Surveying Real-Time Rendering Algorithms Beyond Programmable Shading

David Luebke
NVIDIA Research
The continuum “Beyond Programmable Shading”

“Just” programmable shading: DX, OGL
The continuum “Beyond Programmable Shading”

“Just” programmable shading: DX, OGL

“Pure” compute-based graphics: CUDA, OptiX
The continuum “Beyond Programmable Shading”

“Just” programmable shading: DX, OGL

Interesting middle ground!

“Pure” compute-based graphics: CUDA, OptiX
3 Hybrid Graphics/Compute Patterns

- Render – Analyze – Render
- Render to alternate data structure
- Rasterize to invoke spatially localized compute
Render – Analyze – Render

- Stochastic Transparency
- Sample Distribution Shadow Maps
Stochastic Transparency

Eric Enderton, Erik Sintorn, Peter Shirley, David Luebke, 2010 ACM Symposium on Interactive 3D Graphics & Games
Stochastic Transparency

Screen-door transparency
screen door transparency

Stochastic Transparency
Stochastic Transparency

Screen-door + multi-sampling

(“Alpha-to-coverage”)
Stochastic Transparency

Screen-door + multi-sampling + random masks.

• Correct on average, in all cases
  – Foliage, Smoke, Hair, Glass. CAD. Shadows.
  – Mixed together

• Fast
  – One order-independent pass, one MSAA z-buffer

• But noisy
  → More samples
  → More algorithms
(Reference)
Alg 1. Basic
8 spp
Alg 1. Basic

16 spp
Alg 1. Basic

32 spp
Alg 1. Basic

64 spp
Alg 1. Basic

512 spp
8 spp

Alg 1. Basic
“Depth-based algorithm”, 8 spp
Analyze depth and accumulated total alpha / pixel
Stochastic Transparency

15,000 hairs
6,000 grass cards
shadows

12 fps 1024x768 8x MSAA
Stochastic Transparency

Scales with MSAA

Better GPU → Better image

Turns OIT into a Moore’s Law problem
Sample Distribution Shadow Maps

Andrew Lauritzen, Marco Salvi, Aaron Lefohn,
Advances in Real-Time Rendering in 3D Graphics and Games, SIGGRAPH 2010 Courses

http://visual-computing.intel-research.net/art/publications/sdsm/
The Problem with Shadow Maps

Scene from Left 4 Dead 2, courtesy of Valve Corporation
Cascaded Shadow Maps

Scene from Left 4 Dead 2, courtesy of Valve Corporation
Seen from Light Space
PSSM Partitions in Light Space
Sample Distribution Shadow Maps

- Analyze the shadow sample distribution
  - Find tight Z min/max
  - Partition Z range logarithmically – fully automatic!

- Compute tight light-space bounds per partition
  - Axis-aligned bounding box on view samples
  - Greatly increases useful shadow resolution

Beyond Programmable Shading, SIGGRAPH 2010

7/29/2010
SDSM Partitions in Light Space
Light-Space Bounds for Partitions
Sample Distribution Shadow Maps

Scene from Left 4 Dead 2, courtesy of Valve Corporation
Compare to Cascaded Shadow Maps

Scene from Left 4 Dead 2, courtesy of Valve Corporation
Render to Data Structure

• Alias-Free Shadow Maps

• Append-Consum OIT

• GPU Progressive Photon Mapping
Alias-Free Shadow Maps


See related publications (Irregular Z-buffer and Alias-Free Shadow Maps) in speaker notes.
The Problem With Shadow Maps: Multiple View Samples per Light Sample

Scene rendered as seen from camera

Depth of closest fragments are rendered to shadow map texel

Scene rendered as seen from light
Shadow Map Data Structure:
Compact List of View Samples per Light Texel

Transform and project view-samples into lightspace, then store in a compact datastructure with a list per lightspace texel
Using the Shadow Map:
Test all light-space triangles against view samples

Render all geometry (conservatively) from lights point of view and for each generated fragment, test all view-samples in that list against the triangle and set the corresponding bit in the output if occluding.
Using the Shadow Map:
Look up each view-space sample as needed

Finally, use the result from the previous step and the SM datastructure in a fullscreen pass over the camera-view image to find shadowing information for each pixel
Append-Consume OIT

Real-Time Concurrent Linked List Construction on the GPU, Yang, Hensley, Grün, and Thibieroz, AMD

EGSR 2010 paper and presented in Advances in Real-Time Rendering in 3D Graphics and Games, SIGGRAPH 2010 Courses
The Problem with Transparency

Sorting is hard!

