

# Foreword

---

Composition, the organization of elemental operations into a nonobvious whole, is the essence of imperative programming. The instruction set architecture (ISA) of a microprocessor is a versatile composition interface, which programmers of software renderers have used effectively and creatively in their quest for image realism. Early graphics hardware increased rendering performance, but often at a high cost in composability, and thus in programmability and application innovation. Hardware with microprocessor-like programmability did evolve (for example, the Ikonas Graphics System), but the dominant form of graphics hardware acceleration has been organized around a fixed sequence of rendering operations, often referred to as the *graphics pipeline*. Early interfaces to these systems—such as CORE and later, PHIGS—allowed programmers to specify rendering results, but they were not designed for composition.

OpenGL, which I helped to evolve from its Silicon Graphics-defined predecessor IRIS GL in the early 1990s, addressed the need for composability by specifying an architecture (informally called the *OpenGL Machine*) that was accessed through an imperative programmatic interface. Many features—for example, tightly specified semantics; table-driven operations such as stencil and depth-buffer functions; texture mapping exposed as a general 1D, 2D, and 3D lookup function; and required repeatability properties—ensured that programmers could compose OpenGL operations with powerful and reliable results. Some of the useful techniques that OpenGL enabled include texture-based volume rendering, shadow volumes using stencil buffers, and constructive solid geometry algorithms such as capping (the computation of surface planes at the intersections of clipping planes and solid objects defined by polygons). Ultimately, Mark Peercy and the coauthors of the SIGGRAPH 2000 paper “Interactive Multi-Pass Programmable Shading” demonstrated that arbitrary RenderMan shaders could be accelerated through the composition of OpenGL rendering operations.

During this decade, increases in the raw capability of integrated circuit technology allowed the OpenGL architecture (and later, Direct3D) to be extended to expose an

---

ISA interface. These extensions appeared as programmable vertex and fragment shaders within the graphics pipeline and now, with the introduction of CUDA, as a data-parallel ISA in near parity with that of the microprocessor. Although the cycle toward complete microprocessor-like versatility is not complete, the tremendous power of graphics hardware acceleration is more accessible than ever to programmers.

And what computational power it is! At this writing, the NVIDIA GeForce 8800 Ultra performs over 400 billion floating-point operations per second—more than the most powerful supercomputer available a decade ago, and five times more than today’s most powerful microprocessor. The data-parallel programming model the Ultra supports allows its computational power to be harnessed without concern for the number of processors employed. This is critical, because while today’s Ultra already includes over 100 processors, tomorrow’s will include thousands, and then more. With no end in sight to the annual compounding of integrated circuit density known as Moore’s Law, massively parallel systems are clearly the future of computing, with graphics hardware leading the way.

*GPU Gems 3* is a collection of state-of-the-art GPU programming examples. It is about putting data-parallel processing to work. The first four sections focus on graphics-specific applications of GPUs in the areas of geometry, lighting and shadows, rendering, and image effects. Topics in the fifth and sixth sections broaden the scope by providing concrete examples of nongraphical applications that can now be addressed with data-parallel GPU technology. These applications are diverse, ranging from rigid-body simulation to fluid flow simulation, from virus signature matching to encryption and decryption, and from random number generation to computation of the Gaussian.

Where is this all leading? The cover art reminds us that the mind remains the most capable parallel computing system of all. A long-term goal of computer science is to achieve and, ultimately, to surpass the capabilities of the human mind. It’s exciting to think that the computer graphics community, as we identify, address, and master the challenges of massively parallel computing, is contributing to the realization of this dream.

*Kurt Akeley*  
*Microsoft Research*