SAND2006-3612P

Sandia National Laboratories

Fact Sheet

Programmable Hardware Shaders

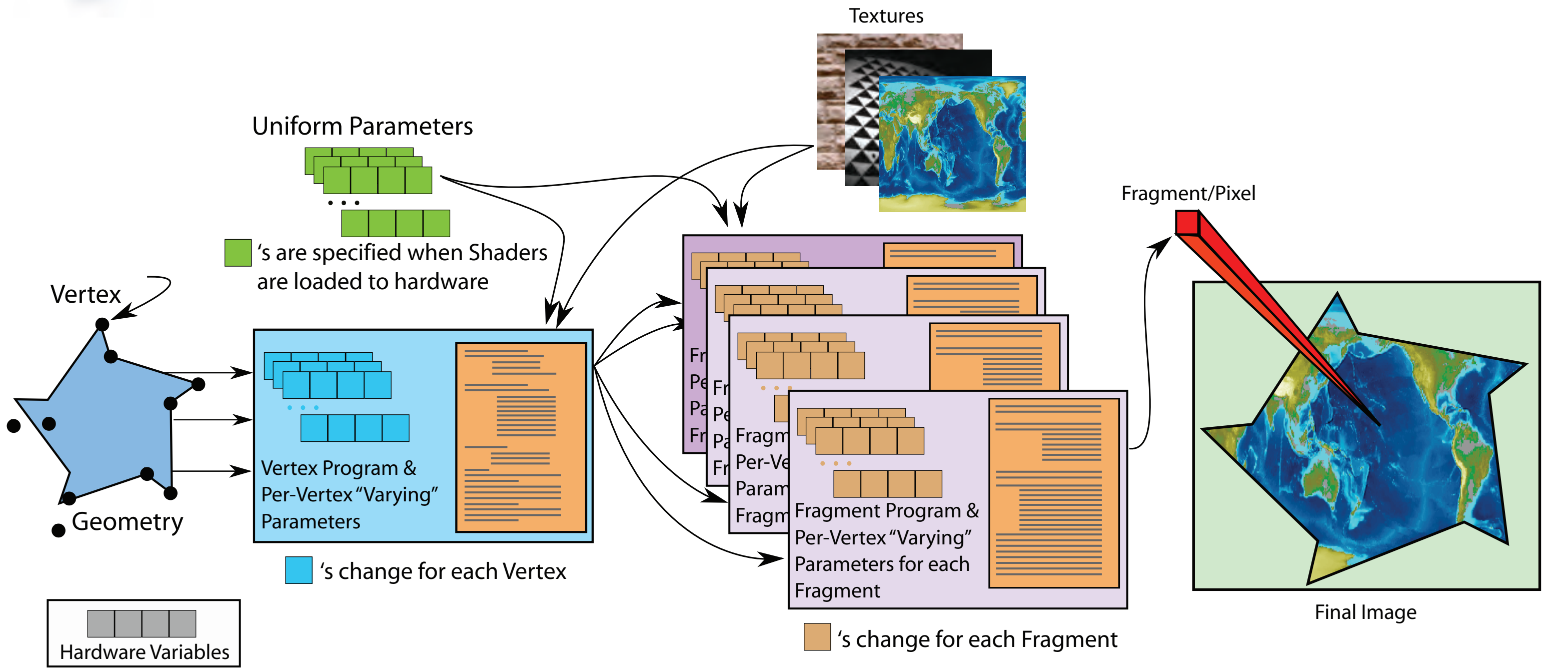

Textures
Uniform Parameters
's are specified when Shaders are loaded to hardware
Fragment/Pixel
Vertex
Geometry
Vertex Program & Per-Vertex "Varying" Parameters
's change for each Vertex
Hardware Variables
Fragment Program & Per-Vertex "Varying" Parameters for each Fragment
's change for each Fragment
Final Image

Figure 1. GPU schematic with programmable hardware shaders.

Programmable hardware shaders can replace the fixed-functionality vertex and pixel shaders traditionally found on Graphics Processing Units (graphics cards or GPUs). This allows greater control of how data are rendered to images and makes possible the use of sophisticated rendering techniques at interactive frame rates. What was once fixed in hardware is now programmable, making available to application developers a wide range of graphics techniques.

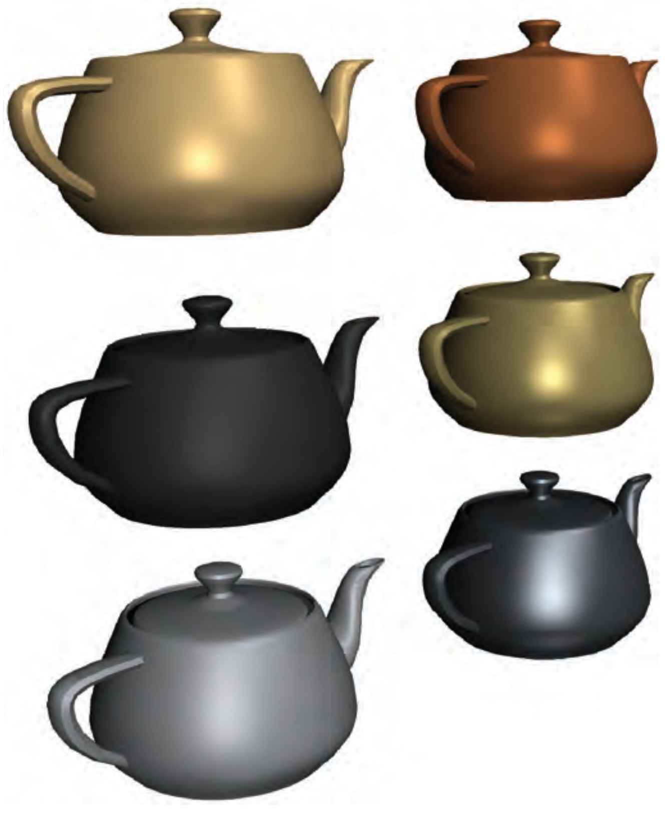
GPUs are designed to process geometry (vertices) and topology (e.g., triangles, lines, polygons, etc.) to 2D images. For example, in the diagram above, the blue polygon is rasterized to pixels (or fragments) after each vertex is processed by an instance of a vertex shader. Each pixel is then processed by a pixel or fragment shader to produce the final image. User-defined textures and variables (Uniform Parameters) can act as input parameters to custom shaders. These parameters can be initialized for each geometric object before it is rendered.

