Coarse-Grain and Fine-Grain Parallelism in the Next Generation Pixel-Planes Graphics System

TR88-014

March 1988

Henry Fuchs, John Poulton
John Eyles, Trey Greer

The University of North Carolina at Chapel Hill
Department of Computer Science
CB#3175, Sitterson Hall
Chapel Hill, NC 27599-3175

To appear in Proceedings of the International Conference and Exhibition on Parallel Processing for Computer Vision and Display, University of Leeds, United Kingdom, January 12-15, 1988.

Coarse-Grain and Fine-Grain Parallelism in the Next Generation Pixel-planes Graphics System

Henry Fuchs, John Poulton, John Eyles, Trey Greer
Department of Computer Science
University of North Carolina
Chapel Hill, NC 27599-3175 USA

Abstract

In this paper we describe the current state of our graphics system, Pixel-planes 4 (Pxpl4), and the basic architecture that we are planning for its successor, Pixel-planes 5 (Pxpl5). At the time of its introduction (at Siggraph'86 Conference in August 1986), Pxpl4 was one of the fastest machines for near real-time rendering of 3D scenes. In a year of constant use, it has opened new research possibilities for us and for other local investigators, while also frustrating us and others with its limitations. This paper introduces the preliminary design of its successor, Pxpl5, which we expect to: 1) be some 20 times faster, 2) overcome many of Pxpl4's limitations by having a much wider range of capabilities and applications, 3) be realized in a desk-height workstation pedestal, and 4) be reproducible in a variety of configurations from a few to a few dozen boards. Both coarse-grain and fine-grain parallelism will be used to realize its expected performance and generality.

1. Introduction

1.1 Goals For Our Experimental Graphics Systems

A primary goal with Pixel-planes has been to provide more effective, near real-time visual interaction for several difficult 3D applications: 1) radiation therapy planning for cancerous tumors, 2) molecular modeling of complex proteins, 3) comprehension of building designs by architects and their clients. We work closely with colleagues specializing in these three applications to test the effectiveness of our graphics systems. We also make design decisions for our future systems based on their specific needs. Thus, although we hope to have graphics systems that are widely useful, we concentrate on specific applications in the belief that more successful systems will result from first satisfying a few specific needs, rather than working in isolation, trying to meet vaguely-perceived general needs and hoping that the resulting systems will somehow be useful for many applications. The most challenging goal from these three applications has been to generate images from 3D scene descriptions in real-time or as close to it as possible--certainly at several updates per second. We and our colleagues have found that the comprehension of complicated 3D structures (such as the relationship between a radiation isodose surface and the neighboring anatomy, or the shape of an active site in a protein) is drastically reduced whenever the update time for an object rotation or a cutting-plane move is reduced from, say 0.1 second to 1 second. We also find that comprehension is significantly increased whenever we can combine various 3D depth cues, for example, adding to dynamic object motion (kinetic depth effect) both stereo

and head-motion parallax. Unfortunately such capabilities require still greater graphics computation power. Thus, although our current system, Pxpl4, meets some of these needs more effectively than any other system available today, it is still insufficient for many of our applications' needs. It is largely these and other still unmet goals that inspire us to embark on the design and construction of another generation of Pixel-planes systems.

1.2 3D Graphics Fundamentals

Our systems, as well as most others, use a conventional graphics pipeline organization in which a display list of 3D objects is traversed by a graphics processor, each polygon (or other) primitive is transformed into the screen space, its parts outside the viewing frustum are clipped out, the proper colors at its vertices are calculated, and it is rendered into a frame buffer using a depth-buffer algorithm. (Pixel-planes is often programmed to perform other tasks, but the standard one sketched above is the most common.) Since it has been possible for many years to purchase systems that will generate, in real-time, wire-frame versions of quite complex objects, we (and many others) have been concentrating on building systems that solve the back-end bottleneck, the image rendering. For a review of some of these related systems, see [Fuchs, 1987]. A notable new system that had not appeared up to the time of that review and one that devotes dozens of custom processors to the back-end rendering is described in [Silicon Graphics, 1987] and in [Jermoluk and Akeley, 1988].