With Sorting

No Sorting

Skeleton hidden

Arm appears in front of body
Solution: Sort Fragments Per Pixel

• Store linked list of fragments per pixel
  – Each list record stores color, alpha, depth
  – Thread linked lists through DirectCompute UAV

• A “resolve” pass sorts then blends fragments
Create Linked List

**Start Offset Buffer**

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

**Fragment and Link Buffer**

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Render Target
Create Linked List

<table>
<thead>
<tr>
<th>-1</th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

0.87
-1
Create Linked List

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>2</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

0.87 | 0.89 | 0.90
-1   | -1   | -1
Create Linked List

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>-1</td>
<td></td>
<td>3</td>
<td>4</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td></td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td></td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>2</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.87</td>
<td>0.89</td>
<td>0.90</td>
<td>0.65</td>
<td>0.65</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
<td></td>
</tr>
</tbody>
</table>
Create Linked List

```
<table>
<thead>
<tr>
<th>-1</th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>5</td>
<td>4</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>-1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>0.87</th>
<th>0.89</th>
<th>0.90</th>
<th>0.65</th>
<th>0.65</th>
<th>0.71</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>
```

Diagram showing the creation of a linked list with nodes containing the values from the table.
Render Fragments

Walk the list and store in temporary array

<table>
<thead>
<tr>
<th></th>
<th>0.71</th>
<th>0.65</th>
<th>0.87</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>0.87</td>
<td>-1</td>
<td>0.71</td>
</tr>
<tr>
<td>-1</td>
<td>0.89</td>
<td>-1</td>
<td>0.65</td>
</tr>
<tr>
<td>-1</td>
<td>0.90</td>
<td>-1</td>
<td>0.65</td>
</tr>
<tr>
<td>-1</td>
<td>0.65</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>-1</td>
<td>0.65</td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td>0.71</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Render Fragments

Sort temporary array
Blend colors and write out
GPU Progressive Photon Mapping

Pre-publication preview
Toshiya Hachisuka & Henrik Wann Jensen, UC San Diego
Progressive Photon Mapping

- [Hachisuka et al. 08, 09]
Key Idea

- New stochastic hashed-grid algorithm
  - Only keep a single photon stochastically
  - Data-parallel and uniform work distribution
Algorithm

Photons

Stochastic Hashed Grid

Hash entries

Hash counters

Scatter (overwrite)

Scatter (atomic inc)

Gather (mul)

Hash

Eye ray hit points
Results: Path Tracing vs PPM

- Path Tracing: 10min rendering time, full global illumination
- PPM: Same ray tracing core

Full global illumination
Rendering time: 10min
Same ray tracing core
Rasterization as spatialized compute

- Image Space Photon Mapping
- Ambient Occlusion Volumes
- Stochastic Rasterization
Image-Space Photon Mapping

Hardware-Accelerated Global Illumination by Image Space Photon Mapping, by Morgan McGuire and David Luebke

High Performance Graphics 2009
Image Space Photon Mapping

First bounce: \textit{bounce map}

Final bounce: \textit{photon volumes}
ISPM Radiance Estimate

- Traditional photon mapping: \textit{gather}
  - Per pixel
  - $k$-NN search in $k$-d tree
  - World-space (3D)

- Image space photon mapping: \textit{scatter}
  - Per photon
  - Hardware rasterization using photon volumes
  - Image space (2D)
Invoke an illumination contribution on all pixels for which a photon might be a valid estimate of incident radiance. Not virtual point lights (a.k.a. instant radiosity) not 2D splatting.
Ambient Occlusion Volumes

Ambient occlusion volumes by Morgan McGuire, High Performance Graphics 2010

Ambient Occlusion in Theory

• Occlusion of incoming ambient light

Light from these directions reaches the surface

Light from these directions does not reach the surface
What It Looks Like
Ray Casts for Ambient Occlusion

- Cast a number of rays from the point to be shaded
- Determine occlusion distance, apply falloff, sum
Occlusion of a Triangle

- Combine four precomputed bitwise occlusion masks
Rasterize Triangle Regions of Influence
Bounding the Region of Influence

Multiple choices for bounding volume

• Large triangles: construct hexagonal prism
• Small triangles: construct hemispherical billboard
Ambient Occlusion Volumes

Visualization of a similar analytic approach (McGuire, HPG2010)
Stochastic Rasterization

Real-Time Stochastic Rasterization on Conventional GPU Architectures, by McGuire, Enderton, Shirley, Luebke
High Performance Graphics 2010
Stochastic Rasterization

- Rasterize bound of time-continuous triangle
- Intersect in fragment shader
Summary

• Three interesting computational graphics patterns
  1. Render $\rightarrow$ analyze $\rightarrow$ render
  2. Render directly to alternate data structure
  3. Rasterize to invoke spatially localized computation

• A frequent theme: *deferred shading*
Many more examples!

From *Metro 2033*, © THQ and 4A Games

Real-Time Lens Blur Effects and Focus Control, Lee et al., SIGGRAPH 2010
Acknowledgements

Many thanks for use of their slides & images:

Samuli Laine  Tero Karras  Morgan McGuire
Eric Enderton  Justin Hensley  Jason Yang
Mike Houston  Aaron Lefohn  Andrew Lauritzen
Sungkil Lee  Toshiya Hachisuka  Metro 2033