the portion of its polygon lying within the pixel to the sequence of inputs it received for the current pixel. Also, it would cull from the list any polygon partsthat became obscured by its own polygon. The processor would then output all polygon parts still visible at that pixel. This enhancement significantly increases the amount of data passing through the pipeline and the amount of work required from each polygon processor. However, it allows a filter processor at the end of the pipeline to combine appropriately all visible portions of polygons for each pixel and, therefore, calculate a reasonable anti-aliased image.

Processor-per-CSG-Primitive Tree Machine. [Kedem and Ellis, 1984] introduce a design for a machine that could render solid objects directly from their CSG (Constructive Solid Geometry) desriptions. A CSG tree consists of a primitive (such as cylinders and rectangular solids) at each leaf node and a set operator such as union and intersection at each internal node. The machine consists of a reconfigurable collection of processors onto which the CSG tree of the object is directly mapped. (It is assumed that the number of processors in the machine is typically larger than the nodes in the tree, although larger trees could be processed with multiple passes.) There are two kinds of processors, one to render a CSG primitive, such as a cylinder or a rectangular solid, and another to perform the set operation on the output stream from two renderers. Each rendering processor calculates the Z values and perhaps the shade, for its primitive in each pixel, in raster scan order. This stream of data is passed up the tree to a processor that performs the required set operation on this and another stream. The output at the root of the tree consists of a stream of packets, one for each pixel. The first color of each pixel packet is the color of the visible surface. These values are stored in a conventional frame buffer for screen refresh. The machine is currently under construction.

Rectangle-filling Memory Chip. A memory chip that rapidly fills axially-oriented rectangles of constant color is proposed in [Whelan, 1982]. In this design, the addressing structures of both the row and column within a memory grid decode a minimum and a maximum address and propagate the enable signal to all cells rows (or columns) in between. Cells, whose row and column are both enabled, are changed to the new value.

Video RAM. Introduced by Texas Instruments in 1983, the video RAM TMS4161 elegantly solved the problem of accessing a high-resolution frame buffer memory for screen refresh [Pinkham, Novak, and Guttag, 1983]. The 64Kb x 1 DRAM includes an internal 256 x 1 bit shift register which can be accessed independently from the rest of the chip. In one memory cycle, an entire row is transferred from the main memory to the shift register. Data can be shifted out of this register at up to 25MHz. During this read-out, the main memory is free to be accessed by the image generator. With standard DRAM's in a high-resolution frame buffer, up to 50% or more of memory cycles may go to screen refresh. This may significantly slow down the image generation system. The use of the VRAM can reduce this access rate to less than 2% [Guttag, Van Aken, and Asal, 1986]. To achieve a similarly low rate with standard RAM's, an elaborate memory organization with numerous extra parts may be needed. However, if dual frame buffers can be implemented, then one entire buffer can be devoted to screen refresh while the image generation system is creating the next image in the other one. This effectively eliminates the interference between the two systems, but at a substantial parts cost. (See [Whitton, 1984] for a good discussion of this topic.)

Scan-Line Access Memory. [Demetrescu, 1985] describes a memory chip enhanced to allow

reading or writing of an entire row of the memory in a single cycle. The memory grid in the chip is treated as a rectangular part of one bit-plane of the frame buffer. Access to the chip is via op-codes and values that allow setting of the Y row, the X-Left and the X-Right edges for writing a specific span of the row (via read-modify-write), and setting of a 16-bit fill pattern. When writing a sequence of spans, the chip assumes reasonable values for the parameters in order to minimize the number of cycles taken up specifying them: the Y register increments, the X registers remain constant. Thus, filling a rectangular area can be accomplished with a single read-modify-write cycle per scan line. For instance, in order to fill a polygon, it is sufficient to specify the starting row and thereafter only the left and right boundaries for each successive row. Characters are written one scan-segment at a time. As each successive horizontal segment is loaded and the write command invoked, the proper addresses will be already in place (Y incremented, X's remain the same). A small system consisting of some dozen chips has been built and demonstrated [Demetrescu, 1986].