2. Current Pixel-planes System (Pxpl4)

2.1 System Overview

This section presents a short overview of Pxpl4; see [Eyles, 1987] for a more complete description. The heart of the system and its most unusual feature is its frame buffer, which is composed of custom logic-enhanced memory chips that can be programmed to perform most of the time-consuming pixel-oriented tasks at every pixel in parallel (See Figure I). 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/pixel, implemented with 2,048 custom 3-micron nMOS chips (63,000 transistors in each, operating at 8 million micro-instructions per second) [Poulton, 1987]. Algorithms for rendering spheres (for molecular modeling), for adding shadows, for enhancing medical images, and for rendering objects described by constructive solid geometry (CSG) directly from the CSG description have been devised by various individuals within and also outside our research group. The Pxpl4 system is in daily use in our department's Computer Graphics Laboratory, where applications in molecular modeling, medical imaging, and architecture are being developed.

Concept. The front part of the system (the part that performs geometric transformations, clipping and lighting calculations) specifies the objects on the screen to the frame buffer in geometric, pixel-independent terms, and the frame-buffer memory chips themselves work from this description to generate the final image. Image primitives such as lines, polygons, and spheres are each described by expressions (and operations) that are 'linear in screen space,' that is, by 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 that is broadcast to the frame buffer is a sequence of sets (A,B,C, instruction), rather than the usual (pixel-address, RGB-data) pairs.

How It Works. Pxpl4 contains a fairly conventional 'front end' graphics processor, implemented using the Weitek XL chip set, that traverses a segmented, hierarchical display list, computes viewing transformations, performs lighting calculations, clips polygons (or other primitives) that are not visible, and performs perspective division. It then translates the description of each object into the (linear coefficients and op-codes) form of data for the 'smart' frame buffer. An image generation controller converts the word-parallel data and instructions into the bit-serial form required by the enhanced memory chips. A video controller scans out video data from the frame buffer and refreshes a standard raster display. The system is hosted by a conventional UNIX workstation that supports the system's user interface through various graphics input devices and that provides system programming tools (e.g., graphics libraries, microcode assemblers, language compilers). During system initialization, the host downloads microcode and setup information to Pxpl4 via a service bus not shown in the figure. (The fundamentals of the system are covered by U.S. Patent No. 4,590,465, and another patent is pending.)

