Automatic Shader Level of Detail

Marc Olano – UMBC
Bob Kuehne – SGI
Maryann Simmons – SGI
What is Shading

- Ultimate control of appearance
- Programmable
  - Arbitrary computation
- Procedural
  - Simple procedures
  - High-level language
Interactive Rendering

- Illusion of Presence
  - 10 – 30 – 60 frames per second
  - Immediate response
  - Simple appearance

- Shading HW
  - ISL, GLslang, Cg, HLSL
Uses for Real-Time Shading

• More realistic appearance
  – Automotive styling

• Visualization
  – Data fields on surfaces

• Non-realistic appearance
  – Games, Illustration
Non-Real Time / Real Time

- Not Real-Time
  - General CPU
  - Seconds to hours per frame
  - Thousand line shaders
  - “Unlimited” computation, texture, memory, ...
  - [Cook84] [Perlin85] [Hanrahan90]

- Real-Time
  - Graphics HW
  - Tens of frames per second
  - Thousand instruction shaders
  - Limited computation, texture, memory, ...
  - [Rhoades92] [Olano98] [Peercy00] [Proudfoot01] [Mark02]
Stretching the Limits

• Want for shading
  – Expensive shaders: good up close
  – Real-time performance
  – Lots of objects

• Similar to geometric models
  – Detailed models: good up close
  – Real-time performance
  – Lots of objects
Geometric Level of Detail

• Multiple representations of object
• Differing complexity
• Choose based on distance, screen size, rendering budget, …
  – [Clark76] [Funkhouser93]

Image Removed: Figure 3.3

Shader Level of Detail

- Multiple representations for shader
  - [Goldman97] [Apodaca00] [Olano02]
- Differing rendering cost
- Similar considerations for level

Image Removed: Close ups from Figure 7
Dan B Goldman, “Fake Fur Rendering,” SIGGRAPH 97
Geometric Simplification

- Start with complex model

Image Removed: Figure 5d
Hughes Hoppe, “Progressive Meshes,” SIGGRAPH ‘96
Geometric Simplification

- **Automatically** create new models
  - Collapse, merge, volumetric, ...
- Separate models or progressive mesh [Hoppe96]

Image Removed: Figure 5
Hughes Hoppe, “Progressive Meshes,” SIGGRAPH ‘96
Geometric Simplification

• Evaluate possible collapse costs
• Choose least-cost remaining
  – Top of heap / full sort not necessary
• Re-evaluate changed costs
  – Usually local
  – Collapse moves monotonically toward goal, no backtracking
Shader Simplification

• Start with complex shader
  – Typically built in layers [Apodaca00]
Shader Simplification

- LOD building blocks [Olano01]
  - After [Cook84] [Abram90]
  - Bump, BRDF, Fresnel, …
Automatic Simplification

- Goal
- Simplification operation
  - Guaranteed convergence
- Cost function
Simplification Goal

- Reduce texture accesses
  - Direct benefit on most hardware
  - Indirectly reduces instruction count
  - Indirectly reduces active textures
Simplification vs Optimization

- **Simplification**
  - Rewrite to reduce cost
  - Allow possible loss of fidelity

- **Optimization**
  - Rewrite to reduce cost
  - Must produce identical result
Simplification operations

• Lossless
  – Identical results: optimization

• Resolution-specific lossless
  – Resampling errors only

• Lossy
  – Approximation errors
Simplification operations

- Texture Removal
- Texture Collapse
Texture Removal

• Replace texture with non-texture approximation
  – Lossy
  – Cost = RMS error at MIP level
• Demonstrated
  – Replace texture with constant
• Future
  – Replace environment w/ Phong
  – Replace texture with built-in operation
Texture Collapse

- Replace static sequence of operations including at least one texture with one new texture
- Similar to specializing shaders [Guenter95]
- Demonstrated
  - Lossless: single texture resolution
- Future
  - Resolution-specific lossless
  - Choose new size & resample
## Results

<table>
<thead>
<tr>
<th>Access</th>
<th>Active</th>
<th>Reduction</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>45</td>
<td>14</td>
<td>0.00</td>
<td>1.0</td>
</tr>
<tr>
<td>23</td>
<td>11</td>
<td>0.49</td>
<td>1.8</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td>0.80</td>
<td>1.9</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1.00</td>
<td>2.3</td>
</tr>
</tbody>
</table>
Observations

- RMS error at MIP level
  - Measure of error and frequency
- Possible error amplification
  - Solvable / not problem for most shaders
Observations

• Early antialiasing
  – Similar to automatic antialiasing [Perlin98] [Heidrich98]
  – Modify for NPR?

• Collapse enables Removal
System Interface

- SGI OpenGL Shader
- Source: ISL
- Simplify if autoLOD present
- Output: Single compiled shader
  
  if (autoLOD < threshold1)
  original_shader
  else if (autoLOD < threshold2)
  simplified_once
  else
  simplified_twice
Conclusions

• Shader simplification
  – Possible, practical, useful
  – Necessary?

• General framework
  – modeled on geometric simplification

• Implementation
  – modeled as lossy compiler optimization
See Also

- Sketch: Per-Pixel Smooth Shader Level of Detail, Maryann Simmons and Dave Shreiner
  - Wednesday 10:30 Session
  - Convention Center Room 30 A-D
Future work

• Track error amplification
• Extend existing operations
• Consider other costs & goals
  – Reduce instructions: replace with texture
• Generalize for NPR
  – User-provided operations?
• Couple with geometric LOD