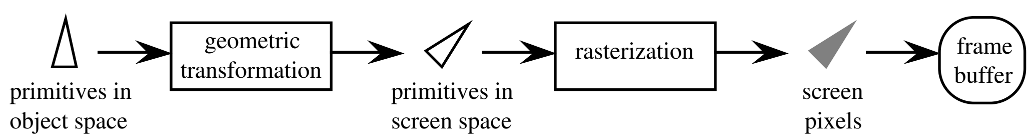
High-Performance Rendering via the Sort-First Architecture

A Dissertation Proposal by
Carl Mueller
November 17, 1993

1. Inroduction

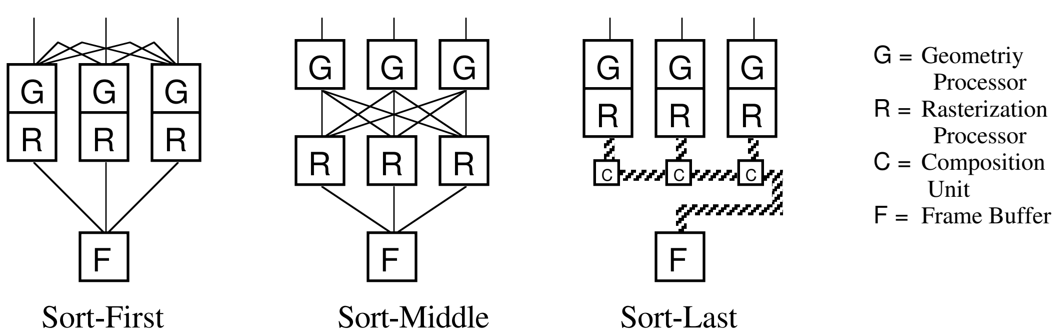
Applications involving interactive graphics are becoming more and more commonplace. As computer systems rise to meet the needs of such applications, new demands arise. Simple display models and single output screens have long been obsolete for cutting-edge applications such as vehicle simulation, architectural walkthrough, and scientific visualization. The need for large, detailed models and multiple high-resolution screens challenges the limits of current graphics computer architectures. To satisfy these criteria, very high communication bandwidth is required for both primitive and pixel traffic. My research focuses on an architecture that will address these requirements and provide a proper solution for the types of applications mentioned. The architecture that I will detail is known as "sort-first."

The name sort-first comes from the way the rendering algorithm is mapped to a parallel computer architecture. In the general rendering algorithm, one starts with primitives in object space; these are transformed by matrix calculations into screen space; and finally the primitives are rasterized into pixels:



In a multiprocessor system, the primitives, which are initially distributed among the processors, must be "sorted" at some point in the pipeline so that they end up affecting the right screen pixels. In fact, this sorting may happen before or after either of the two operations shown, thus leading to the three categories of algorithms: sort-first, sort-middle, and sort-last.

In a sort-last system the primitive work-load remains independent until the final stage when pixels are composited. A sort-middle system has each processor transform its primitives and then send them off to other processors (mapped to screen regions) for rasterization. In a sort-first system, the primitives must start out at the processor corresponding to the screen region where the primitive will finally be displayed. Primitives found on the "wrong" processor must be relocated to the correct ones. This capitalizes on the fact that viewpoint changes are usually quite small, meaning few primitives usually must be communicated. The sorting methods are summarized in the drawings below.



The different algorithms lead to different architectures when a rendering engine is designed around them. The different architectures will obviously have their own sets of advantages and disadvantages, depending upon the nature of the application chosen to run on them. The question of which architecture best supports the aforementioned applications I believe I can answer simply: sort-first. After showing this fact, however, I must show that I can design a feasbile sort-first system that meets the appropriate criteria.

2. Prior work

While many different parallel rendering systems have been proposed and built, very few reach the level of parallelization necessary to even apply the sorting paradigms mentioned. Sort-middle systems were the first to surpass the one-million triangle-per-second performance mark [FUCH89, AKEL93]. Since previous systems had already applied parallelization separately to the transformation and rasterization stages in the traditional rendering pipeline, it was only natural that these should be applied together to achieve sort-middle. Ellsworth discusses advantages and techniques that may be applied in sort-middle systems [ELLS93].