Processor-per-pixel on a Scan-Line. [Gharachorloo and Pottle, 1985] describe a rendering system that is a combination of the classic Watkins scan-line processor [Watkins, 1970] and a variation on the processor-per-polygon design of [Demetrescu, 1980]. It sorts polygons by their top-most scan line and maintains an active queue of the polygons crossing the current scan line. For each of these, it calculates the left and right boundaries, and the starting and incremental Z values and colors. These are passed to a string of processors, one for each pixel on the scan-line. Each such processor performs a Z-buffer algorithm for its pixel on each data packet and passes the incremented values to its neighbor. The video data is scanned out of these pixel processors in the opposite direction of the packets: the packets travel right, the video data travels left. This system is currently being further developed at IBM TJ Watson Research Center in Yorktown Heights, NY [Gharachorloo, 1987].

TI's Programmable Graphics Processor. The Texas Instruments TMS34010 is a fully programmable single-chip 32-bit CPU [Asal et al,1986]. Although it still handles video control and timing, the chip is expected to be programmed in C to perform a variety of image-generating tasks more quickly than a general purpose 32-bit microprocessor. The processor contains 30 general purpose registers, stack instructions, a barrel shifter, field selection and control logic, and an instruction cache. It executes most instructions at 6MHz. Its instruction set has been enhanced with graphics-specific codes such as block-move-and-modify RasterOps, and such variants of arithmetic operations as Maximum and Add-with-Saturate for combining multiple image patterns. The chip's designers appear to emphasize text generation and movement. It is not yet clear how suitable the chip will prove to be for image-generation tasks such as vector generation, polygon fill, and shading and lighting calculations. The approach is certainly reminiscent of the "wheel of reincarnation" effect described in [Myer and Sutherland, 1968]. They note that designers tend to add more and more registers and functions to a display processor until it becomes essentially a CPU again, at which time it has become sufficiently slow and inefficient so that the designers put on it a small, fast display processor, and thus the cycle starts again. It should be no surprise that there are so many functions implemented within this chip; with some 200,000 transistors, it is three times the size of the popular Motorola 68000 32-bit CPU that is the main processor in many professional workstations.

NCR Low-Cost Integrated Controller. NCR recently announced its 7300 and 7301 chips set for the low-to-medium priced desk-top personal computers [Electronics "NCR ..", 1986] . The system facilitates low parts count by including within the 7300 character generation for two complete fonts as well as a look-up table and four 4-bit DAC's. The chip set can also control up to 8 windows.

Intel 82786 Integrated Controller. This integrated controller that is really three nearly-separate processors on the same chip [Shires, 1986; Electronics "Intel ..", 1986]. In addition to the usual frame buffer refresh control, there is novel high-level control of virtually any number of windows and a separate graphics processor for drawing into those windows. The graphics processor can generate text, lines, circles, and other geometric primitives. To control multiple windows, the CPU supplies the display processor with a map of the parts of various windows it wants displayed on the screen. The display processor fetches the appropriate pixels from main graphics memory.

Other Systems. An early disclosure of a graphics controller chip from Advanced Micro Devices

appeared in the June 27, 1985 issue of *Electronic Design*. General Electric's Silicon Systems Technology Department developed the Graphicon 700 Graphics Processor which does high-speed rendering using several new custom chips of their own design [General Electric]. Another new high-speed rendering system, this one from Hewlett-Packard, is described in [Swanson and Thayer, 1986]. A chip set from National Semiconductor is described in [Carinalli and Blair, 1986]. [Dill and Grimes, 1986] includes [Asal et al, 1986], [Carinalli and Blair, 1986], and [Shires, 1986]. [Lineback, 86] claims that nearly a dozen new designs are being developed currently in various chip houses in England, the USA and Japan.

