Stanford Real-Time Programmable Shading Project

Kekoa Proudfoot
Stanford University
http://graphics.stanford.edu/projects/shading/

(joint work with Bill Mark, Svetoslav Tzvetkov, and Pat Hanrahan)

Motivation

Real-time shading languages
- Easy-to-use, high-level interface to hardware
- Shader compilation results in platform independence
- Fast exploration of interesting new effects

Multipass rendering is not enough
- Fragment processing is expensive
- Today’s fragment operations are limited: fixed point, simple operators
Multiple computation frequencies

- Constant
- Per Primitive Group
- Per Vertex
- Per Fragment

Evaluated less often
More complex operations
Floating point

Evaluated more often
Simpler operations
Fixed point

Programmable pipeline abstraction

A framework for multiple computation frequencies

Shader Parameters

- Primitive Group Processing
  - e.g. matrix setup
- Vertex Processing
  - e.g. lighting
  - e.g. transforms
- Fragment Processing
  - e.g. texturing

Programmable pipeline
- All stages may be fully programmed

Traditional pipeline
- Fixed but configurable processing
- Fragment programmability enabled by multipass
A shading language

A shading language is the user-level interface to pipeline programmability

Language highlights

- C-like syntax for computations
- Scalar, vector, matrix types and operators
- Automated (but user-controllable) management of computation frequencies
- Support for surface and light shaders