SPECIALIZED COMPUTER ORGANIZATION FOR RASTER GRAPHICS DISPLAY

Henry Fuchs
Gregory D. Abram

Computer Science Department
University of North Carolina at Chapel Hill

The generation of images by computer has been both an interesting and a computationally expensive task for over thirty years [7]. The availability of LSI building block devices have enabled graphic system designers to build special-purpose systems for various classes of applications from "glass teletypes" to drafting tools to real-time 3D scene generators. Recent emergence of custom VLSI design capabilities are stimulating developments of even more radical solutions. We review the organization of a number of existing systems, describe several experimental designs, and outline some remaining problems and suggest some possible solutions.

I. INTRODUCTION

A. Motivation

1. The Use of Images As increasing processing power becomes available, a successively larger share of the overall problem of computer usage becomes not how fast the computer can calculate the task, but how easily the user can specify the input and comprehend and evaluate the results. The visual medium is perhaps the most effective interface.

2. Current Problems Many applications of computer-generated images call for user *interaction* with the image, so generating the image within a fraction of a second is often crucial for the effectiveness of the application. For example, a crystallographer interactively fits a residue chain inside its 3-dimensional volume, and pilot trainees practice landing on simulated runways under adverse conditions. Unfortunately, graphics systems that serve these needs are often

quite expensive, decreasing the cost/performance of such systems remains a major topic of study in computer graphics.

B. Problem Statement

The goal of these systems is to generate images in a small fraction of a second. These images may consist of 20,000 or more line segments or several thousand polygons. (A longer range goal is for some or all of these polygons to be curved patches, but this seems beyond attainment in the next few years.) For many applications such images should be generated in 1/30 of a second. The image generation consists of two major phases: transforming the database (geometric operations on the data such as rotations, transformations, scaling and clipping) and painting the resulting lines and polygons. Since these geometric operations are done adequately using existing hardware[1], further discussion will focus on systems whose goals is to paint the lines and polygons quickly.

C. Taxonomy of Solutions

Long term interest in this problem, coupled with rapidly developing technology, has inspired much recent activity. In this paper we review of some of this work and outline certain shared principles that may help to predict future trends. Although the work we shall review is principally "polygon based" we shall use the lessons learned there to outline a possible approach to fast line generation on raster display, a topic which has not received much attention in the technical lierature.

Before embarking on a review of individual systems, we outline here the various general approaches to the problem.

The major problem in image generation is calculating the the 750,000-3,000,000 individual pixel values 30 times a second. Since no current single processor can do this, the prolem becomes finding ways to partition this task among a great many different processing elements. Several approaches to this partitioniing task have been and are continuing to be explored.

1. Dividing the algorithm The traditional method pof speeding up image generation has been to partition the overall algorithm among a number of special purpose processing units. For example, moderately priced systems often include specialized geometric transformation hardware, to handle this process in parallel with other tasks. Flight simulators carry this notion much further with additional special hardware, often connected together in a pipeline.

2. Dividing the input or output data Either subsets of the input data or the output pixels can be partitioned over may processing elements.

Distinctly different solutions are possible when the system is restricted to standard components than if custom ICs are allowed.

---

[1] Clark, in [4], describes a design for a VLSI-based transformation and clipping unit that makes use of several identical, pipelined floating point processor chips to provide these functions at lower cost than the current board-level subsystems.

a. Standard components The most promising solutions appear to be based on partitioning the output image among multiple units. The various solutions organize the memory in different ways in appempting to balance the image generation load among the many processors without interfering with the video-san refresh.

b. Custom ICs The flexibility of custom ICs allows more radical solutions. Two promising approaches are based on 1) assigning a processor per input datum, and 2) designing image buffer emory chips themselves with more smarts so higher than pixel-level read/write commands can be executed on them.

II. POLYGON-BASED IMAGE GENERATION

A. Introduction

Rendered images composed of thousands of polygons are among the most realistic images that can currently be generated by computer and their realism makes them useful for many applications. (Strictly speaking, the most realistic images may be those that also allow curved surface patches in addition to planar tiles; display of these images, however, is so time consuming that they have not been feasible for applications needing fast display.) Real-time systems are extremely expensive; other systems can either generate very limited images in interactive time -- several images per second, the rate necessary for effective manipulation of an object by joystick or sliders -- or they take several minutes or more to generate images of quite complex scenes.