Pixel-planes

Overview. The rest of this paper is devoted to a more detailed description of one particular VLSI-intensive graphics system, Pixel-planes. This high-speed rendering system features a 'frame buffer' composed of custom logic-enhanced memory chips that can be programmed to perform most of the time-consuming pixel-oriented tasks in parallel at each pixel. The novel feature of this approach is a unified mathematical formulation for these tasks and an efficient tree-structured computation unit that calculates inside each chip the proper values for every pixel in parallel. The current system contains 512 x 512 pixels x 72 bits per pixel implemented with 2,048 custom 3micron nMOS chips (63,000 transistors in each, operating at 10 million micro-instructions per second). New algorithms for rendering spheres (for molecular modeling), for adding shadows, and for enhancing medical images have been devised by various individuals within and also outside the group. The current system will shortly be placed in our graphics laboratory for use in molecular modeling, medical imaging, and architecture. We are building the next version of the system which should be available in 1988. This upgraded system will have chips that are much faster and that allow direct rendering of certain curved surfaces.

Concept. We are exploring the utility of a radical approach to raster graphics, having the front part of the system *specify* the objects on the screen in pixel-independent terms and having the memory chips themselves work directly from this description to generate the final image. The image primitives such as lines, polygons and spheres are each described by expressions (and operations) that are "linear in screen space", that is, coefficients A,B,C such that the value desired at each pixel is Ax+By+C, where x,y is the pixel's location on the screen. Thus the information going to the 'frame buffer' is not address, data pairs (x,y addresses, RGB data), but ABC's and operation codes. In contrast to other raster systems, the most time-consuming calculations are not done by (1) general purpose computers or (2) special hardware that executes only a particular set of graphics functions. Instead, *Pixel-planes* is a rather general-purpose raster engine, especially powerful when most of the *pixel* operations can be described in linear (or *planar*) expressions. The system was introduced in [Fuchs and Poulton, 1981] and further developed in [Fuchs, Poulton, Paeth and Bell, 1982]. Basic algorithm descriptions to perform polygon rendering, Z-buffer tests, Gouraud-shading, as well as more elaborate algorithms such as spherical display and shading, and shadow casting can be found in [Fuchs et al, 1985]. Detailed description of a small working prototype can be found in [Poulton et al, 1985]. Extensions of the linear tree to evaluate full six-coefficient quadratic expressions is described in [Goldfeather and Fuchs, 1986]; algorithms to use the quadratic extensions to render solid models defined by Constructive Solid Geometry are described in [Goldfeather, Hultquist, and Fuchs, 1986]. [Poulton et al, 1987] describes the design and construction of our latest syste, a 512 x 512-pixel prototype.

How It Works. The overall Pixel-planes system contains a fairly conventional graphics pipeline that traverses a hierarchical display list, computes viewing transformations, performs lighting calculations, clips polygons (or other primitives) that are not visible, and performs perspective division. The resulting colored polygon vertices in screen coordinates are then 'translated' into the form of data (A,B,C) for linear expressions together with instructions for the 'smart' frame buffer. An Image Generation Controller converts word-parallel A,B,C+Instructions to the bit-serial form required by the enhanced memory chips. A video controller scans out video data and refreshes a standard raster display (see Figure 2). In our prototype system, the graphics pipeline and the translator have been implemented on a one-board general purpose processor from off-the-shelf components.