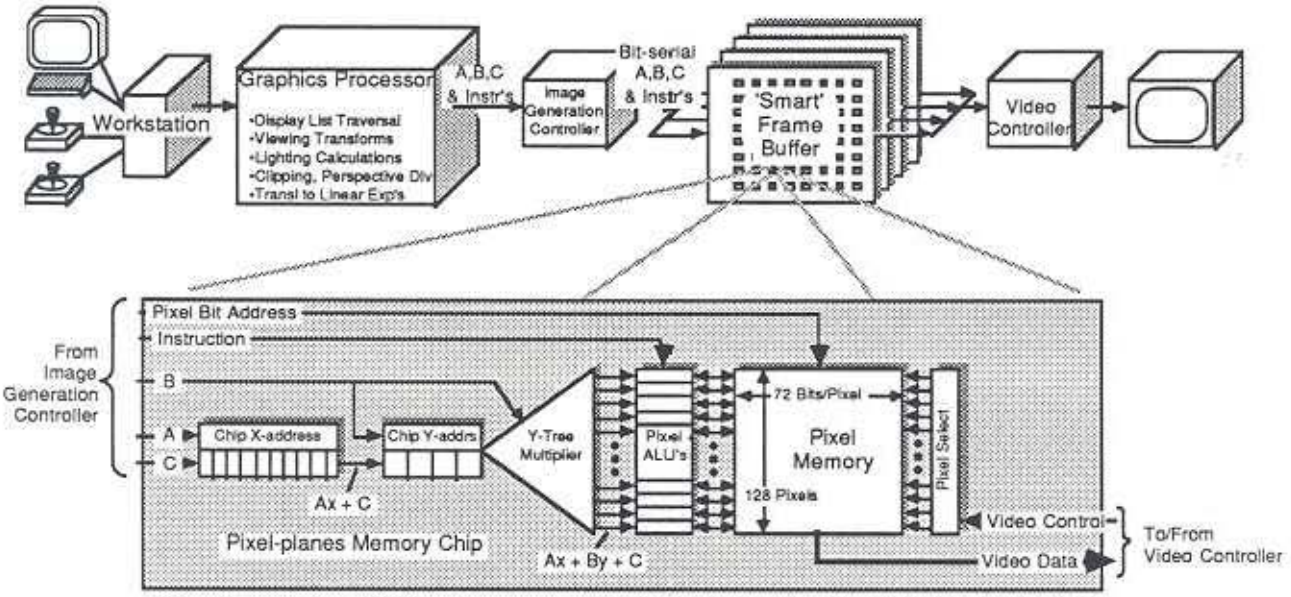

Figure 1: Pixel-planes 4 System Overview

Smart Memory Chips. The heart of the system is the logic-enhanced frame buffer, an array of custom, VLSI, processor-enhanced memory chips. Each of these chips contains two identical 64-pixel modules. Each module has three main parts: a conventional memory array that stores all pixel data for a 64-pixel column on the screen, an array of 64 tiny one-bit ALU's, and a linear expression evaluator (LEE) that generates Ax+By+C simultaneously for all pixels. All ALU's in the system execute the same micro-instruction at the same time (in Single-Instruction-Multiple-Data fashion), and all memories receive the

same address (each pixel ALU operates on its corresponding bit of data) at the same time. The LEE provides the power of two 10-bit multiplier/accumulators at every pixel, but at much less expense in silicon area on the chip. The part of the LEE that is common to the pixels in a single column is factored out (10 stages of the X-multiplier and the first 4 stages of the Y-multiplier). The last 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 multiplier/accumulator stages per pixel for the entire LEE. Pxpl4 chips have 70% of their active area devoted to memory and 30% to processing circuitry.

Hardware Configuration. The Pixel-planes 4 system consists of a Digital Equipment Corporation MicroVax II workstation, which acts as host, and a separate cabinet containing the prototype custom hardware. The prototype contains two card cages in a single rack: a multibus cage and a cage with a fully custom backplane. The custom cage contains the 512 x 512-pixel frame buffer on 32 15" x 15" boards, each with 64 logic-enhanced memory chips. The multibus cage contains all the other boards: a host interface board through which a DMA link to the host is realized, an analog input board to which joysticks and sliders are connected, the graphics processor (with 8 MBytes of RAM), the image generation controller; and the video controller.

Performance. Pxpl4 can process about 35,000 smooth-shaded, Z-buffered triangles per second (quadrilaterals are about 20% slower). Shadows are cast at about 11,000 triangles per second, using true shadow volumes. About 13,000 smooth-shaded, Z-buffered, interpenetrating spheres can be rendered per second. Virtually any number of updates per second can be realized, since the entire Z buffer and all the RBG buffers can be cleared in less than 10 microseconds.

2.2 Algorithms On Pixel-planes 4

New algorithms and applications have been appearing in a flurry since the system became operational in August 1986 and especially since the new, C-programmable graphics processor became operational in Spring 1987.

CSG System. Claire Durand, Steve Molnar and Greg Turk have implemented an interactive design system based on constructive solid geometry (CSG) and building on the methods described in [Goldfeather, Hultquist, and Fuchs, 1986] and in [Jansen, 1986]. This is the first system, to our knowledge, that allows direct-manipulation of the CSG tree via a Macintosh-style user interface and continuous rendering of the resulting solid object. Casual users have begun designing simple objects with the system: telephone sets and dining service and goblets. The design team has begun to interface their system with a CSG-to-polygons program (part of the BAGS package from Brown University developed by Professor Andries van Dam and students). The capability to deal with both CSG structure and polygons may allow the user-designer to enjoy both the ease of design within a CSG structure and the still much faster rendering of polygonal representation for objects that are no longer being changed.

Transparency. Two different algorithms have been implemented for transparency, one by David Ellsworth, the other by John Rhoades. Ellsworth's method displays transparent polygons as opaque polygons with randomly positioned single-pixel holes--the greater the polygon's transparency, the more holes. This method allows simple processing of polygons in an order, but results in distracting sparkling

of holes as the polygon moves from frame to frame. Ellsworth is working on reducing this effect. Rhoades' method processes transparent polygons only after all the opaque ones. This involves partitioning the display list and, for the method to work properly, involves sorting the transparent polygons from, for instance, back to front. Rhoades' current implementation does not sort the transparent polygons but still gives results that are acceptable to our users. He is working on more advanced implementations, especially ones that will run efficiently on Pxpl5.