B. Current Systems

1. Flight Simulators

a. Definition and historical background Since the early 1960's, the driving application in real-time rendered image systems has been to generate out-the-window views of flight simulators for pilot training. These systems typically cost over a million dollars due to the very high performance requirements [17].

b. Typical organization These systems have enormous computational burdens, with tens of thousands of polygons in the world data base of which several thousand may be visible at any one time. They need to calculate lighting effects that may include transparecy, textured surfaces, fog, and shadows. Not even the most powerful general-purpose computer is likely to be able to handle this task. Even in a moderate resolution image of 512x512 pixels, there are 250,000 pixels to calculate 30 times a second, each with red, green and blue components; 22 million values must be calculated in a second, or about 50 nanoseconds per value.

In general, such performance has been accomplished by dividing the processing into a pipeline of 5-10 different kinds of special-purpose units, such as geometric processors, sorting processors, polygon extent calculators, and painting units. In some situations, a stage of the pipeline consists of several copies of the same unit operating in parallel in order to keep the pipeline running at the required rate. These systems are reasonably large; Schachter reports that a recent GE system contains 1.4x10**5 ICs. [16] Another system, a middle range Evans and Sutherland CT-5 with 6 channels and 500,000 pixels per channel, uses 80,000 chips in ten racks, of which 10,000 are 16K RAM chips [18].

The database is often hand-tuned in order to minimize processing time and reduce the possibility of bottlenecks. When bottlenecks do occur, rather elaborate recovery procedures must be used to recover gracefully, for example, automaticallty substituting less complex descriptions of certain objects in the scene.

2. Image Buffer-based Systems The other extreme in image generation hardware are the ubiquitous "frame buffer"-based displays. These consist fundamentally of sufficient solid-state memory to store an image from which the video screen is constantly refreshed, and into which image values are stored as they are calculated by a general-purpose computer. These systems, with retail cost beween $5,000 and $100,000, may include microprocessors, transformation units, color maps, and other devices to speed the image generation process. Because these systems are programmable, virtually any algorithm can be implemented on them, but executing slowly, taking seconds or minutes to generate each image.

A few systems have recently been introduced (by Megatek and Ikonas, for example) with some special hardware features to speed up the polygon (and line) generation task by parallel writing of consecutive pixels.

C. Experimental VLSI-Oriented Systems

1. Introduction: VLSI, Regularity and Parallel Processing In the last few years, there has been increasing interest in bringing to bear the capabilities of VLSI to solve this computationally massive task. The task seems a natural candidate for solution by VLSI systems; the problem has great regularity and uniformity and there is a large potential market for low cost systems. Regretably our review here does not include experimental systems of commercial vendors, as details of these syetems are closely guarded secrets of the individual companies.

Published research efforts fall into two categories: using parallelism in the input, and using parallelism in the output. Specifically, the input data can be distributed many processors by assigning a processor for each polygon in the input data, or alternately, the screen can be divided among the many processors.

2. Distributing input data among multiple processors Cohen and Demetrescu, in [5], have proposed a system that assigns a processor to each potentially visible polygon in the image space (i.e., already transformed world model polygon). These, processors are connected as a pipeline and are operated in sync with the video generation. For each pixel on the screen, a token passes through the

pipeline which carries the shade and depth of the closest point found so far. This depth is the distance from the viewing position of the closest polygon so far; thus the shade is the best guess so far of the color seen at this pixel. Each processor in turn tests whether the pixel lies inside its polygon. If the point lies inside, the processor compares this depth with its polygon's depth at this point. If the polygon's depth is closer, its depth and color replace the token's data. Tokens pass in raster-scan order and travel at video rates; that is, each processor must make each decision in one pixel time.

An elegant feature of this approach is that the pixels stream out of the end of the pipeline in raster-scan order and each value represents the color of the absolute nearest polygon at that pixel; thus the data can be routed directly onto a video display screen.

Weinberg, in [22], proposes an elaboration on this design which addresses the problem of anti-aliasing by passing multiple depth-sorted tokens for each pixel along with subpixel masks. Each processor then determines the portion of the pixel covered by its polygon, and compares with the token's mask when the correct position in the depth order is found. If preceding polygons do not completely obscure it, it is added to the token chain. Subsequent tokens are then examined to see if the new polygon completely obscures them, deleting those that are. A filter section at the output uses this data to determine an output shade for each pixel.

