CS 563 Advanced Topics in Computer Graphics
Intro to Vertex and Pixel Shaders

by Matthew Maziarz

Outline

- History before Shaders
  - Pre-Hardware T/L
  - Hardware T/L
- Vertex Shaders
- Pixel Shaders
- Extras
- References

Before Shaders

- Prior to 1999 all Transformation and Lighting was done by the CPU
- This caused the CPU to do almost all the work
- Could use assembly to make the card do more of the work

Before Shaders countinued

- 1999 cards introduced Hardware T/L
  - This moved the transformation and lighting to the card which alleviated some work from the CPU
  - The problem was this was a fixed function pipeline
  - Once you sent it to the card you had no control
  - Forced programmers to use basic Gouraud/Phong for lighting because it was the only model supported

Shader

- "Shading is the assignment of colors - or more specifically - outoging radiance, to points on a surface“ [pg 5 Real Time Shading]

- Vertex and Pixel shaders do much more, including manipulation and movement of vertices.

General Shader

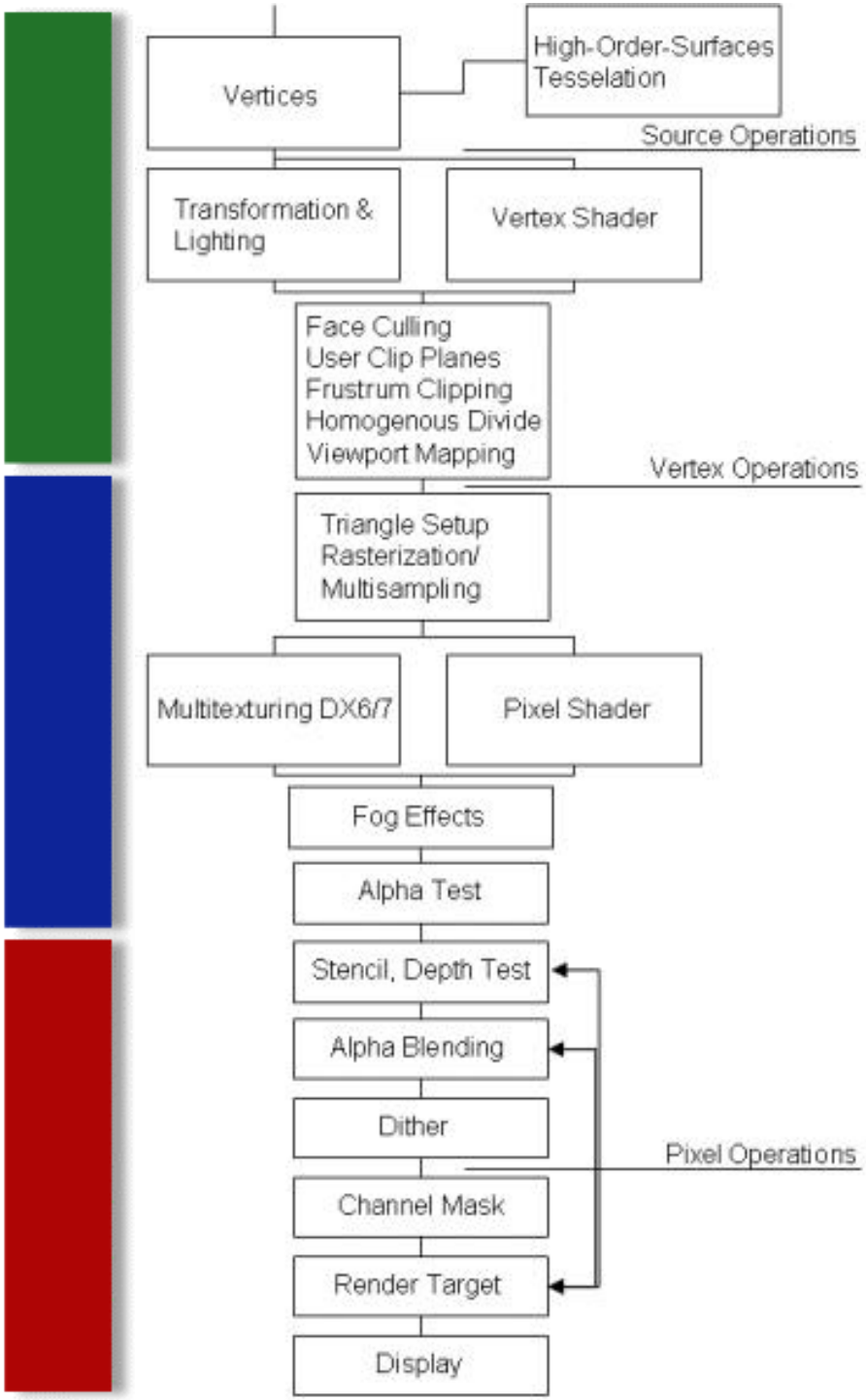
- Both Pixel and Vertex Shader can be used if API supports even if card does not. It will run on CPU (slow)
- Both the fixed pipeline and programmable can be used, just not in parrallel
- Limited number of Instructions (keep getting raised)