Sort-last has existed in various forms since at least 1967. Molnar classifies various sort-last systems and describes the issues required to make a high-performance sort-last system [MOLN91]. Molnar is also the first to publish the sorting paradigm for parallel rendering systems.

The concept of assigning different processors (with independent rendering pipelines) to different areas of the screen has been around for a long time. However, as of this writing, there an no known true sort-first systems that use primitive shuffling to subdivide the transformation task among the processors.

3. Statement of thesis

I propose that for interactive graphics applications having the following characteristics:

- large screen resolution
  (in excess of 4M pixels, implemented with one or more displays)
- a large number of primitives to be displayed
  (on the order of 200K primitives)
- a primarily static display database

one can design a reasonable system based upon the sort-first architecture, whereas a system based upon sort-middle or sort-last would require unreasonable communication bandwidths.

4. Method of demonstration of thesis

The requirement for large screen resolution will rule out a sort-last system. This is because the bandwidth requirement for sort-last's pixel composition network is directly proportional to the desired screen resolution. A large enough resolution will simply be impossible for sort-last to support at interactive rates with today's technology. Even an architecture such as PixelFlow, with 3200 MW/s bandwidth on its composition bus, will only just barely produce a megapixel image at interactive rates without sacrificing either antialiasing or deferred shading features.

The two application requirements cause major problems for sort-middle as well. Both place large bandwidth requirements on its primitive distribution network. Since sort-middle requires that each transform processor send all its primitives to every appropriate rasterization processor each frame, its bandwidth requirements grow proportionally with the number of primitives, and the number of communications grows proportionally with the product of the number of transformation processors and the number of screen regions. These communications requirements can quickly become unreasonable as the problem is scaled upward.

Sort-first, on the other hand, is capable of handling large numbers of both primitives and pixels. Its primitive-handling scheme means that relatively few primitives need to be communicated from frame to frame. And since each processor is completely responsible for the pixels it generates, only finished pixels need to be delivered to the frame buffer(s). Both of these facts mean that sort-first's communication bandwidth requirements should fall well within practical limits for even very large-scale problems.

I must then show how to design a feasible sort-first system. This will require finding solutions to the following problems:

- Load balancing
- Display-list management
- Frame-buffer design

The overhead associated with the solutions should not unduly compromise the scalability of the resulting system.

The load-balancing problem is perhaps the main stumbling block behind a sort-first system. It is important that the screen regions assigned to the different processors have approximately equal numbers of primitives in them; otherwise the more heavily-loaded processors will cause the others to stall. Likely I will not be able to design a load-balancing scheme that works well for all cases. I will show that the schemes I propose will work for "typical" application cases (to find a "best" scheme will probably be a topic onto itself).

I have already devised two schemes: one revolving around adaptively changing screen-region boundaries, another around a static screen subdivision with multiple regions assigned to processors in an interleaved fashion. The adaptive method has a certain amount of processing and communication overhead which will be detailed. The static method places restrictions on the processor configuration.

Analysis of a load-balancing scheme must carefully consider the communication bandwidth required by primitive traffic. This will be studied through the use of a simulation system in order to demonstrate that the methods I under study shall have characteristics in keeping with a reasonable sort-first system.

The management of the display list is another stumbling block of sort-first. Since the primitives actually migrate from processor to processor, the tasks of distributing, keeping up with, and modifying the display list become nontrivial. In particular, support for a hierarchical display database with instancing will be a difficult challenge. Solutions will be proposed.

Frame-buffer design is an issue that cannot be ignored. Pixel traffic for large resolution screens at interactive rates demands considerable bandwidth. For the adaptive scheme, I will explore possible solutions and examine their requirements. For certain arrangements of the static subdivision, I can devise a scheme that requires no pixel traffic and no unused frame-buffer memory. It will, however, require a special video-switching technique which I will detail and show to be viable.

5. Stopping criterion

I must come up with a feasible sort-first system. I will lay out general hardware specifications for a possible system. I will then show:

-A load-balancing algorithm (or two) that satisfies real-time performance requirements for the driving applications. The algorithms should not place severe limitations on scalability.

-The system's primitive-communication patterns will not require inordinate bandwidth to offer interactive performance.

-A method of distributing and managing a hierarchical display database.