This general approach features great modularity; it consists of identical processors hung together in a simple pipeline. It is easily expandible by simply adding more processor chips. The design costs are held down by the fact that only a single IC needs to be developed; manufacturing costs are held down by the simple structure. The only difficulties may be 1) implementing enough processors so that there is one for each and every polygon in the most complex scene in the intended application and 2) making each processors sufficiently fast to complete all its calculations for a pixel in one pixel time.

3. Output Parallel Systems Several proposals have been made for restructuring the frame buffer memory in ways to facilitate the division of the screen among many processors. These fall in two classes: 1) systems that use special configurations of standard memory chips to allow parallel access from many processors, and 2) a smart image memory system which integrates the memory and processing elements in each pixel.

a. Standard memory systems Fuchs proposed in [8] and Fuchs and Johnson elaborated in [9], a system which distributes the pixels in an interlaced fashion in both the x and y directions so that if there are MxN processors in the system, each processor is assigned every Mth pixel on every Nth scan line. The polygons are processed sequentially and are broadcast to all the processors simultaneously. Since a polygon covers a contiguous region of the image, it is guaranteed to be distributed among many processors, with each processor responsible for only a small fraction of any polygon's pixels.

Parke, in [15] points out that in a distributed system such as the one above all the processors must perform the polygon startup overhead calculations for every polygon. He proposes an alternate design in which the processors are assigned fixed contiguous areas of the screen, and are arranged at the leaves of a binary tree that has a polygon splitter at each non-terminal node. Polygons in sequence enter at the root of the tree and trickle down to the appropriate processors, depending on their location in the image. Although this organization

minimizes the number of processors which must deal with a polygon, it is extremely sensitive to the distribution of the polygons in the image; processors responsible for regions with many polygons become clogged while processors in other areas of the image lie idle. Parks, in [16] compares the interlaced and contiguous area approaches and reports expected execution times for several "real" 3D scene descriptions.

Clark and Hanna, in [3], describe a 64 processor interlace design in an 8x8 pattern similar to the design in [8] and [9]. This system has an additional layer of 8 interlaced column processors that shields the 64 lower level processors from the polygon startup calculations.

b. Smart memory systems The Pixel-Planes system, as described by Fuchs and Poulton in [10] and by Fuchs, Poulton, Paeth and Bell in [11], differs from the above systems in that it incorporates the actual image buffer memories in the design rather than using standard large RAM chips. This allows the system to calculate concurrently values for every pixel in the image. By having contiguous pixels within a single chip, the system allows a tree of processing elements to perform each calculation at the highest possible level, thereby achieving the effect of a processor at every pixel, with only a small fraction of an actual processor at each pixel.

D. Remaining Common Problems

There are advanced tasks beyond the basic scan conversion - visibility - rendering discussed above which the commercial systems, forced by market pressure, have been forced to overcome. Advanced experimental designs of commercial manufacturers are closely guarded secrets and thus, unfortunately, cannot be reviewed here. The experimental systems, for the most part, have not been described in the literature in sufficient detail to enable evaluation of their ability to handle advanced tasks such as anti-aliasing, complex lighting and textured surfaces. In addition, not even the most advanced commercial systems can yet handle true curved surface objects (the curved surfaces are approximated by planar tiles).

III. LINE-GENERATING SYSTEMS

We now turn to the topic of line-drawings on raster displays. Since, traditionally, lines are done on random stroke displays, there is relatively little literature on fast line drawing on raster devices; further, since drawing lines on raster displays is a special case of drawing polygons, we will discuss how the ideas from polygon-based systems can be used for this special application.

Line graphics, the earliest approach to computer-generated graphical images, remains a very important part of the industry. Used in a wide range of applications, line-drawn graphics can easily be handled both by the user and by the graphics system designer. It presently is the only choice for real-time interactive graphics for most users.

A. Problems With Current Solutions (Stroke Graphics)

In the past, high performance line-drawing displays have used random-scan technology to create images. This approach has several benefits: such systems can draw a large number of very high-quality lines in real-time. With appropriately fast transformation hardware, this approach can allow rotation, translation and scaling in real-time.

