PixelFlow: Scalable Graphics

(completed 1997)

Department of Computer Science **University of North Carolina at Chapel Hill**

Summary

For many years we have been exploring computer architectures for 3D graphics that, compared to today's systems, offer dramatically higher performance with wide flexibility for a broad range of applications. A major continuing motivation has been to provide useful systems for our researchers whose work in medical visualization, molecular modeling, and architectural design exploration requires graphics power far beyond that available in today's commercial systems. Since there does not appear to be a sizable market for such "ultra-high-end," and therefore very expensive systems, commercial vendors do not seem to be exploring even the possibilities there. We have thus been able to build systems that exhibit new levels of performance, explore sometimes "revolutionary" graphics architectures, demonstrate the feasibility of some ideas, and at the same time provide systems with which our local colleagues can perform research years before they would be able to conduct it if they had to rely on commercial systems.

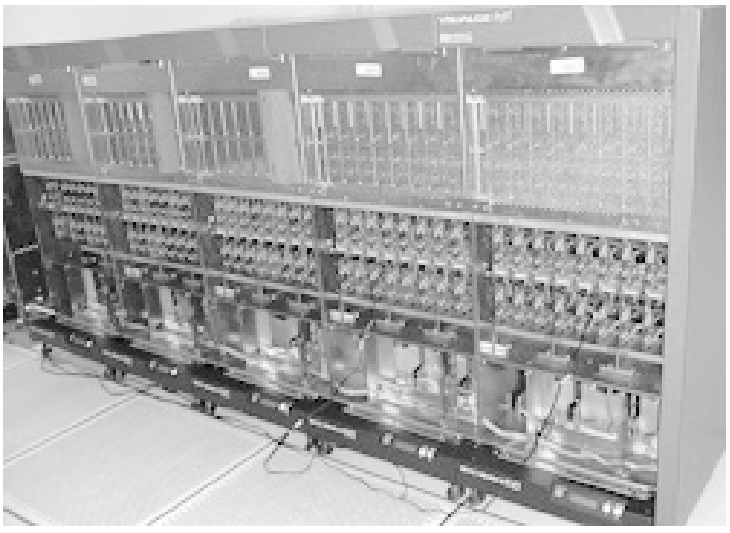
Our latest machine, developed jointly with Hewlett-Packard is PixelFlow, and it is our third full-scale machine design. Our first full-sized machine, Pixel-Planes 4, was originally demonstrated at the ACM SIGGRAPH '86 Conference, and it generated images at the then trailblazing speed of 45,000 Gouraud-shaded polygons per second. It was in daily use as the main interactive graphics machine in our Graphics and Image Lab until 1992 when Pixel-Planes 5 replaced it. Pixel-Planes 5 first became operational in 1990. It was originally designed to deliver one million Phong-shaded polygons per second to an application at interactive rates. It was publicly demonstrated at performance levels of 2.3 million polygons/sec (on a simple terrain fly-through application). In 1993 Pixel-Planes 5 demonstrated the highest performance of any reported system on the most complex national graphics benchmark.

Highlights

- **Scalable graphics performance. More nodes can be added to increase performance almost linearly.**
- **Supports complex shading. The first machine to support complex user-written shading and lighting algorithms (1998 dissertation by Marc Olano).**

Our motivation for building PixelFlow was to create a computer system that would be capable of scalable performance within a broad range of sizes: from single board personal machines to large visualization servers. This capability is unlike any graphics system that we know of (including our previous machines).

Object Parallel Graphics using Image Composition. Image composition architectures are a type of object-parallel graphics hardware architecture in which independent, self-contained graphics units, which we call *Renderers*, each generate a full-screen image of a fraction of the primitives that will be displayed ultimately. These sub-images are then depth composited by a high-speed, image-composition network to form the final image.