General continued

- Must be compiled, usually at run-time
- Nvidia
  - Vertex Shader -> Texture Shader -> Register Combiner
  - Texture Shader not really programmable, just choose some options
- Direct X
  - Vertex Shader -> Pixel Shader

- Shader languages are inherently different from most programming languages They are based on a data flow computational model, in other words, computation is dependant on that data that comes in.

Direct 3D pipeline



http://www.gamedev.net/columns/hardcore/dxshader1/page2.asp

Vertex Shader

- 16 input registers.
- 9 output registers for GeForce cards and 11 for Radeon cards.
- 96 constant registers for GeForce cards and 192 for Radeon cards.
- 12 temporary registers.
- 1 address register (for vertex shader version vs.2.0).

Vertex Shader continued

- Reasons why
  - Procedural Geometry (cloth)
  - Particle Systems
  - Advanced Animation Interpolation
  - Lens Effects
  - Newer Lighting effects
- Can Change
  - Position, Color, Size, Texture Coordinates

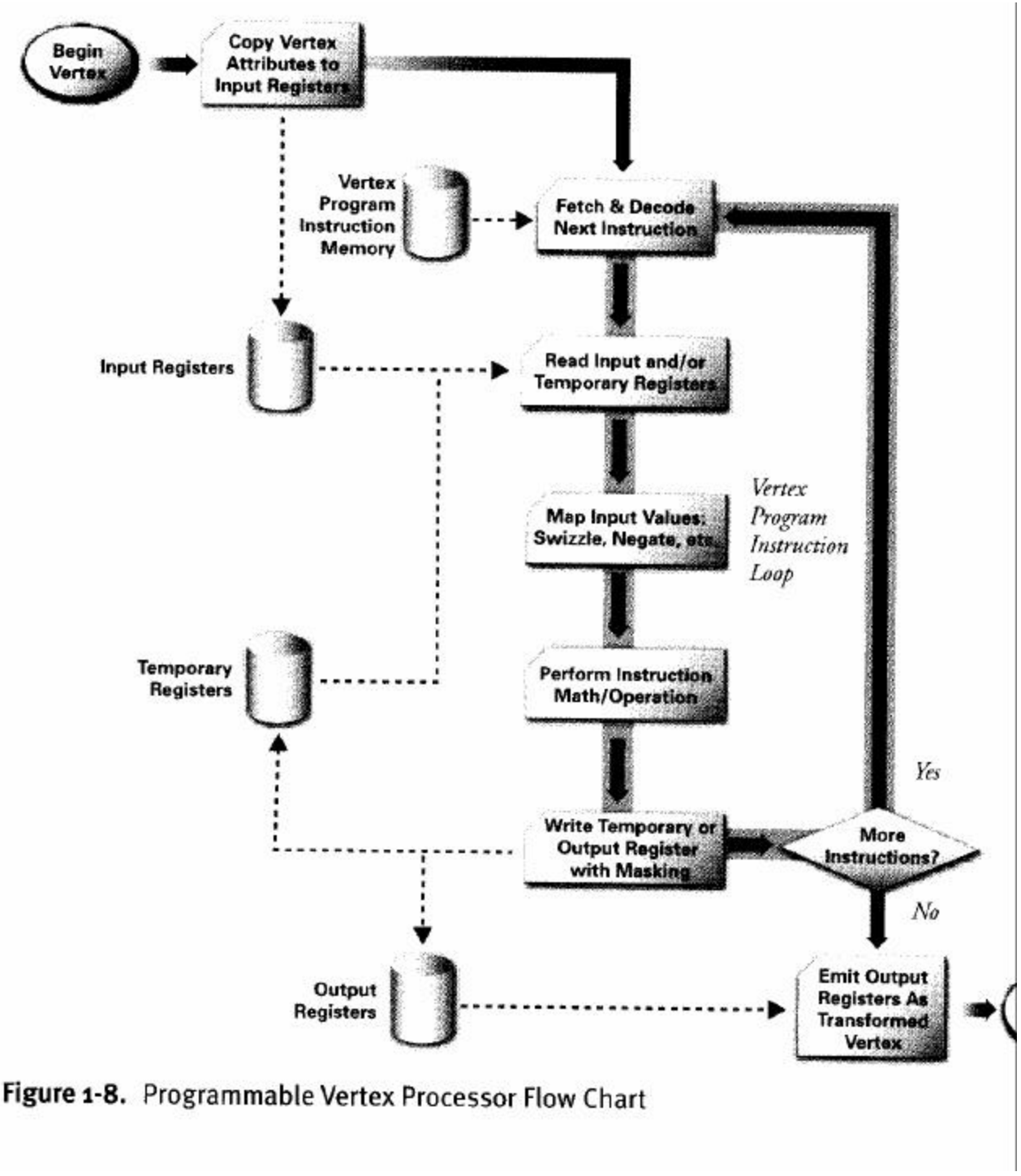


Figure 1-8. Programmable Vertex Processor Flow Chart

Pg 18 The CG Tutorial

Vertex Shader Continued

- Does
  - Can load and unload different shaders to only run some shaders on certain streams of vertices
  - If card supports multiple shaders they can run in parallel on multiple processing units
  - Each vertex take same amount of time to pass through shader

Vertex Shader Continued

- Don’ts
  - Vertices cannot be created or destroyed
    - Can be moved off screen
  - Each vertex independent of others so vertices cannot share information
    - This is what allows you to run in parallel if hardward supports it
  - No loops or GoTo commands (yet)
    - Certain hardware are now trying to implement this

Pixel Shader

- 8 constant registers.
- 4 texture registers (6 in DirectX pixel shader version ps.1.4).
- 2 temporary registers (6 in DirectX pixel shader version ps.1.4).
- 2 color registers.