Random-scan approaches also have several problems. The random-scan monitor itself is quite expensive (often in the range of $5,000 - $10,000), particularly in comparison to raster scan video monitors, which which takes advantage of lessons learned from the enormous television industry. The cost of these devices is most apparent in color systems; where a full color stroke monitor may cost $25,000 or more, color video displays are often $5000 or less.

Random-scan also suffers because the display has to be refreshed at 30 Hz. If more lines are specified than can be displayed in 1/30 of a second, either disturbing flicker or the deletion of lines will occur. Raster-scan approaches, however, are guaranteed to refresh the entire screen at rates that prevent flicker, so that theupdate rate (the rate at which a new frame is prepared) can be independent of the refresh rate. Thus raster scan devices can display arbitrarily complex line drawn images, while random-scan systems have an inherant limit.

C. Problems With Raster

Raster devices have their own set of problems, which must be addressed if they are to replace random-scan devices.

1. **Memory Bandwidth** A major drawback is the rate at which current systems can draw lines. In raster technology, each pixel along a line must in some way be specified and the appropriate shade stored into the frame buffer. Most frame buffer systems generate lines at about 1 microsecond per pixel. Even with special purpose hardware enhancements, they run no faster (today) than160 nanoseconds per pixel. Assuming a 1024x1024 pixel image, this rate is almost an order of magnitude slower than current stroke systems (12,500 1" lines to 95,000 1/10" lines at 30 Hz.).

2. **Resolution** Raster-scan systems also suffer from their low resolution, which causes the phenomena of staircasing, the rastering effect caused by a low sampling rate of the actual screen points along the true line. This problem has two solutions, both of which are costly. The display can be calculated at higher resolution (note that the number of points to determine along the line rises as a function of both the length and the resolution). Alternatively, the lines can be anti-aliased; for this the pixel values have to be precisely determined from the proportion of the pixel covered by the line. It is harder to determine how much of a pixel is covered than simply to determine that the line touches it, so this process again costs either time or money.

D. Solution: A Sketch

The lessons learned in the designs for polygonal systems may also be applied in designing a line-oriented raster system.

1. Organization For fast line generation, several processors must be used to share the image generation task. Since the number of lines displayed by a typical line-generating system is much higher than the number of polygons processed by even the most sophisticated polygon system, a processor-per-input approach is impractical. We are therefore left with an image-area subdivision approach. Vertical interlacing of processors shares the scan-conversion of most lines among many processors. Horizontal interlacing of the pixels along the scan-line alows parallel access to several adjacent pixels along the scan-line, covering the intersection of the true line with the scan-line (the "span") in fewer memory cycles.

In this organization each processor can compute the extent ("span") of the line along each of the scan lines under its control. The pixels in these spans can then filled quickly by taking advantage of the parallel access to adjacent pixels.

2. Anti-Aliasing This organization expands simply to higher resolution displays without image generation time cost. Although doubling the resolution of the display quadruples the number of memory chips required, we need only double the number of processors. The increased number of memory chips allows us to double the number of pixels in a scan-line that can be accessed in parallel to fill a span. Therefore, although twice as many scan lines intersect the desired line, twice as many processors are computing spans. Although spans may be twice as long, twice as many pixels can be stored in parallel. Thus, doubling the resolution should not significantly affect the line drawing rate.

IV. CONCLUSIONS AND SUMMARY

The most promising way to speed up the image generation task is to distribute the task among a great many processing units. The two ways that this has been done are 1) distributing the input among the units, and 2) distributing the image to be calculated among many processors. The major distinction for any of these solutions is the degree to which custom-designed VLSI components can be used. For systems restricted to using standard components, the most promising approach appears to be to distribute the output image among the processors, since no designs have been demonstrated which effectively distribute the input among standard components. The bottleneck of such systems is the large-scale RAM, being able to change only a single value in each chip in one memory cycle time.

If custom ICs are allowed, this bottleneck can be overcome with custom IC chips tht include some on-board processing circuitry at each pixel. Alternately, one could design small, special-purpose processors among which the input data can be efficiently distributed.

When the above problems are solved and techniques for inexpensive real-time generation of arbitrarily curved, textured and shadowed surfaces are developed, we shall be close to realizing Ivan Sutherland's 1965 vision of having an effective window into an imaginary world within everyone's reach [19].