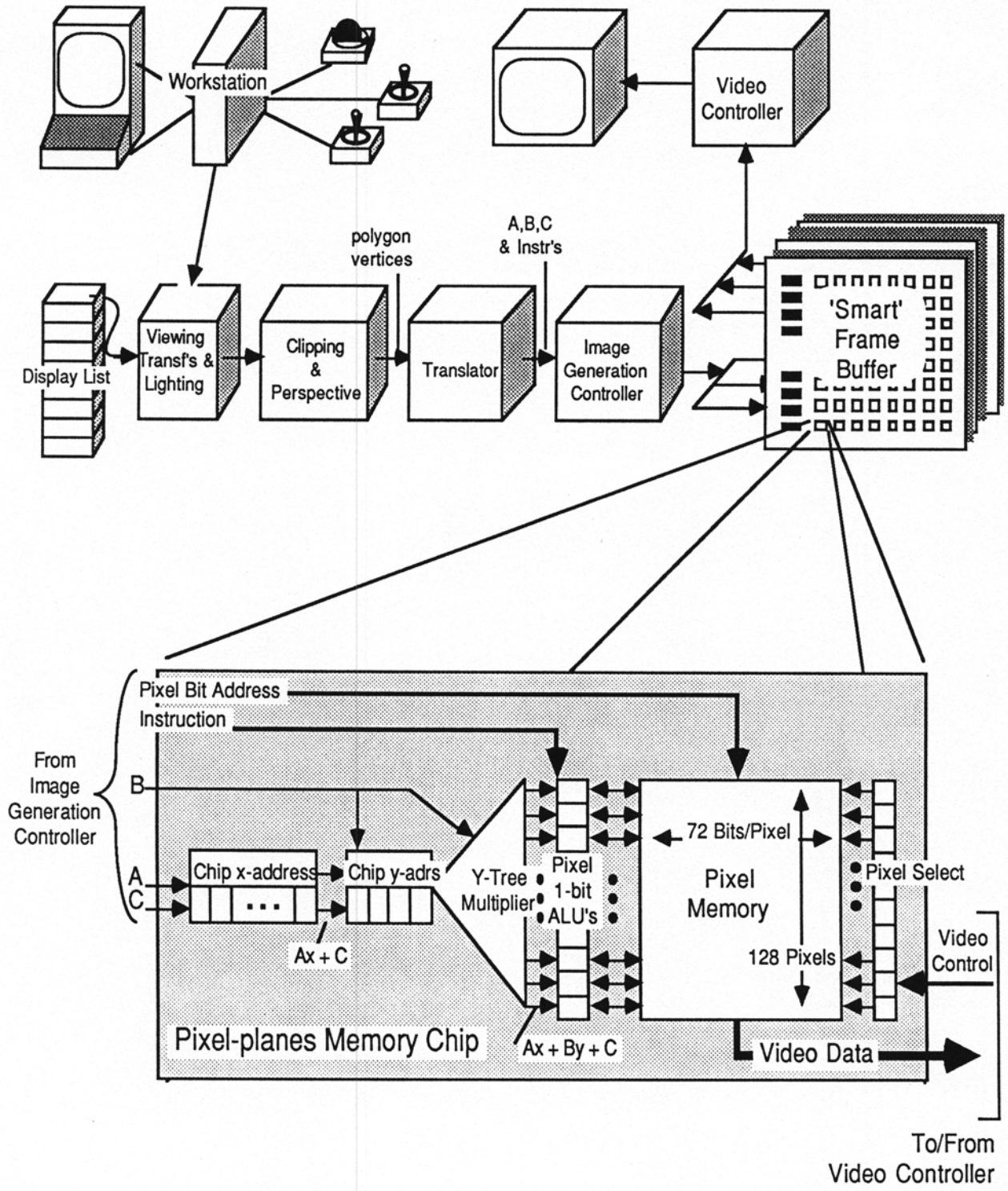

Figure 2: Pixel-planes Functional Organization

The heart of the system is the 'smart' frame buffer, an array of custom, VLSI, processor-enhanced memory chips. These chips have three main parts: a conventional dynamic memory array that stores all pixel data for two 64-pixel columns on the screen, 128 tiny one-bit ALU's, and a multiplier/accumulator (M/A) that generates linear expressions simultaneously for all pixels. All ALU's in the system execute the