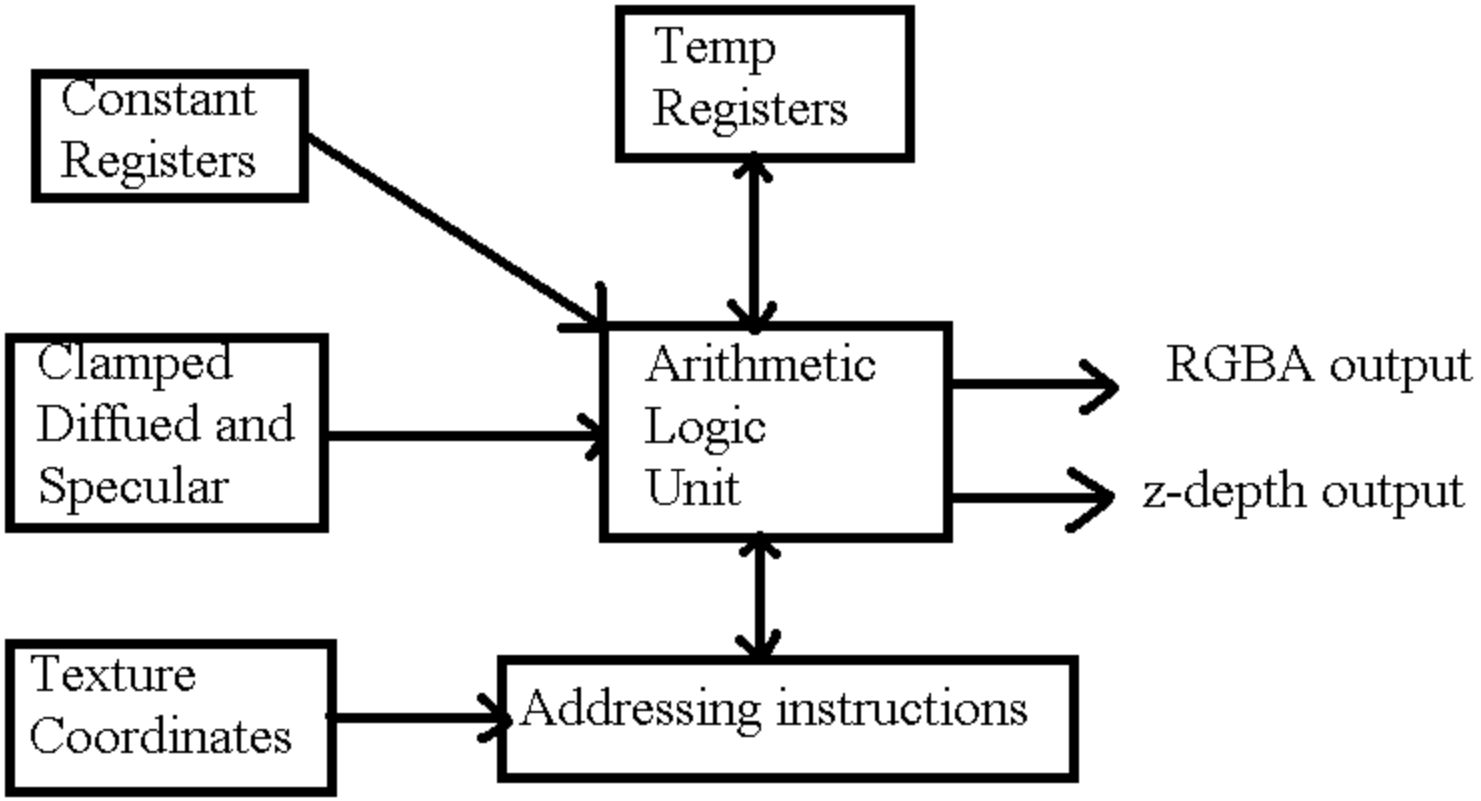
Pixel Shader Continued

- Reasons Why
  - Single Pass per-pixel lighting (true phong)
  - Anisotropic Lighting
  - Cell/Toon/Non-Photorealistic rendering
  - Volumetric effects
  - Procedural textures
  - Horizon (self-shadowing bump) maps
- Can Change
  - Perform math on texture coordinates
  - Use texture lookups to modify other textures