Acknowledgements

We are grateful for assistance from Michael Pique of our department, Robert Schumacker of Evans and Sutherland Computer Corporation, Sang Lee and Jerry Kennedy of Advanced Electronic Design, Inc., and Mitchell Reis of Megatek Corporation.

References

[1] A. Bechtolsheim, and F. Baskett, "High-Performance Raster Graphics for Microcomputer Systems", *SIGGRAPH '80 Proceedings*, (July 1980).

[2] T. Blank, Mark Stefik, and Willem vanCleempu, "A Parallel Bit Map Processor Architecture for DA Algorithms", *ACM IEEE Eighteenth Design Automation Conference Proceedings*, (June 1981).

[3] J. Clark, and M. Hannah, "Distributed Processing in a High-Performance Smart Image Memory", *VLSI Design*, Vol.I, No. 3, 4rd. Quarter.s (1980).

[4] J. Clark, "The Geometry Engine: A VLSI Geometry System for Graphics", *SIGGRAPH '82 Proceedings*, (July 1982).

[5] D. Cohen and S. Demetrescu, Presentation at SIGGRAPH '80 Panel on Trends on High Performance Graphic Systems (1980).

[6] F. Crow, "A Frame Buffer System With Enhanced Functionality", SIGGRAPH '81 Proceedings, (August 1981).

[7] R.R. Everett, "The Whirlwind I Computer", *Review of Electronic Digital Computers*, *Joint AIEE-IRE Conference*, 70, (February 1952) (quoted in [Newman and Sproull, 1979]).

[8] H. Fuchs, "Distributing a Visible Surface Algorithm Over Multiple Processors", *Proceedings of the 1977 ACM Annual Conference* (1977).

[9] H. Fuchs and B.W. Johnson, "An Expanded Multiprocessor Architecture for Video Graphics", *Proc. of the 6th Annual Symposium on Computer Architecture*, ACM Inc., New York, NY (1979).

[10] H. Fuchs and J. Poulton, "PIXEL-PLANES: A VLSI-Oriented Design for a Raster Graphics Engine", *VLSI Design*, Vol.II, No. 3, 3rd. Quarter (1981).

[11] H. Fuchs, J. Poulton, A. Paeth and A. Bell, "Developing Pixel Planes, A Smart Memory-Based Raster Graphics System", *Proc. Conference On Advanced Research in VLSI*, Massachussetts Institute of Technology, Cambridge, MA., (January 1982).

[12] S. Gupta, and R. Sproull, A VLSI Architecture for Updating Raster-Scan Displays", *SIGGRAPH '81 Proceedings*, (August 1981).

[13] C. Mead and L. Conway, Introduction to VLSI Systems, Addison-Wesley.

[14] W.M. Newman, and R.F. Sproull, Principles of Interacive Computer Graphics, McGraw-Hill, 2nd edition (1979).

[15] F.I. Parke, "Simulation and Expected Performance Analysis of Multiple

Processor Z-Buffer Systems", *SIGGRAPH '80 Proceedings*, (July 1980).

[16] J.K. Parks, "A Comparison of Two Graphics Computer Designs", University of North Carolina at Chapel Hill Computer Science Department Tech. Report TR82-001 (1982).

[17] B. Schachter, "Computer Image Generation for Flight Simulation", *IEEE Computer Graphics and Applications*, Vol. 1, No. 4. (1981).

[18] R.A. Schumacker, personal communication (1982).

[19] I.E. Sutherland, I.E., "The Ultimate Display," summary in *Proceedings of the IFIP Congress 1965*, Vol. 2, pp. 506-509 (1965).

[20] I.E. Sutherland, R.F. Sproull, and R.A. Schumaker, "A Characterization of Ten Hidden Surface Algorithms", *ACM Computing Surveys*, Vol. 6, No. 1 (1974).

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

[22] R. Weinberg, "Parallel Processing Image Synthesis and Anti-Aliasing", *SIGGRAPH '81 Proceedings*, (August 1981).

[23] D. Whelan, "A Rectangular Area Filling Display System Architecture", *SIGGRAPH '82 Proceedings*, (July 1982).

[24] J.T. Whitted, "Hardware Enhanced 3-D Raster Display System", *Proceedings of the 7th Canadian Man-Computer Communications Conference*, Waterloo, (June 1981).