Textures. Two different algorithms for rendering textures are being explored, one by Vicki Interrante, the other by Brice Tebbs. Interrante's version displays a textured surface as one created by a cover of many tiny polygons, one between every consecutive texture sample. It works fine, although slowly. We plan to speed up this version by defining a hierarchy of textures, each with a different number of polygons, using progressive refinement techniques. When a texture is moved rapidly, only a rough (blurred) version will be displayed. When it is stopped, a more refined version will automatically appear. Tebbs' version is an implementation of the techniques described by our team in [Fuchs, et al, 1985]. Pixels within a textured surface are "colored" with texture coordinates when they are originally rendered. After all the polygons have been rendered, the system broadcasts the texture(s) once; the pixels storing texture coordinates then replace these coordinates with the corresponding broadcast texture values. This method works well when there are only a few small textures used repeatedly throughout the image--like bricks and asphalt in a building scene. Still a problem is the situation in which a texture is compressed and multiple texture samples map onto the same pixel. The new partitioned parallelism of Pxpl5 and its graphics processor's ability to read back pixel values should help both with the speed of this processing and with the handling of special cases like these which occur only at a small percentage of the pixels.

Soft Shadows and Multiple Light Sources. Vicki Interrante is implementing a capability to specify multiple light sources. Already implemented is the ability to have an area, rather than point light source, so shadows can appear more naturally soft on the edges, with true penumbras. This is achieved by moving the light source slightly during each of the multiple passes that are normally done for anti-aliasing; just as area sampling within a pixel is approximated by random point sampling within that area, area sampling of the light source is approximated by random point sampling within the light's area. Interrante is now generalizing this method to allow multiple, arbitrarily positioned light sources.

Adaptive Histogram Equalization By Progressive Refinement. John Austin (a member of the project until recently joining Sun Microsystems in Raleigh, NC) implemented a version of Stephen Pizer's AHE algorithm for image enhancement that works by progressive refinement. This image enhancement technique, which takes more than one hour on a VAX 780, has been done in about 4 seconds on Pixel-planes 4. Austin adapted his method to calculate an approximate version of the enhanced image in a small fraction of a second and then refine it. This method works very well. Our medical colleagues are eager to use Pixel-planes for this and other applications.

Specular Phong Shading Studies. John Eyles has developed a fast new method of Phong shading for Pixel-planes. His method performs only part of the rendering of each polygon during that polygon's processing period and leaves the final calculations to be done after all the polygons have been processed. Then each pixel performs the final calculations with the already stored parameters for the surface that is visible at that pixel. Specifically, during each polygon's processing period, the linear

coefficients for each component of the normal to the surface are broadcast. Then, after all the polygons have been processed, each pixel processor re-normalizes the "normal" that it is storing and performs the lighting calculations precisely for its visible surface. Although there are not enough bits per pixel in Pxpl4 to implement this method fully, it will provide in Pxpl5 the much more realistic specular shading (with multiple light sources) at almost the same speed as the cruder Gouraud shading we currently use.

Curved Surfaces. Howard Good has started to render curved surfaces on Pxpl4, using recursive subdivision within the graphics processor followed by rendering of polygonal approximation in the enhanced memory chips. Good is now implementing adaptive subdivision methods. With Pxpl5, we hope to use the enhanced memory chips for a larger share of the computations. Both the quadratic expression evaluation and the memory readout capabilities of Pxpl5 will be used for this task.

Mandelbrot and Julia Sets. Greg Turk has developed an algorithm for displaying Mandelbrot and Julia sets. It allows the user to explore the 2D plane of the set interactively, while the image is updated in real time. Since all pixel processors are enabled virtually all of the time, this algorithm gives far better utilization of the raw computational power of Pxpl4 than we have achieved with any other algorithm; to update these images on the 512 x 512 display at 25 Hz, the enhanced memories perform 1300 million 15-bit adds plus 655 million 15-bit multiplies per second.

PHIGS+ Compatibility. David Ellsworth and Brice Tebbs have begun implementing a programmer's interface to Pxpl4 that is very close to a subset of PHIGS+, the latest version of the Programmer's Hierarchical Interactive Graphics Standard. This implementation is just beginning to run, with only polygons currently supported (other support is expected within weeks). The basic interface is expected to remain virtually the same for casual users of Pxpl5.