-A method for performing limited editing on this database at interactive rates.

-A frame-buffer design that is technologically feasible and does not require inordinate amounts of memory or bandwidth.

In addition, I will implement a convincing demonstration that shows beyond doubt the feasibility and worth of the sort-first approach. The machine that will serve as a base for this is yet to be determined. PixelFlow is a prime candidate, pending its availability.

6. Schedule for completion of the work

While some of the issues are overlapping and will require common study, here is a general breakdown:

Fall 1993:

* Solve load-balancing issues
  - Define characteristics of a sort-first machine.
  - Expand existing simulation system to include these characteristics.
  - Gather traces from "typical applications".
  - Run simulations with various load-balancing options.
    (some preliminary solutions for problems below will be incorporated)
  - Analyze the results and adjust as necessary.
* Study for orals

Spring 1994:

* Take orals
* Solve display-database handling problems:
  - Study and find solutions for distributing the database.
  - Study and find solutions for display-list editing.
  - Apply more simulations as appropriate.
* Solve frame-buffer design issues:
  - Study and find solutions for dynamic screen regions.
  - Study proposed solution for fixed interlaced screen regions.
  - Apply more simulations as appropriate.

Summer 1994:

* Solve unforseen issues
* Perhaps implement limited system prototype.
* Writing for dissertation

Fall 1994:

* Finish writing
* Defend

7. Proposed topic areas for oral exam

graphics:
- basic primitive rendering
- image sampling and filtering
- polygon rendering pipelines
- hierarchical display database structure
- advanced raster graphics architecture
- video hardware
- flight-simulator technology

architecture:
- processing units and pipelining
- memory subsystems

parallel computing:
- parallel algorithms on distributed databases
- architectures (overview)
- network topologies and communication issues (overview)

etc:
- high-level modeling of parallel systems
- computer performance analysis (overview)

Annotated Bibliography

[AKEL88] Akeley, K. and T. Jermoluk, "High-Performance Polygon Rendering," Computer Graphics, (Proceedings of SIGGRAPH '88), Vol. 22, No. 4, pp. 239-246.

Describes the steps taken to accomplish 100K quads/sec. on the 4D/240GTX.

[AKEL89] Akeley, K., "The Silicon Graphics 4D/240GTX Superworkstation," IEEE Computer Graphics and Applications, Vol. 9, No. 4, July 1989, pp. 71-83.

Talks about host/graphics-subsystem issues, immediate mode, and also describes the 4D/240GTX architecture.

[AKEL93] Akeley, K., "RealityEngine Graphics," Computer Graphics (Proceedings of SIGGRAPH '93), August 1993, pp. 109-116.

Describes the architecture of the SGI Reality Engine system.

[APGA88] Apgar, B., B. Bersack, and A. Mammen, "A Display System for the Stellar Graphics supercomputer Model GS1000," Computer Graphics (Proceedings of SIGGRAPH '88), Vol. 22, No. 4, pp. 255-262.

Discusses the additions to the GS1000 supercomputer to make it a graphics computer as well (basically adds a specialized 4x4 SIMD array processor to the bus). The resulting system achieves over 100K polygons/sec.

[CRUZ93a] Cruz-Neira, Carolina, D. Sandin, T. DeFanti, "Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE," Computer Graphics (Proceedings of SIGGRAPH '93), August 1993, pp. 135-142.

Describes the CAVE, a VE setup with projection screens on 3 sides plus the bottom. Stereo images (viewed with shutter glasses) are generated by 4 SGIs. Targetted at education and scientific visualization. Notes how fixed-display VEs offer fewer tracking problems than HMDs.

[CRUZ93b] Cruz-Neira, Carolina, J. Leigh, M. Papka, C. Barnes, S. Cohen, S. Das, R. Engelmann, R. Hudson, T. Roy, L. Siegel, C. Vasilakis, T. DeFanti, D. Sandin, “Scientists in Wonderland, A Report on Visualization Applications in the CAVE Virtual Reality Environment,” Proceedings of the 1993 Parallel Rendering Symposium (San Jose, CA, Oct 25-26, 1993).

Describes various applications that have been implemented for the CAVE.

