Automated Reprojection-Based Pixel Shader Optimization

Pitchaya Sitthi-amorn, Jason Lawrence
(Presenter)
University of Virginia

Lei Yang, Pedro V. Sander
Hong Kong University of Science and Technology

Diego Nehab
Microsoft Research

Jiahe Xi
Hong Kong University of Science and Technology

SIGGRAPHASIA2008
Motivation

Original Scene with expensive pixel shader
Motivation

Optimized
Talk Outline

• Motivation
• Background
• Error/Performance Models
• Results
Related Work: Code Simplification

- Replace subexpressions with simpler expressions
- Automatic shader level of detail [Olano et al. 2003]
- User-configurable automatic simplification [Pellacini 2005]
Related Work: Dynamic Resizing

- Render scene to lo-res off-screen buffer and upsample to target resolution
- InfiniteReality system [Montrym et al. 1997]
- Geometry-aware resizing [Yang et al. 2008]
Related Work: Temporal Reprojection

- Reuse partial shading calculations across consecutive frames

frame n-1  frame n  green = mutually visible
red = occluded
Background: Temporal Reprojection

Frame n-1

- framebuffer
- payload
- depth

Shading Cache

Frame n
Background: Temporal Reprojection

Frame n-1

- framebuffer
- payload
- depth

Shading Cache

Frame n

- framebuffer
Background: Temporal Reprojection

Frame n-1

- framebuffer
- payload
- depth

Frame n

- framebuffer

Shading Cache
Background: Temporal Reprojection

Frame n-1
- framebuffer
- payload
- depth

Frame n
- framebuffer
- payload
- depth

Shading Cache
• Cached entries will become stale due to changes in shader inputs and from resampling error
• Explicitly recompute cached values within a pre-defined refresh period ($\Delta n$)
Explicitly recompute cached values within a pre-defined refresh period ($\Delta n$)
Goal of this Work

- Automate allocation of a shading cache
Goal of this Work

• Identify a node \( f_m \) and refresh period \( \Delta n \) that minimizes render time \( r(f_m, \Delta n) \) and approximation error \( \varepsilon(f_m, \Delta n) \)
System Overview
Talk Outline

• Motivation
• Background
• Error/Performance Models
• Results
Hit Rate

\[ \gamma(\Delta n) = \frac{\#\text{hit}}{\#\text{hit} + \#\text{miss}} = \mu(1 - 1/\Delta n) \]

- green = reused
- blue = explicitly refresh
- red = occluded
- \#hit = \#green
- \#miss = \#blue + \#red
Error Model

\[ \hat{e}(f_m, \Delta n) \]

average L² distance between color returned by the original shader and the shader modified to cache \( f_m \) at refresh period \( \Delta n \)

\[ \hat{e}(f_m, \Delta n) = \alpha_m \left(1 - e^{-\lambda_m (\Delta n - 1)}\right) \]

(Parameters to be fitted)

Magnitude of error \hspace{1cm} Rate of decay
Error Model (Validation)
Performance Model

\[ \hat{r}(f_m, \Delta n) \]

average time required to shade a single pixel when caching node \( f_m \) at refresh period \( \Delta n \)

\[
\begin{pmatrix}
  h_1 & m_1 & 1 \\
  h_2 & m_2 & 1 \\
  \vdots \\
  h_L & m_L & 1
\end{pmatrix}
\begin{pmatrix}
  E[\text{time\_hit}(f_m)] \\
  E[\text{time\_miss}(f_m)] \\
  E[\text{time\_overhead}]
\end{pmatrix}
= 
\begin{pmatrix}
  t_1 \\
  t_2 \\
  \vdots \\
  t_L
\end{pmatrix}
\]

\[ \hat{r}(f_m, \Delta n) = \gamma(\Delta n)E[\text{time\_hit}(f_m)] + (1 - \gamma(\Delta n))E[\text{time\_miss}(f_m)] \]
Performance Model (Validation)

Marble Shader (NVIDIA 8600GT)

Average Render Time (ms) vs. Refresh Period $\Delta n$

- Measured
- Predicted
Talk Outline

• Motivation
• Background
• Error/Performance Models
• Results
Test Scenes

Marble shader

procedural noise with Blinn-Phong specular layer (75K triangles)

Trashcan shader

supersampled (25) environment map (15K triangles)

Ambient occlusion shader

Real-time ambient occlusion approximation (36K triangles)
## Statistics

<table>
<thead>
<tr>
<th>Input Shader</th>
<th>Marble</th>
<th>Trashcan</th>
<th>Ambient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instructions</td>
<td>419</td>
<td>367</td>
<td>370</td>
</tr>
<tr>
<td>Cacheable Nodes</td>
<td>114</td>
<td>105</td>
<td>37</td>
</tr>
<tr>
<td>Precomputation Time</td>
<td>1.5h</td>
<td>1.2h</td>
<td>9.5h</td>
</tr>
</tbody>
</table>
Sample Rendering Session

(video)
Marble Shader
Marble Shader
Marble Shader

Cluster A

Cluster B

Cluster C

Cluster D

Average Pixel Error

Average Render Time (s)

Δn=2 • 

Δn=25 ▲

Δn=50 ▼
Marble Shader
Marble Shader
Trashcan Shader
Trashcan Shader

Original (video) 59.2
Ambient Occlusion Shader

Graph showing the relationship between Average Pixel Error and Average Render Time (ms) for different Δn values (Δn=2, Δn=25, Δn=50). The graph illustrates how the error decreases as the render time increases.
Ambient Occlusion Shader

(video) 1.8

Original
Comparison

Our approach (2.8x) Original (1x)
Summary

• A method for automatically allocating a pixel cache
• Models of the error and performance of different caching decisions
• Interactive profiler allows user to set the expected error and automatically returns the best node and refresh period
Future Work

• Online analysis and allocation
• Caching multiple nodes
• A more compact shading cache
• Alternative parameterization strategies
• Automatic level of detail
Acknowledgements

- AMD/ATI for test scenes
- NVIDIA for funding
- NSF CAREER award CCF-0747220
- RGC CERG grant #619207
Thank You