2.3 Applications on Pxpl4

Building Walkthrough. As part of a project for exploring and modifying buildings in early design stages, John Airey, under the direction of Professor Fred Brooks, is developing a new visible surface algorithm optimized for dense, static environments in which the viewing position is inside that environment. Airey's method divides the environment automatically into convex regions and associates with each region the list of polygons possibly visible from any point in that region. In contrast to applications in which the rendered object, say, a mechanical part, covers only a part of the screen, in the building "walkthrough" application, every pixel on the screen is covered by some part of the rendered object. Although Airey's method can use any display device, it is particularly well suited for Pxpl4 (and Pxpl5), on which rendering time is only a function of the number of polygons, not the size of each polygon on the screen.

Molecular Modeling. Michael Pique (of Scripps Clinic, La Jolla, CA) and others on the GRIP molecular modeling team (led by Professor Fred Brooks) have programmed Pxpl4 to display molecular vibrations. They are eager to program Pxpl4 to help with molecular docking studies but await the enhancement that allows pixel memories to be read back by the graphics processor.

Medical Imaging. Pxpl4 is used regularly and extensively for display of 3D anatomical structures and radiation therapy doses by the NIH-funded Med3D project led by Professors Henry Fuchs and Stephen Pizer. Kevin Novins, while employed by the UNC School of Medicine's Division of Radiation Oncology, developed a method of reducing the number of polygons in a reconstructed object with minimal degradation to its appearance. This work was motivated in part by the use of Pxpl4, on which the same surface represented by fewer polygons is displayed more rapidly. On other systems, which are more limited by the number of pixels on a surface than by the number of polygons, such polygon-reduction methods would not offer such great advantages.

3. Inadequacies of the Current System and New Opportunities

Although the Pxpl4 system's availability for more than a year has allowed us and others to explore new research, its limitations, both in performance and generality, leave us dissatisfied. We want not only to expand the present capacities of the system, but we also plan to explore some new avenues that experience with it has suggested:

Polygon Rendering Rate. For many situations, especially in medical imaging, we and our collaborators would like much higher rendering rate. Often, for instance, we would like to model a cancer patient's anatomy, radiation treatment beams, and radiation isodensity surfaces with 50,000 or more polygons, and we would like to manipulate smoothly the model on the screen to understand its structural subtleties. We need a 10X speedup to achieve this kind of modeling. More advanced visualization mechanisms, such as stereo, head-motion parallax and head-mounted display, put still greater requirements on the polygon rendering rate.

Memory Readback. Many new applications we would like to pursue require the ability to process the results of computations in the frame buffer. Currently these results can only go to the video stream. In the near future we might enhance our video controller to put these results, under program control, on the multibus, for access by the graphics processor or the system host. For the long run, however, we would like this memory access to be more rapid and flexible than just coming through the video stream. Algorithms for collision detection in molecular modeling and for the calculation of radiosity "form factors" could be sped up considerably with the pixel readback capability.

More Memory. As with virtually any computer, many applications run out of memory. Although 72 bits is a reasonable size for a frame buffer, it is very small for a memory system. Since additional memory comes at the expense of other silicon resources, a reasonable solution to the memory limitation problem is to implement a secondary memory mechanism.

Volumetric Rendering. For many medical situations (and for some molecular modeling situations), we expect that volumetric rendering will be the best imaging modality. Two separate investigations by PhD students Marc Levoy and Lee Westover are producing very exciting images. Mostly, however, these take a long time to generate, since they want to avoid voxel artifacts in their images. Although at least one group of researchers has demonstrated a special-purpose machine for generating volumetric images rapidly [Goldwasser, 1988], even they estimate that a full-size real-time version of their machine

will cost about $250,000. (Even this machine's images would have obvious voxel artifacts!) We would like to render volumetric images, but within a more general-purpose graphics engine.

4. Pixel-planes 5 Design

We expect Pxpl5 to be a dramatically more powerful machine than Pxpl4 for two reasons: 1) it will be much faster on currently-run algorithms due to the higher clock rates, more parallelism throughout screen partitioning, and more front-end graphics processor power; and 2) the architecture will be far less restricted than Pxpl4 with backing store and random access to pixels by the front-end graphics processor (and other parts of the system), which should allow whole new classes of algorithms to be implemented. In addition to its enhanced performance, Pixel-planes 5 will be configurable in a variety of ways that allow cost to be traded for performance.