same micro-instruction at the same time (**S**ingle-**I**nstruction-**M**ultiple-**D**ata fashion), and all memories receive the same address (each pixel ALU operates on its corresponding bit of data) at the same time. The multiplier provides the power of two 10-bit M/A's at every pixel, but at much less expense in silicon area on the chip. The part of the M/A that is common to the pixels in a single column is factored out (10 stages of X-multiplier and the first 4 stages of the Y-multiplier). The six stages of the Y-multiplier can be built as a binary tree, since Y-products for a column are closely related, thereby reducing the cost in silicon area to about 1.2 bit-serial M/A stages per pixel for the entire linear expression evaluator. The current prototype chips have 70% of their area devoted to memory, 30% to processing circuits; each chip contains two identical 64-pixel modules like the one shown in the figure. (A U.S. patent , No. 4,590,465, has recently been granted for the basic design of the system; another is pending.)

Work continues on this system to implement solid modeling by near real-time rendering directly from the CSG tree. This work extends that described in [Goldfeather, Hultquist, and Fuchs, 1986]. Transparent polygons by pseudo-random screens has also recently been implemented, using a variant of the method described in [Fuchs et al, 1985]. Textured surfaces as grids of tiny polygons has recently (June 1987) been demonstated on the system. We hope to include in the next generation hardware the capability to evaluate quadratic expressions directly, as described in [Goldfeather and Fuchs, 1986] and to speed up rendering with the capability to render simultaneously multiple primitives at different parts of the screen.

Summary

The continuing decline of DRAM prices and the steady increase in capabilities that can be squeezed into a single chip are fueling an explosion of interest and activity in graphics-specific chips among the chip design houses around the world. Most suppliers are aiming for general capabilities and low chip count in order to reduce parts cost to increase their market size. A small, but increasing number of suppliers (Silicon Graphics, General Electric, HP) are focusing on high-performance 3D graphics workstations. Encouraging results continue to be reported from the research community, where working prototypes using custom chips are starting to appear.

Acknowledgments

An earlier version of the survey section of this paper was presented at the 1986 International Summer Institute, "State of the Art in Computer Graphics," at the University of Stirling, Scotland. The author is grateful for assistance with this section from Dr. Rae Earnshaw of Leeds Univesity, Mr. Julian Ball of Raven Computers, Bradford, Yorkshire, England, and Dr. Melanie Mintzer. The author also thanks long-time colleague Professor John Poulton for the diagram on which Figure 2 is based.

This research was supported in part by the (U.S.) Defense Advanced Research Projects Agency (monitored by the U.S. Army Research Office, Research Triangle Park, North Carolina) under Contract DAAG29-83-K-0148, by the National Institutes of Health under Grant R01-CA39060, and by the National Science Foundation under Grants ECS-8300970 and DCI-8601552.

References

Abram, G.D. and H. Fuchs, "VLSI Architectures for Computer Graphics," *Proceedings of the NATO Advanced Study Institute on Microarchitecture for VLSI Computers*, eds. P. Antognetti, F. Anceau, and J. Vuillemin, Urbino, Italy, Springer-Verlag, July 1984, pp. 189-205.

Asal, M., G. Short, T. Preston, R. Simpson, D. Roskell, and K. Guttag, "The Texas Instruments 34010 Graphics System Processor," *IEEE Computer Graphics and Applications*, Vol. 6, No. 10, October 1986, pp. 24-39.

Bechtolsheim, A. and F. Baskett, "High-Performance Raster Graphics for Microcomputer Systems,"

Computer Graphics, Vol. 14, No. 3 (Proceedings of 1980 SIGGRAPH Conference), July 1980, pp. 43-47.

Carinalli, C. and J. Blair, "National's Advanced Graphics Chip Set for High-Performance Graphics," *IEEE Computer Graphics and Applications*, Vol. 6, No. 10, October 1986, pp. 40-48.

Clark, J., "A VLSI Geometry Processor for Graphics," *IEEE Computer*, Vol. 13, No. 7, July 1980, pp. 59-68.

