SIGGRAPH ASIA 2011
HONG KONG
The 4th ACM SIGGRAPH Conference and Exhibition on Computer Graphics and Interactive Techniques in Asia

Conference: 12-15 December | Exhibition: 13-15 December
Hong Kong Convention and Exhibition Centre

www.SIGGRAPH.org/ASIA2011
Genetic Programming for Shader Simplification

Pitchaya Sitthi-Amorn
Westley Weimer
Nicholas Modly
Jason Lawrence

University of Virginia
Motivation

NVIDIA and XYZRGB Inc.
Motivation

GPU Computing Power

source: http://www.anandtech.com
Sequence of shaders that trade accuracy for speed:

- Original
- x1.4 faster
- x2.2
- x6.2
- x4.3
Prior Work: Simplification

- Automatic Shader Level of Detail [Olano et al. 2003]
  - Restricted to simplifying textures
- User-configurable Automatic Shader Simplification [Pellacini 2005]
  - Set of local code simplifications + greedy search
Prior Work: GP-Based Code Repair

- Automatic program repair with evolutionary computation [Weimer et al. 2009, 2010]
  - Use GP to automatically fix bugs in large programs
  - Optimizes single discrete value (number of test cases passed) as opposed to multi-objective (rendering time and visual fidelity)
Problem Statement

```c
float cosi = cos
float R = 1.0f;
float n12 = 1.0f
float sint = n12
if (sint < 1.0f)
    float cost =
    float r_ortho
        / (cosi +
    float r_par
        / (cost +
    R=(r_ortho *
```
Our Approach

1. **Define search space**: sequence of mutations applied to input shader (delete, swap, insert, crossover, etc.).
2. **Define objective function**: L2 and SSIM over representative sequence.
3. **Perform search** using Genetic Programming.
1. **Define search space:** sequence of mutations applied to input shader (delete, swap, insert, crossover, etc.).

2. **Define objective function:** L2 and SSIM over representative sequence.

3. **Perform search** using Genetic Programming.
float cosi = cos
float R = 1.0f;
float n12 = 1.0f
float sint = n12
if (sint < 1.0f)
    float cost =
    float r_ortho
        / (cosi +
    float r_par
        / (cost +
    R=(r_ortho *

source code

abstract syntax tree (AST)
Mutation Operators

original

variant

delete
Mutation Operators

original

variant

insert
Mutation Operators

original

variant

swap
Mutation Operators

original

replace with average

variant

\[ E[ \quad ] \]
Mutation Operators

original

variant

delete
insert
swap
repace w/ average
Our Approach

1. **Define search space:** sequence of mutations applied to input shader (deletion, swap, insert, crossover, etc.)

2. **Define objective function:** L2 and SSIM over representative sequence.

3. **Perform search** using Genetic Programming.
Error Measurement

original

variant

Average Euclidean
or
SSIM [Wang et al. 2004]
(could use others)
Measuring Error and Performance

float4 main_ps(float3 Pos, float3 Normal)
{
  .
  .
  .
  return colorOut;
}

pos normal
Measuring Error and Performance
Measuring Error and Performance

Frame 2
Fast Error Measurement
Measuring Error and Performance

original

variant

www.SIGGRAPH.org/ASIA2011
Our Approach

1. **Define search space:** sequence of mutations applied to input shader (deletion, swap, insert, crossover, etc.)
2. **Define objective function:** L2 and SSIM over representative sequence.
3. **Perform search** using Genetic Programming.
Multi-Objective Genetic Programming

Initialize

Rendering Time

Error
Multi-Objective Genetic Programming

Initialize

Error

Rendering Time
Multi-Objective Genetic Programming

Initialise Sort

Rendering Time Error

Pareto Frontier
Multi-Objective Genetic Programming

Rendering Time vs. Error

Pareto Frontier

variants
Multi-Objective Genetic Programming

Initialize   Sort

Rendering Time   Error

rank 0
rank 1
rank 2
rank 3
Multi-Objective Genetic Programming

Initialize  Sort  Select Pairs

Error!