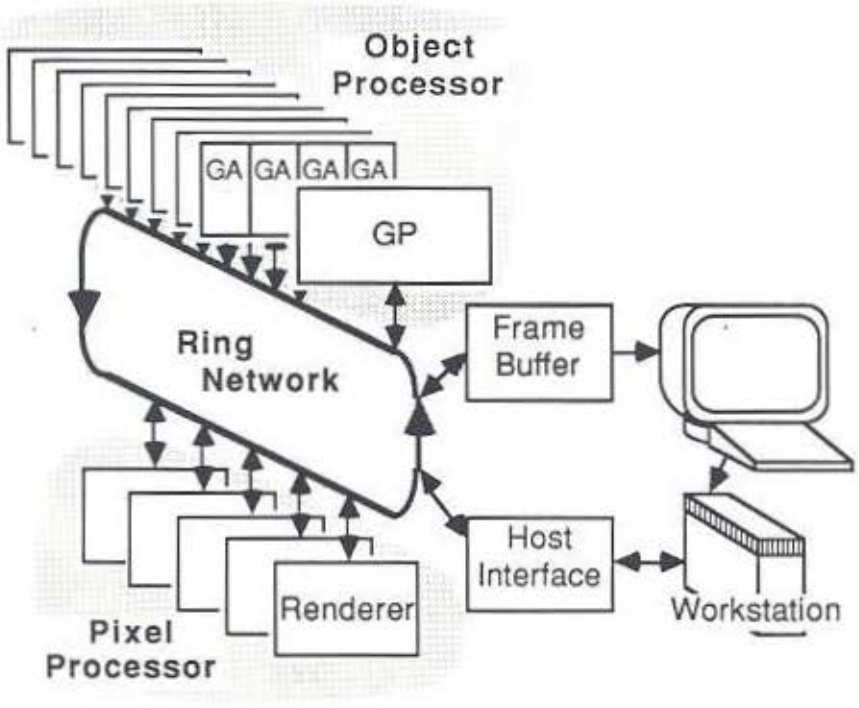

Figure 2: Pixel-planes 5 System Overview

The machine will also be much smaller physically than Pixel-planes 4 and will consume far less electrical power. The system's key features are:

• **Screen Partitioning.** When rendering polygons, Pixel-Planes 4 first disables all pixels on the outside of the first edge of the polygon. Subsequently, all pixels outside the polygon are disabled, and not until the beginning of the next polygon are any of these pixels' processors performing any useful computation. Since complex databases generally contain mostly very small polygons, the processors of the enhanced memories have a very low utilization. To remedy this, we plan to provide the means for

processing simultaneously multiple polygons, lying in disjoint parts of the display screen. To implement this screen partitioning, the logic-enhanced frame buffer will be partitioned into a number of disjoint pieces, each controlled by a separate image generation controller (IGC). Each IGC has an FIFO input buffer. The graphics processor (which computes the A,B,C coefficient sets and frame-buffer instructions) broadcasts each primitive only to those IGC's whose portion of the display is covered by that primitive. Since polygons will be quite small, the majority of polygons will be sent to only one IGC. Thus, if the IGC's FIFO's are deep enough, and if the order of the primitives is reasonably uncorrelated with their screen position, the multiple IGC's should be able to process several different primitives simultaneously. Simulations with even simple data sets and small FIFO's show speed-ups of 4X or better for a system with 16 IGC's.

• **Backing Store.** The new enhanced memory chips will communicate through a special I/O port with a large dynamic RAM "backing store," thus forming a two-tiered pixel memory system. The backing store RAM will support direct addressing of pixel data and will allow rapid block transfers of pixel data to other system components, including a conventional frame buffer from which the display screen will be refreshed. Since pixels can also be moved rapidly to the elements of the MIMD front end, it can be used to carry out pixel-level calculations that cannot be readily done on the SIMD array of pixel processors. Rapid texturing will likely be supported in this way.

• **Separate Refresh Image Buffer.** A separate frame buffer will allow a flexible mapping between the pixel processors in the enhanced memory chips (EMC) and the pixels on the screen. This will permit much more effective utilization of these processors, since the EMC's can be dynamically assigned to those portions of the screen for which there is the most work. This feature will also allow much smaller configurations of Pxpl5 systems without sacrificing significant performance for many applications.

• **Fast Communications Network.** The building blocks of the system, called "application boards," will be linked together by a high-speed ring network, capable of moving data between the memory subsystems of the application boards at 160 mega-words per second.