Programmable hardware shaders are written in C-like languages that take advantage of GPU architecture. For instance, GPUs represent four-dimensional vectors as first-class variable types and provide hardware support for efficient vector operations. This is helpful in lighting calculations with typically involved vector operations.

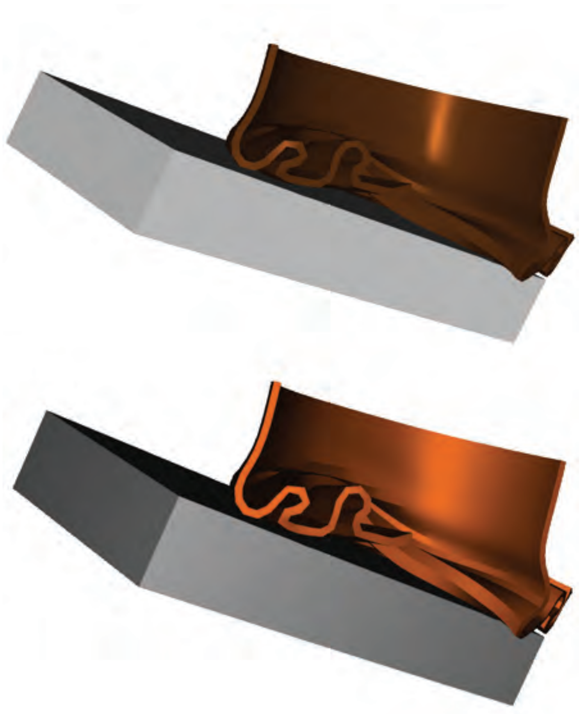
In practice, shaders can be compiled and installed to hardware just before they are used in rendering. The application configures the GPU to use a specific shader and sets its parameters. It then processes geometry and topology to hardware through standard graphics APIs (e.g. OpenGL or DirectX).

SAND2005-3035P
April 2006

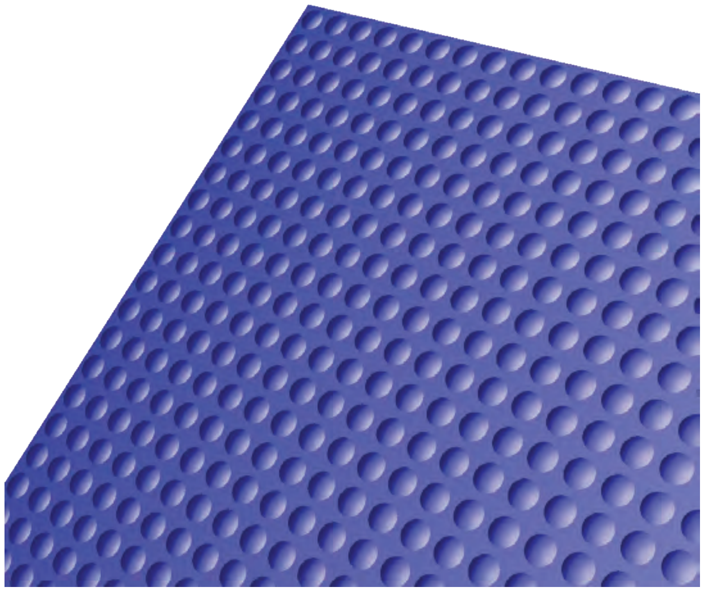
Sandia National Laboratories



Additional realistic effects can help users distinguish between materials. In the composite image above shininess, surface roughness, and color, combine to provide visual cues about material composition and surface finish. Rendered here are CW from top left, 18K Gold, Copper, Polished Brass, Chromed Plastic, Aluminum, and Vinyl.



The images above hint at the realism that can be achieved with custom shaders. The top image results from the standard fixed-functionality pipelines, per-vertex Phong shading. The bottom images results from a custom fragment shader that implements per-pixel shading algorithms that account for microscopic surface features through a phenomenological micro-facet model. This is particularly well-suited for rendering metals.



Procedural techniques can add details not captured in the original computer-aided design (CAD) model. Bump and displacement mapping can be implemented in hardware to create a visual impression of macroscopic surface features. In the image above, dimples that were not in the original geometry are applied to a flat plane in the hardware-rendering step.

For more information, contact
Gary Templet (925) 294-4540