[DEER88] Deering, M., S. Winner, B. Schediwey, C. Duffy, and N. Hunt, "The Triangle Processor and Normal Vector Shader: A VLSI System for High Performance Graphics," Computer Graphics (Proceedings of SIGGRAPH '88), Vol. 22, No. 4, pp. 21-30.

Describes a processor-per-triangle, deeply pipelined, scan-line based VLSI system. Also uses deferred shading for pixels plus an accumulation buffer for fancy effects plus taking care of overflow (more triangles than processors in one scan line).

[DEER92] Deering, Michael, "High Resolution Virtual Reality," Computer Graphics (Proceedings of SIGGRAPH '92), Vol. 26, No. 2, July 1992, pp. 195-202.

Describes a head-tracked stereo CRT display system where exact correspondence with the physical world was the goal. Mentions that the field of view for a stereo CRT is limited to 20-60 degrees, and "Projection stereo displays (perhaps multiple thereof) can support wider fields of view."

[DEER93] Deering, Michael, S. Nelson, "Leo: A System for Cost Effective 3D Shaded Graphics," Computer Graphics (Proceedings of SIGGRAPH '93), August 1993, pp. 101-108.

Describes Sun's approach to packing a limited sort-middle approach into a desktop workstation.

[ELLS89] Ellsworth, David, Howard Good, and Brice Tebbs, "Distributing Display Lists on a Multicomputer," Computer Grapphics (Proceedings of SIGGRAPH '90), Vol. 24, No. 2, March 1990, pp. 147-155.

Discusses distribution of display hierarchies in PPHIGS on Pxpl5. Mentions two approaches: distributing structures and distributing primitives. Talks in detail about problems and optimizations of the latter.

[ELLS93] Ellsworth, David, "A Multicomputer Polygon Rendering Algorithm for Interactive Applications," Proceedings of the 1993 Parallel Rendering Symposium (San Jose, CA, Oct 25-26, 1993), pp. 43-48.

Describes a sort-middle implementation on the Intel Delta.

[FOLE90] Foley, J.D., A. van Dam, S.K. Feiner, and J.F. Hughes, Computer Graphics: Principles and Practice, Addison-Wesley, Reading, MA, 1990.

Chapter 18 reviews advanced graphics architectures (see [MOLN90]).

[FUCH89] Fuchs, Henry, John Poulton, John Eyles, Trey Greer, Jack Goldfeather, David Ellsworth, Steve Molnar, Greg Turk, Brice Tebbs, and Laura Israel, "Pixel-Planes 5: A Heterogeneous Multiprocessor Graphics System Using Processor-Enhanced Memories," Computer Graphics (Proceedings of SIGGRAPH '89), Vol. 23, No. 3, July 1989, pp. 79-88.

General overview of Pixel-Planes 5, a true sort-middle system.

[GHAR88] Gharachorloo, N., S. Gupta, E. Hokenek, P. Balasubramanian, B. Bogholtz, C. Mathieu, and C. Zoulas, "Subnanosecond Pixel Rendering with Million Transistor Chips," Computer Graphics (Proceedings of SIGGRAPH '88), Vol. 22, No. 4, pp. 41-49.

Describes SAGE, a linear array processor designed to fill pixel spans at high pixel rates. Combined with VIP (vert. interp. proc.), it may be able to produce 1M gouraud-shaded tris per sec.

[GHAR88] Gharachorloo, N., Satish Gupta, Robert F. Sproull, and Ivan E. Sutherland, "A Characterization of Ten Rasterization Techniques," Computer Graphics (Proceedings of SIGGRAPH '89), Vol. 23, No. 3, July 1989, pp. 355-368.

Presents a hierarchical taxonomy of certain rasterization techniques. It is mostly concerned with frame-buffer access techniques. Has a chart showing parameters and formulas that determine performance limits.

[HARR93] Harrell, Chandlee, F. Fouladi, "Graphics Rendering Architecture for a High Performance Desktop Workstation," Computer Graphics (Proceedings of SIGGRAPH '93), August 1993, pp. 93-100.

Describes SGI's approach to packing a limited sort-middle approach into a desktop workstation.

[HU85] Hu, Mei-Cheng, and James Foley, "Parallel Processing Approaches to Hidden Surface Removal," Computers & Graphics, Vol. 9, No. 3, 1985, pp. 303-317.