• **MIMD Graphics Processor.** The "front end" of Pxpl5 will consist of about 32 processors built with the Weitek XL chip set. One of these processors will have more memory resources than the others and will be the master of the system, although with no special privileges on the ring network.

• **Faster Enhanced Memory Chips.** We are re-implementing our custom enhanced memory chips in 1.2μ CMOS. We plan the new chips to have a 40MHz instruction rate, 5 times that of the Pxpl4 chips, and to contain 256 pixels, with 256 bits per pixel. The new chips will also employ a quadratic expression evaluator (in place of the linear one in Pxpl4 chips); quadratic expressions are especially useful for rendering spheres for molecular modeling, for rendering objects described by CSG, and for rendering certain curves and surfaces. The quadratic expression evaluator and the additional memory will also speed up the calculation of higher-order polynomials.

• **Physical Constraints.** Pxpl5 will occupy the volume of a workstation pedestal and should consume no more that 2KW of power. The machine will be built entirely from multi-level printed circuit boards (since wire-wrap is not useful in systems that must run at 40MHz and faster) and will make considerable use of

surface-mount packaging. Pixel-planes 5 will be a more modular system than its predecessor. The pre-processor and enhanced memory array will each be implemented on multiple identical boards. The component boards can be combined in a variety of ways to form a range of configurations from small, inexpensive systems with modest performance to large systems with very high performance.

Current Status. We have finished the logic design for the Pxpl5 enhanced memory chips. The design includes a much more efficient quadratic expression evaluation tree than we had originally developed [Goldfeather and Fuchs, 1986]. Detailed design of the communication network is well under way. We hope to have Pxpl5 operational by mid-1989.

5. Conclusions

We are encouraged that more graphics power is still opening up new opportunities for applications of interactive computer graphics. The widespread use of personal computers and workstations is making 2D graphics so much the norm that one can hardly imagine a workstation or an Apple Macintosh without a graphics display. In the future, there may also be such expectations for 3D graphics capabilities. We hope that we are contributing to the coming of that day.

6. Acknowledgments

The work we have reported was funded jointly by the U.S. National Science Foundation (grant no. MIP-8601552) and the U.S. Defense Advanced Research Projects Agency (order no. 6090).

We thank our team of graduate student research assistants, who developed much of the software environment and many of the algorithms described here: Clare Durand (now with the U.S. Geologolical Service), David Ellsworth, Howard Good, Victoria Interrante, Roman Kuchkuda (now with Megatek, San Diego, CA), Steve Molnar, John Rhoades, Brice Tebbs, and Greg Turk.

Our work in system building would have been impossible without our department's Microelectronic Systems Laboratory, its director Vernon Chi, and staff members John Austin, Mark Monger, John Thomas, and Brad Bennett. Austin and Monger have now joined Sun Microsystems in Raleigh, NC.

We thank Professor Jack Goldfeather of Carleton College for numerous ideas about a wide variety of topics, especially about curved surfaces and quadratic expression evaluation. We thank Mark Kellam and Wayne Dettloff of the Microelectronics Center of North Carolina for help with chip fabrication and system design and testing. We thank our colleague, Professor Frederick P. Brooks, Jr., for years of advice and support and for the sphere-rendering algorithm. We thank our past graduate assistants, Greg Abram, John Cromer, Amarie Helton, Justin Heinecke, Scott Hennes, Cheng-Hong Hsieh, Jeff Hultquist, Alex Melnick, Mary Ranade, and Susan Spach, for years of dedicated, creative effort.

We thank Linda Houseman for patient editing of this paper.

We also thank: The MOSIS Project for IC and circuit board fabrication; John Ousterhout, who has provided tools for the U.S. university VLSI community; Chuck Seitz, whose ideas have greatly influenced our design style [Seitz, 1985]; our colleagues at Xerox PARC, Alan Paeth (now at University of Waterloo), Lynn Conway (now at University of Michigan), and Alan Bell, who collaborated on early designs; Data General for gifts of cabinets and power supplies; and SCI, Inc., for wave soldering services.

7. References and Bibliography

Austin, J. and S. Pizer. 1987. "A Multiprocessor Histogram Equalization Machine," *Proceedings of the Xth Information Processing in Medical Imaging International Conference*, Utrecht, The Netherlands.
Bishop, T.G. and D. Weimer. 1986. "Fast Phong Shading," *Computer Graphics*, 20(4), (Proceedings of SIGGRAPH '86), pp 103-106.