PixelFlow

The main advantages of this approach are that network bandwidth is fixed by the display requirements for frame rate, bits-per-pixel, and screen size, and all links in the network carry the same bandwidth. This leads to the important property that image-composition architectures are *linearly scalable to very high performance.*

Furthermore, the system is built of an array of identical hardware units, each of which is a simple graphics pipeline that computes its sub-image essentially independently of the other Renderers. Renderers, therefore, can operate with little synchronization other than the low-level control to assure that pixels are fed into the composition network in the proper order. Scalable performance and identical building blocks result in a system that can grow from modest configurations as small as one board, to very large systems of unprecedented graphics performance.

System Design. The backplane implements an image composition network on which partial images generated on each system board are composited into a single final image. The addition of an optional daughter card may augment any board for video output to displays or input from cameras or sensors. Any board can likewise have an I/O daughter card that supports communication with a host computer, graphics input devices, disks, or communications networks. The machine is intended to be a scalable attached graphics processor for a parallel computer host but can also be hosted, for example, by a workstation or workstation cluster. The system can also drive a large number of independent displays, making it useful for video walls and other large-format display systems.

Software System. The PixelFlow application programmer interacts mainly with a software library that implements a system similar to OpenGL. This layer, the Application Programmer's Interface (API), is not the only one that a user can access, but we expect that it will suffice for the majority of the applications. For more demanding applications that want to exert control over lower-level functions, such as applications that want explicitly to control the parallelism, we provide access to other layers.

The PixelFlow system uses deferred shading—shading that is not computed until *after* final visibility has been determined, *i.e.*, after depth composition. Unlike the majority of graphics systems, the PixelFlow software does not generate pixel color during scan conversion, but rather scan converts the *appearance parameters*, such as normals and texture coordinates, necessary to shade the pixel. These pixel-level parameters are sent over the composition network to a Renderer that executes *shading functions* to determine the color of a pixel. These shading functions are written by the user, eventually in a high-level language similar to Pixar's RenderMan. During normal rendering, the pixel color is sent from the shading node to the frame-buffer node for display. Optionally, code may be executed at this node to perform *image manipulations*, such as warping.

Status

The PixelFlow prototype was completed jointly by UNC–Chapel Hill and Hewlett-Packard Co. in the summer of 1997 and was demonstrated at SIGGRAPH '97. The Department of Computer Science has the 54-node machine at work in its Graphics and Image Laboratory.

Past Project Participants

Project Leaders

Henry Fuchs (Co-Principal Investigator), Federico Gil professor

John Poulton (Co-Principal Investigator), research professor

Research Faculty and Staff

Lynne Cohen Duncan, systems programmer/administrator
Nick England, research professor
John G. Eyles, research associate professor
Kurtis Keller, research engineer
Anselmo Lastra, research associate professor
Steven Molnar, adjunct assistant professor
Sherry Palmer, program assistant
Stephen G. Tell, senior research associate
John Thomas, research associate and manager of the Microelectronic Systems Laboratory
Leandra Vicci, lecturer and director of the Microelectronic Systems Laboratory

Graduate Research Assistants

Daniel G. Aliaga, Jonathan Cohen, David Ellsworth, Arthur Gregory, Paul Hilts, Paul Keller, Lawrence Kesteloot, Jonathan Leech, David McAllister, Jonathan McAllister, Peter McMurry, Carl Mueller, Bryon Nordquist, Marc Olano, M. K. Ponamgi, Voicu Popescu, Gregory Pruett, Yulan Wang, Gregory F. Welch, Robert Wheeler

Research Sponsors

Advanced Research Projects Agency (Order No. A410)
National Science Foundation (Grant No. MIP-9306208)
Significant Additional Assistance From: Division Group PLC; Hewlett-Packard Co.

For More Information

Dr. Anselmo Lastra
Department of Computer Science
University of North Carolina at Chapel Hill
CB#3175, Sitterson Hall
Chapel Hill, NC 27599-3175
Phone: (919) 962-1958
Fax: (919) 962-1799
E-mail: lastra@cs.unc.edu
Web: www.cs.unc.edu/~pxfl/