Rendering Time

rank 0  rank 1  rank 2  rank 3

Error
Multi-Objective Genetic Programming

Initialize | Sort | Select Pairs | Crossover+Mutate

Rendering Time

Error

rank 0

rank 1

rank 2

rank 3
Multi-Objective Genetic Programming

Initialize → Sort → Select Pairs → Crossover+Mutate → Evaluate → Select

Rendering Time vs. Error

www.SIGGRAPH.org/ASIA2011
Multi-Objective Genetic Programming
Multi-Objective Genetic Programming
<table>
<thead>
<tr>
<th>Evaluation</th>
<th>Marble</th>
<th>Trashcan</th>
<th>Human Head</th>
</tr>
</thead>
<tbody>
<tr>
<td>procedural noise with</td>
<td>supersampled (5x5) environment map</td>
<td>texture-space diffusion for subsurface scattering in</td>
<td></td>
</tr>
<tr>
<td>Blinn–Phong surface</td>
<td>(15K triangles)</td>
<td>human skin</td>
<td></td>
</tr>
<tr>
<td>(75K triangles)</td>
<td></td>
<td>(300K triangles)</td>
<td></td>
</tr>
</tbody>
</table>

* Paper and supplemental doc contain additional results on SSAO and variance shadow map.
Marble Shader

![Graph showing Shader Variants and Pareto Frontier with corresponding images of rendered results with different error and rendering times.]

- Original, 0.311ms
- Error=1.5e-2, 0.22ms
- Error=3.4e-2, 0.14ms
- Error=7.0e-2, 0.07ms

www.SIGGRAPH.org/ASIA2011
Trashcan Shader

![Shader Variants Pareto Frontier]

- Original, 3.68ms
- Error=1.2e-2, 1.09ms
- Error=2.5e-3, 1.93ms
- Error=8.2e-2, 0.98ms

www.SIGGRAPH.org/ASIA2011
Comparison to Pellacini [2005]

Marble Shader

- Our GP Algorithm
- Pellacini’s Algorithm

Rendering Time (ms)

Error ($L^2$ in RGB)

- Our Technique
- [Pellacini 2005]

- Error=1.5e-2, 0.26ms
- Error=3.7e-2, 0.25ms

- Error=3.4e-2, 0.14ms
- Error=1.0e-1, 0.14ms
Comparison to Pellacini [2005]

Marble Shader

Rendering Time (ms)

Error (1-SSIM)

Our GP Algorithm
Pellacini’s Algorithm

Our Technique
[Pellacini 2005]

Error=2.7e-2, 0.26ms
Error=1.9e-1, 0.29ms

Error=1.5e-1, 0.14ms
Error=3.2e-1, 0.14ms

www.SIGGRAPH.org/ASIA2011
Conclusion and Future Work

• Genetic Programming for shader simplification
  • Handles multi-pass effects and perceptual error metrics
  • Outperforms previous state-of-the-art due to exploration of a wider search space

• Unified simplification of vertex, geometry, and fragment
• Further reduce preprocessing time
• Other problems: shader synthesis, antialiasing, etc.
Acknowledgements

• NVidia and XYZRGB Inc. for “Human Head” scene
• NSF CAREER Awards #CCF-0747220, #CCF-0954024
• NSF grant #CCF-0905373
• AFOSR grants #FA8750-11-2-0039, #FA9550-07-1-0532
• DARPA grant #FA8650-10-C-7089
• CAVGRAPH and SIGGRAPH reviewers
Questions?
Some Statistics

<table>
<thead>
<tr>
<th>Shader</th>
<th>Lines of Source</th>
<th>Shader Variants (generated+unique)</th>
<th>Shader Variants (on frontier)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marble</td>
<td>120</td>
<td>1,725</td>
<td>294</td>
</tr>
<tr>
<td>Trashcan</td>
<td>127</td>
<td>1,461</td>
<td>119</td>
</tr>
<tr>
<td>Human Head</td>
<td>962</td>
<td>2,960</td>
<td>88</td>
</tr>
</tbody>
</table>
# Distribution of Mutation Operators

<table>
<thead>
<tr>
<th>Shader</th>
<th>Insertion</th>
<th>Replace w/ Average</th>
<th>Swap</th>
<th>Delete</th>
<th>Cross Over</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marble</td>
<td>235</td>
<td>477</td>
<td>68</td>
<td>194</td>
<td>941</td>
</tr>
<tr>
<td>Trashcan</td>
<td>202</td>
<td>331</td>
<td>100</td>
<td>111</td>
<td>393</td>
</tr>
<tr>
<td>Human Head</td>
<td>3,691</td>
<td>4,269</td>
<td>1,961</td>
<td>1,546</td>
<td>2,933</td>
</tr>
</tbody>
</table>
## Processing Times

<table>
<thead>
<tr>
<th>Shader</th>
<th>Compiling</th>
<th>Error + Perf Eval</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marble</td>
<td>0.7</td>
<td>0.3</td>
<td>1</td>
</tr>
<tr>
<td>Trashcan</td>
<td>0.2</td>
<td>0.7</td>
<td>1</td>
</tr>
<tr>
<td>Human Head</td>
<td>2</td>
<td>3</td>
<td>5</td>
</tr>
</tbody>
</table>

All times in hours