Considers different ways for attaching multiple rasterization processors to frame-buffer memory: static assignment to contiguous regions, static assignment to interlaced scan lines, and dynamic assignment to scan lines (in this case, processors are attached to memory with a switching network). This study favors the last approach greatly.

[MOLN90] Molnar, Steven and Henry Fuchs, "Advanced Raster Graphics Architecture," Chapter 18 in Computer Graphics: Principles and Practice, by J.D. Foley, A. van Dam, S.K. Feiner, and J.F. Hughes, Addison-Wesley, Reading, MA, 1990, pp. 855-922.

Describes issues associated with making high performance graphics systems.

[MOLN91] Molnar, Steven E., "Image Composition Architectures for Real-Time Image Generation," Ph.D. dissertation, also available as UNC-Computer Science Technical Report TR91-046, 1991.

Talks about issues related to high-speed rendering and describes in detail an image-composition architecture.

[NEUM93] Ulrich Neumann, "Volume Reconstruction and Parallel Rendering Algorithms," Ph.D. dissertation, also available as UNC-Computer Science Technical Report TR93-017, 1993.

Describes a taxonomy of volume rendering systems. Mentions that using sort-first methods may be helpful to improve rendering performance.

[NIIM84] Haruo Niimi, Yoshirou Imai, Masayoshi Murakami, Shinji Tomita & Hiroshi Hagiwara, "A Parallel Processor For Three-Dimensional Color Graphics," Computer Graphics (Proceedings of SIGGRAPH '84), Vol. 18, No. 3, July 1984, pp. 67-76.

Describes a system named "EXPERT" that uses multiple Scan-Line Processors (divides screen into contiguous sections) and PiXel Processors (divide up scan lines). Goes into the details of this architecture & implementation. Uses lots of double buffering to avoid contention. Describes a simple antialiasing method. Does some analysis based upon microcode execution time, expected paths through code, screen size, # of polys, and depth complexity.

[PARK80] Parke, Frederic, "Simulation and Expected Performance Analysis of Multiple Processor Z-buffer Systems," Computer Graphics (Proceedings of SIGGRAPH '80), Vol. 14, No. 3, July 1980, pp. 48-56.

Studies three rasterization architectures: 1) broadcast all primitives to all processors, dividing up the screen in interlaced fashion, 2) a tree of splitters in front of all the processors partition the primitives into contiguous screen regions (1 per processor), and 3) a hybrid approach, with splitters subdividing the screen to some vel and then diving the screen into interlaced regions. Includes a performance analysis and simulations with lots of graphs.

[THEO89] Theoharis, Theoharis, and Ian Page, "Polygon Rendering on a Dual-Paradigm Parallel Processor," Computers & Graphics, Vol. 13, No. 2, 1989, pp. 207-216.

Describes a system for rendering polygons on the Disputer, a system with Transputers attached to a small SIMD array. Mostly covers implementation details of the rendering method.

[TORB87] Torborg, John, "A Parallel Processor Architecture for Graphics Arithmetic Operations," Computer Graphics (Proceedings of SIGGRAPH '87), Vol. 21, No. 4, July 1987, pp. 197-204.

Describes a multiple-bus-based prototype system with several [1-8] transformation engines and a single rasterization processor. Much of the paper is devoted to the synchronization required between the stages for various types of operations. Simulation (the system was in "debug" stage at the time) indicates 90K tri/sec perf.

[WHIT90a] Whitman, Scott, "Computer Graphics Rendering on a Parallel Processor," in Scott Whitman, Ed., Parallel Algorithms and Architectures for 3D Image Generation, Course 28 Notes for SIGGRAPH '90, pp. 167-182.

Focuses on load-balancing for MIMD shared-memory algorithms (impl. on the BBN Butterfly GP1000). Looks at data non-adaptive partitioning, data-adaptive partitioning, and task-adaptive partitioning. Offers some informative graphs.

[WHIT90b] Whitman, Scott, and Richard Parent, "A Survey of Parallel Hidden Surface Removal Algorithms," in Scott Whitman, Ed., Parallel Algorithms and Architectures for 3D Image Generation, Course 28 Notes for SIGGRAPH '90, pp. 167-182.

Contains many brief summaries of several hidden surface algorithms for various parallel architectures.