Figure 1-9. Programmable Fragment Processor Flow Chart

Cg still allows you to write fragment programs that

Pixel Shader continued



- Pg 221 Real Time Rendering

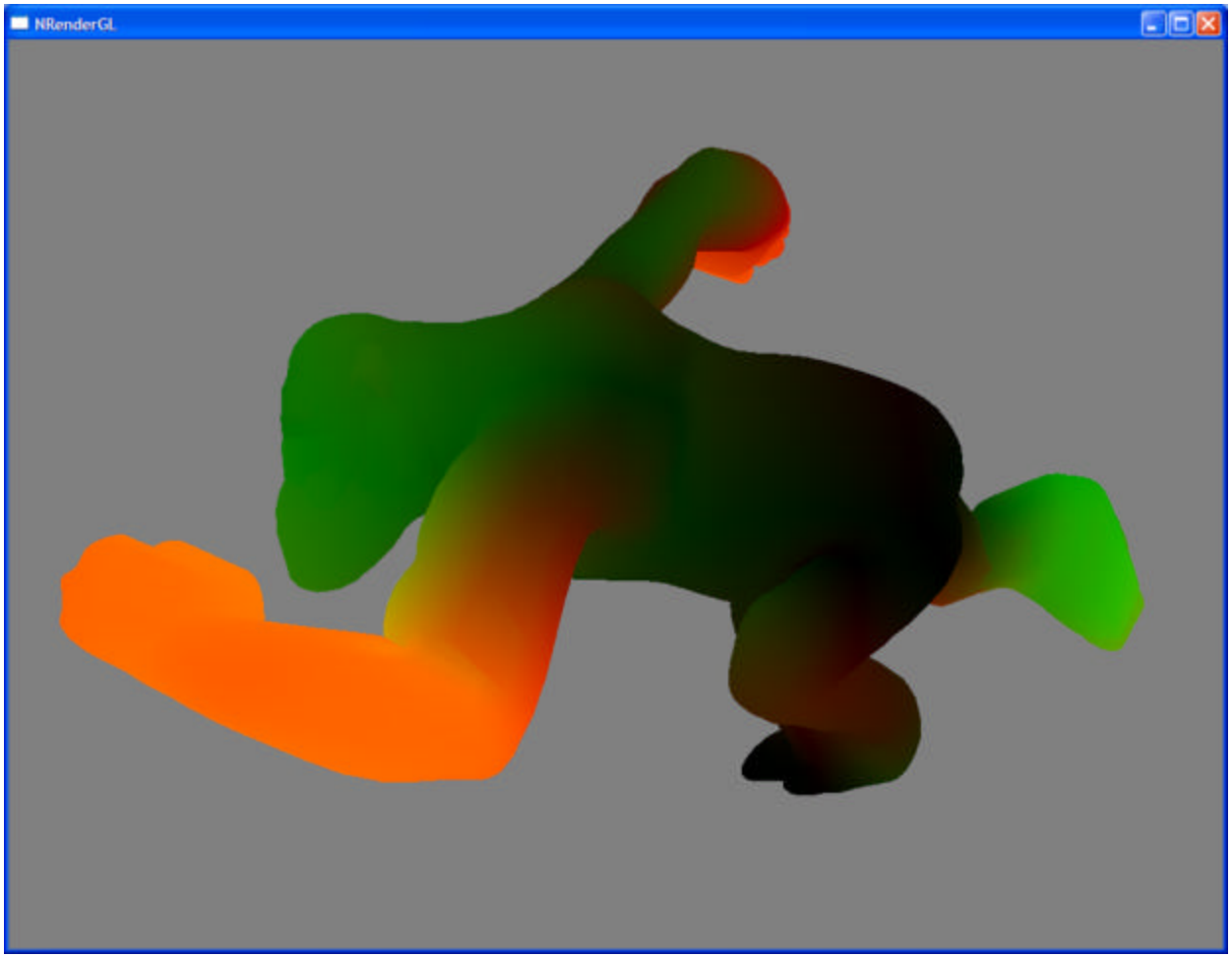
Pixel Shader Continued

- Addressing Instructions
  - Used to look up values in a texture
  - Depending on instruction can treat the coordinate
    - As standard lookup
    - Vector
    - Part of matrix
  - You can also kill fragments
  - Addressing Instruction do not perform operation but they set up the data in a specific form

Extras

- Courtesy of *Stupid OpenGL Shader Tricks* by Simon Green

Extras



Extras

Links

- Here are some recommended links for beginning with shaders
- http://nehe.gamedev.net/data/lessons/lesson.asp?lesson=47
  - A simple pseudo water vertex shader, also walksthrough the opengl code to load and run shader
- Shader Programming by Wolfgang Engel http://www.gamedev.net/columns/hardcore/dxshader1/
  - Beginner Direct X vertex and pixel shader paper
- Best place would be NVidia's CG Toolkit
  - A little more advanced but a ton of examples

References

- Akenine-Moller, Tomas. “Real-Time Rendering”. Second Edition. 2002. AK Peters, Ltd.
- Fernando, Randima and Kilgard, Mark. “The CG Tutorial”. 2003. Adddison-Wesley
- Olano, M. Hart, J. Heidrich, W. and McCool, M. "Real-Time Shading" A K Peters Publishers, 2002
- http://nehe.gamedev.net/data/lessons/lesson.asp?lesson=47
- Shader Programming by Wolfgang Engel http://www.gamedev.net/columns/hardcore/dxshader1/
- http://www.ultimategameprogramming.com/articles/VertexPixelShaderIntro3.php
- http://www.devmaster.net/articles/shaders/