Clark, J. "The Geometry Engine: A VLSI Geometry System for Graphics," *Computer Graphics*, Vol. 16, No. 3 (Proceedings of 1982 SIGGRAPH Conference), July 1982, pp. 127-133.

Clark, J. and M. Hannah, "Distributed Processing in a High-Performance Smart Image Memory," *Lambda* (since 1981, called *VLSI Design)*, Vol. 1, No. 4, 4th Quarter, 1980, pp.40-45.

Cohen, D. and S. Demetrescu, Presentation at 1980 SIGGRAPH Conference Panel on Trends on High Performance Graphics Systems.

Demetrescu, S., "A VLSI-Based Real-Time Hidden-Surface Elimination Display System," Master's Thesis, Department of Computer Science, California Institute of Technology, 1980.

Demetrescu, S., "High Speed Image Rasterization Using Scan Line Access Memories," *Proceedings of the 1985 Chapel Hill Conference on Very Large Scale Integration*, ed. Henry Fuchs, Computer Science Press, May 1985, pp. 221-244.

Demetrescu, S., Personal Communication, June 1986.

Dill, John and Jack Grimes, editors, *IEEE Computer Graphics and Applications*, Vol. 6, No. 10 (October 1986).

Electronics, "Intel Designs a Graphics Chip for both CAD and Business Use," *Electronics*, May 19, 1986, pp. 57-60.

Electronics, "NCR Aims its Graphics Chips at PC instead of Work Stations," *Electronics*, May 19, 1986, pp.61-63.

England, N., "Advanced Architectures for Graphics and Image Processing," *Proceedings of the IEEE*, Vol. 301, August 1981, pp. 54-57.

Evans and Sutherland Computer Corporation, "Line Drawing System Model I System Reference Manual," Evans and Sutherland Computer Corporation, P.O. Box 8700, Salt Lake City, Utah, 84108, 1971.

Everett, R.R., "The Whirlwind I Computer, " Joint AIEE-IRE Conference, 1952. *Review of Electronic Digital Computers,* February 1952, p. 70 (as cited in Newman and Sproull, *Principles of Interactive Computer Graphics,* New York, McGraw-Hill, 1973 [first edition], pp. xxi-xxii).

Fuchs, H. "Distributing a Visible Surface Algorithm Over Multiple Processors," *Proceedings of 1977 ACM Annual Conference*, October 1977, pp. 449-451.

Fuchs, Henry, "VLSI-Intensive Graphics Systems," *Proceedings of the NATO Advanced Study Institute on Mathematics and Computer Science in Medical Imaging*, Il Ciocco, Italy, September 1986, Springer-Verlag, 1987, to appear.

Fuchs, H., J. Goldfeather, J. Hultquist, S. Spach, J. Austin, F. Brooks, J. Eyles, and J. Poulton, "Fast Spheres, Shadows, Textures, Transparencies, and Image Enhancements in Pixel-planes," *Computer Graphics*, Vol. 19, No. 3 (Proceedings of 1985 SIGGRAPH Conference), July 1985, pp.

111-120.

Fuchs, H. and B. Johnson, "An Expandable Multiprocessor Architecture for Video Graphics," *Proceedings of the 6th Annual Symposium on Computer Architecture*, ACM-IEEE, New York, April 1979, pp. 58-67.

Fuchs, H. and J. Poulton, "PIXEL-PLANES: A VLSI- Oriented Design for a Raster Graphics Engine," *VLSI Design*, Vol. 2, No. 3, 3rd Quarter 1981, pp. 20-28.

Fuchs, H., J. Poulton, A. Paeth, and A. Bell, "Developing Pixel-Planes, A Smart Memory-Based Raster Graphics System," *Proceedings of the Conference On Advanced Research in VLSI*, Massachussetts Institute of Technology, ed. Paul Penfield, Jr., published by Artech House, Dedham, Mass., January 1982, pp. 137-146.

