Shading Language Overview

Michael McCool

April 26, 2004

Programmable Graphics Processing Units (GPUs) have become widespread. They now support floating-point computations and general programming models. A shading language is a domain-specific programming language for specifying shading computations. In this section of the course notes, we will review several high-level real-time shading languages for programming modern GPUs. These languages have evolved from academic experiments into necessary tools for real-time graphics.

Historically, the idea of a shading language is often credited to Cook [3]. Pixar's RenderMan shading language was developed shortly thereafter and has become a standard in the offline domain [4, 14, 1, 11]. The RenderMan shading language has strongly influenced the design of modern real-time shading languages, particularly the idea of uniform and varying parameters. However, the RenderMan standard, although originally intended as a hardware API, is no longer used as such, and modern GPU shading languages need to target the hardware architecture of modern GPUs.

The SGI Interactive Shading Language [10] compiles a shader specification to a multipass implementation, but does not generate complex shader kernels, only primitive passes. The Stanford Real-Time Shading Language was one of the first attempts to build a shading language specifically designed for the shading units of graphics processors [12, 7]. It is so far unique in that it supported a single shader for programming both vertex and fragment units.

NVIDIA's Cg [6] language was one of the earlier commercially available shading languages to run on commodity GPU hardware. Although developed by NVIDIA, it supports multiple hardware platforms, and is the only commercial shading language to support both DirectX and OpenGL. It is related to a number of other tools, including CgFX, which is used to specify multipass algorithms.

Microsoft's HLSL for DirectX was originally similar to Cg but has since then diverged. The Direct3D Effects system is similar to CgFX, and permits the specification of multipass algorithms.

The OpenGL standards group recently approved the OpenGL 2.0 Shading Language [5, 13], frequently called GLSL. This shading language will be integrated into the next generation of the OpenGL API and is intended to become the standard-supported way to program GPUs under the OpenGL API.

Brook for GPUs [2] is a scientific computing language for GPUs. It is aimed specifically at general-purpose computation on GPUs, and is built on top of Cg. Brook includes buffer and system management capabilities so it is possible to use Brook to implement a computation without having to use a graphics API.

Sh is an open-source shading language [8, 9]. It takes a metaprogramming approach to the problem and is embedded in C++. It supports close integration between the host application and shaders. Like Brook, it also supports a stream model of computation for general-purpose programming.

References

[1] Anthony A. Apodaca and Larry Gritz. *Advanced RenderMan: Creating CGI for Motion Pictures*. Morgan Kaufmann, 2000.

[2] Ian Buck, Tim Foley, Daniel Horn, Jeremy Sugerman, Kayvon Fatahalian, Mike Houston, and Pat Hanrahan. Brook for GPUs: Stream Computing on Graphics Hardware. *ACM Transactions on Graphics (Proc. ACM SIGGRAPH)*, 23(3), August 2004.

[3] Robert L. Cook. Shade trees. In *Proc. ACM SIGGRAPH*, pages 223–231, July 1984.

[4] Pat Hanrahan and Jim Lawson. A language for shading and lighting calculations. In *Computer Graphics (SIGGRAPH '90 Proceedings)*, pages 289–298, August 1990.

[5] John Kessenich, Dave Baldwin, and Randi Rost. *OpenGL 2.0 Shading Language*, 1.051 edition, February 2003.

[6] William R. Mark, R. Steven Glanville, Kurt Akeley, and Mark J. Kilgard. Cg: A system for programming graphics hardware in a c-like language. *ACM Transactions on Graphics (Proc. ACM SIGGRAPH)*, 22(3):896–907, July 2003.

[7] William R. Mark and Kekoa Proudfoot. Compiling to a VLIW fragment pipeline. In *Graphics Hardware 2001*. SIGGRAPH/Eurographics, April 2001.

[8] Michael McCool, Zheng Qin, and Tiberiu Popa. Shader Metaprogramming. In *Proc. of SIGGRAPH/Eurographics Graphics Hardware*, pages 57–68, September 2002.

[9] Michael McCool, Stefanus Du Toit, Tiberiu Popa, Bryan Chan, and Kevin Moule. Shader Algebra. In *Proc. ACM SIGGRAPH*, August 2004.

[10] Mark S. Peercy, Marc Olano, John Airey, and P. Jeffrey Ungar. Interactive multi-pass programmable shading. In *Proc. SIGGRAPH*, pages 425–432, July 2000.

[11] Pixar. *The RenderMan Interface, version 3.2*, July 2000.

[12] K. Proudfoot, W. R. Mark, P. Hanrahan, and S. Tzvetkov. A real-time procedural shading system for programmable graphics hardware. In *Proc. SIGGRAPH*, August 2001.

[13] Randi J. Rost. *OpenGL Shading Language*. Addison-Wesley, 2004.

[14] Steve Upstill. *The RenderMan companion: A Programmer's Guide to Realistic Computer Graphics*. Addison-Wesley, 1990.