Eyles, J., J. Austin, H. Fuchs, T. Gree, J. Poulton. 1987. "Pixel-planes 4: A Summary," to appear in *Advances in Graphics Hardware 2: Proceedings of the Eurographics '87 Second Workshop on Graphics Hardware.*

Fuchs, H. and J. Poulton. 3rd Quarter, 1981. "Pixel-planes: A VLSI-Oriented Design for a Raster Graphics Engine," *VLSI Design*, 2(3), pp 20-28.

Fuchs, H., J. Poulton, A. Paeth, and A. Bell. January 1982. "Developing Pixel Planes, A Smart Memory-Based Raster Graphics System," *Proceedings of the 1982 MIT Conference on Advanced Research in VLSI*, Dedham, MA, Artech House, pp 137-146.

Fuchs, H., J. Goldfeather, J.P. Hultquist, S. Spach, J.D. Austin, F.P. Brooks, J.G. Eyles, and J. Poulton. July 1985. "Fast Spheres, Shadows, Textures, Transparencies, and Image Enhancements in Pixel-planes," *Computer Graphics*, 19(3), (Proceedings of SIGGRAPH '85), pp 111-120.

Fuchs, H. 1987. "An Introduction to Pixel-planes and other VLSI-intensive Graphics Systems," *Proceedings of the NATO International Advanced Study Institute on Theoretical Foundations of Computer Graphics and CAD*, Il Ciocco International Center, Castelvecchio Pascoli, Lucca, Tuscany, Italy, July 1987 (Springer-Verlag, 1988).

Goldfeather, J. and H. Fuchs. January, 1986. "Quadratic Surface Rendering on a Logic-Enhanced Frame-Buffer Memory System," *IEEE Computer Graphics and Applications*, 6(1), pp. 48-59.

Goldfeather, J., J.P.M. Hultquist, and H. Fuchs. August 1986. "Fast Constructive Solid Geometry Display in the Pixel-Powers Graphics System," *Computer Graphics*, 20(4), (Proceedings of SIGGRAPH '86), pp 107-116.

Goldwasser, S.M., and R.A. Reynolds, D.A. Talton, and E.S. Walsh. January 1988. "High-Performance Graphics Processors for Medical Imaging Applications," these proceedings.

Jansen, F.W. August 1986. "A Pixel-Parallel Hidden Surface Algorithm for Constructive Solid Geometry," *Proceedings of Eurographics '86*, Elsevier Science Publishers B.V. (North-Holland): Amsterdam, New York, pp. 29-40.

Jermoluk, Tom and Kurt Akeley. 1988. "Cost Effective High-Performance Polygon Rendering," submitted for publication. (Authors' address: Silicon Graphics Computer Systems, 2011 Stierlin Road, Mountain View, California 94043, USA.)

Pizer, S.M., J.B. Zimmerman, and E.V. Staab. 1984. "Adaptive Grey Level Assignment in CT Scan Display," *Journal of Computer Assisted Tomography*, 8(2), pp 300-305.

Poulton, J., H. Fuchs, J.D. Austin, J.G. Eyles, J. Heinecke, C-H Hsieh, J. Goldfeather, J.P. Hultquist, and S. Spach. 1985. "PIXEL-PLANES: Building a VLSI-Based Graphic System," *Proceedings of the 1985 Chapel Hill Conference on VLSI*, Rockville, MD, Computer Science Press, pp 35-60.

Poulton, J., H. Fuchs, J.D. Austin, J.G. Eyles, and Trey Greer. 1987. "Building a 512x512 Pixel-planes System," *Proceedings of the 1987 Stanford Conference on Advanced Research in VLSI*, Cambridge, MA, MIT Press, pp 57-71.

Seitz, C.L., A.H. Frey, S. Mattisson, S.D. Rabin, D.A. Speck, and J.L.A. van de Snepscheut. 1985. "Hot-Clock nMOS," *Proceedings of the 1985 Chapel Hill Conference on VLSI*, Rockville, MD, Computer Science Press, pp. 1-17.

Silicon Graphics Computer Systems. 1987. "IRIS GT Graphics Architecture: A Technical Report," Silicon Graphics Computer Systems, 2011 Stierlin Road, Mountain View, California 94043, USA.