General Electric Graphicon 700 Product Literature, General Electric Silicon Systems Technologies Department, 4020 Stirrup Creek Dr., Research Triangle Park, NC 27703

Gharachorloo, N., Personal Communication, May 1987.

Gharachorloo, N., and C. Pottle, "Super Buffer: A Systolic VLSI Graphics Engine for Real Time Raster Image Generation," *Proceedings of the I985 Chapel Hill Conference on Very Large Scale Integration* , ed. Henry Fuchs, Computer Science Press, May I985, pp. 285-306.

Glassner, A. and H. Fuchs, "Hardware Enhancements for Raster Graphics," *Proceedings of the 1985 Advanced Study Institute on Fundamental Algorithms in Computer Graphics*, Ilkley, Yorkshire, England, ed. R. A. Earnshaw, Springer-Verlag, April 1985, pp. 631-658.

Goldfeather, J. and H. Fuchs, "Quadratic Surface Rendering on a Logic-Enhanced Frame-Buffer Memory System," *IEEE Computer Graphics and Applications*, Vol. 6, No. 1, January 1986, pp. 48-59.

Goldfeather, J., J. Hultquist, and H. Fuchs, "Fast Constructive Solid Geometry Display in the Pixel-Powers Graphics System," *Computer Graphics*, Vol. 20, No. 4 (Proceedings of 1986 SIGGRAPH Conference, eds. David C. Evans & Russell J. Athay), August 1986, pp. 107-116.

Gupta, S. and R. Sproull, and I.E. Sutherland "A VLSI Architecture for Updating Raster-Scan Displays," *Computer Graphics*, Vol. 15, No. 3 (Proceedings of 1981 SIGGRAPH Conference), August 1981, pp. 71-78.

Guttag, K., J. Van Aken, and M. Asal, "Requirements for a VLSI Graphics Processor," *IEEE Computer Graphics and Applications*, Vol. 6, No. 1, January 1986, pp. 32-47.

Ikedo, T. "High-Speed Techniques for a 3-D Color Graphics Terminal," *IEEE Computer Graphics and Applications*, Vol. 4, No. 5, May 1984, pp. 46-58.

Kajiya, J. T., I.E. Sutherland, and E.C. Cheadle, "A Random-Access Video Frame Buffer," *Proceedings of the IEEE Conference on Computer Graphics, Pattern Recognition, and Data Structure*, New York: IEEE, 1975, pp. 1-6.

Kay, A., "Microelectronics and the Personal Computer," *Scientific American*, Vol. 237, No. 3, September 1977, pp. 230-244.

Kedem, G. and J. Ellis, "Computer Structures for Curve-Solid Classification in Geometric Modelling," Technical Report TR137, Department of Computer Science, University of Rochester, May 1984.

Kunii, T., Personal Communication, 1986.

Lineback, J.R., "The Scramble to Win in Graphics Chips," *Electronics* , May 19, 1986, pp. 64-65.

Myer, T.H. and I.E. Sutherland, "On the Design of Display Processors," *Communications of the ACM,* Vol. 11, No. 6, June 1968, pp. 410-414.

Parke, F., "Simulation and Expected Performance Analysis of Multiple Processor Z-Buffer Systems," *Computer Graphics*, Vol. 14, No. 3 (Proceedings of 1980 SIGGRAPH Conference), July 1980, pp. 48-56.

Parks, J.K., "A Comparison of Two Graphics Computer Designs," M.S. Thesis, Computer Science Department, University of North Carolina at Chapel Hill, C.S. Tech Report TR82-001, 1982.

Pinkham, R., M. Novak, and K.Guttag, "Video RAM Excels At Fast Graphics," *Electronic Design*, Vol. 31, No. 17, August 18, 1983, pp. 161-172.

Poulton, J. and H. Fuchs, J. Austin, J. Eyles, J. Heinecke, C.-H. Hsieh, J. Goldfeather, J. Hultquist, and S. Spach, "Implementing a Full-Scale Pixel-planes System," *Proceedings of the 1985 Chapel Hill Conference on VLSI*, ed. Henry Fuchs, Computer Science Press, Rockville, Md., pp. 35-60.

Poulton, John, Henry Fuchs, John Austin, John Eyles, Trey Greer, "Building a 512 x 512 Pixel-planes System," *Advanced Research in VLSI: Proceedings of the 1987 Stanford Conference*, Paul Losleben, editor, MIT Press, Cambridge, MA 02142 (March 1987) pp. 57-71.

Schachter, B., "Computer Image Generation for Flight Simulation," *IEEE Computer Graphics and Applications*, Vol. 1, No. 4, 1981, pp. 29-68.

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

Shires, G., "A New VLSI Graphics Coprocessor--The Intel 82786," *IEEE Computer Graphics and Applications*, Vol. 6, No. 10, October 1986, pp. 49-55.

Sproull, R.F. and I.E. Sutherland, "A Clipping Divider," *Proceedings of the Fall Joint Computer Conference*, Thompson Books, Washington, D.C., 1968, pp. 765-775.

Sproull, R.F., I.E. Sutherland, A. Thompson and C. Minter, "The 8 by 8 Graphics Display," *ACM Transactions on Graphics,* Vol. 2, No. 1, January 1983, pp. 32-56.

Sutherland, I.E., "The Ultimate Display," *Proceedings of the IFIP Congress 65: Information Processing*, Vol. 2, ed. Wayne A. Kalenick, Spartan Books, 1966.

Sutherland, I.E., "A Head-Mounted Display," *Proceedings of the Fall Joint Computer Conference*, Thompson Books, Washington, D.C., 1968, pp. 757-764.

Sutherland, I.E. and G.W. Hodgman, "Reentrant Polygon Clipping, *Communications of the ACM*, Vol. 17, No. 1, January 1974, pp.32-42.

Swanson, R.W. and L.J. Thayer, "A Fast Shaded-Polygon Renderer," *Computer Graphics* , Vol. 20, No. 4 (Proceedings of the 1986 SIGGRAPH Conference, eds. David C. Evans & Russell J. Athay), August 1986, pp. 95-101.

Thacker, C.P., E.M. McCreight, B.W. Lampson, R.F. Sproull, and D.R. Boggs, "ALTO: A Personal Computer," Xerox Corp., 1979, in Siewiorek, Daniel P., C. Gordon Bell, and Allen Newell, *Computer Structures: Principles and Examples*, New York: McGraw-Hill, 1982, pp. 549- 572.

Watkins, G.S., "A Real-Time Visible Surface Algorithm," Ph.D. dissertation, University of Utah Computer Science, UTEC-CSc-70-101, June 1970, NTIS AD-762 004.

Weinberg, R., "Parallel Processing Image Synthesis and Anti-Aliasing," *Computer Graphics*, Vol. 15, No. 3 (Proceedings of 1981 SIGGRAPH Conference), August 1981, pp. 55-61.

Weitek Corporation, "Preliminary Data Documentation, Board Level Graphics Processors and Scientific Processors," Weitek Corporation, 1060 East Arques, Sunnyvale, California 94086, l986.

Whelan, D., "A Rectangular Area Filling Display System Architecture," *Computer Graphics*, Vol. 16, No. 3 (Proceedings of 1982 SIGGRAPH Conference), July 1982, pp. 147-153.

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

Wientjes, B., K. Guttag, and D. Roskell, "First Graphics Processor takes Complex Orders to Run Bit-Mapped Displays," *Electronic Design*, January 23, l986, pp. 73-81.

Williamson, R. and P. Rickert, "Dedicated Processor Shrinks Graphics Systems to Three Chips," *Electronic Design*, Vol. 31, No. 16, August 4, 1983, pp. 143-148.