NVIDIA Case Studies:
OptiX & Image Space Photon Mapping

David Luebke
NVIDIA Research
How Far Beyond?

The continuum “Beyond Programmable Shading”

“Just” programmable shading: DX, OGL
How Far Beyond?

The continuum “Beyond Programmable Shading”

“Just” programmable shading: DX, OGL

“Pure” compute-based graphics: CUDA, OptiX
SIGGRAPH 2008 Demo

SIGGRAPH 2009 Product: OptiX
The OptiX Engine

A General Purpose Ray Tracing API
- Rendering, baking, collision detection, A.I. queries, etc.
- Modern shader-centric, stateless and bindless design
- Is not a renderer but can implement many types of renderers

Highly Programmable
- Shading with arbitrary ray payloads
- Ray generation/framebuffer operations (cameras, data unpacking, etc.)
- Programmable intersection (triangles, NURBS, implicits...)

Easy to Program
- Write single ray code (no exposed ray packets)
- No need to rewrite shaders to target different hardware
Programmable Operations

- Hit program
- Miss program
- Intersection program
- Ray generation program
# Programmable Operations

<table>
<thead>
<tr>
<th>Rasterization</th>
<th>Ray Tracing</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Fragment</td>
<td>• Closest Hit</td>
</tr>
<tr>
<td>• Vertex</td>
<td>• Any Hit</td>
</tr>
<tr>
<td>• Geometry</td>
<td>• Intersection</td>
</tr>
<tr>
<td></td>
<td>• Selector</td>
</tr>
<tr>
<td></td>
<td>• Ray Generation</td>
</tr>
<tr>
<td></td>
<td>• Miss</td>
</tr>
<tr>
<td></td>
<td>• Exception</td>
</tr>
</tbody>
</table>
Shading in OptiX

Interconnection of shaders defines the outcome
- Whitted ray tracing, cook, path tracing, photon mapping
- Or collision detection, sound propagation, ...

Shading “language” is based on C/C++ for CUDA
- No new language to learn
- Powerful language features available immediately
  - Pointers
  - Templates
  - Overloading
  - Default arguments
  - Classes (no virtual functions)

Adds a powerful object model designed for ray tracing

Caveat: still need to use it responsibly to get perf
How Far Beyond?

The continuum “Beyond Programmable Shading”

Interesting middle ground!

“Just” programmable shading: DX, OGL

“Pure” compute-based graphics: CUDA, OptiX
How Far Beyond?

Image Space Photon Mapping

“Just” programmable shading: DX, OGL

“Pure” compute-based graphics: CUDA, OptiX
Hardware-Accelerated Global Illumination by Image Space Photon Mapping

Morgan McGuire
Williams College

David Luebke
NVIDIA Corporation
Goal: Dynamic Global Illumination
Multi-Bounce Video
Photon Mapping Time (seconds)
e.g., [Jensen 01, Purcell et al. 03, Zhou et al. 08]

Image Space Photon Mapping (milliseconds)

GPU Bounce Map  CPU Trace  GPU Photon Volumes

Data Transfer
Hardware accelerated global illumination by image space photon mapping

System

CPU Trace

1st Bounce: Bounce Map

Last Bounce: Photon Volumes

Direct + Shadows

Last Bounce:

Photon Volumes

Estimate subsampled indirect radiance via Photon Volumes

Upsample Indirect

Composite and Tonemap

8 cores in our implementation
Bounce Map

Position

Normal

Material Parameters (BSDF)

Outgoing Power

Outgoing Direction

Refractive Index,
A Priori Differential Probability
(Trace Rays)
(Direct Illumination)
Radiance Estimate

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

- **Image-space photon mapping:** *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) or 2D splatting.

Photon Volumes
Hardware-accelerated global illumination by image space photon mapping
Hardware-accelerated global illumination by image space photon mapping
Results
Hardware accelerated global illumination by image space photon mapping
Hardware-accelerated global illumination by image space photon mapping
Hardware accelerated global illumination by image space photon mapping:

Direct + Indirect
Hardware-accelerated global illumination by image space photon mapping

Direct + Indirect
Performance @ 1920 x 1080, 30 lights

- No PLS: 2 fps
- Conservative PLS: 15 fps, 23 fps
- Exact PLS: 21 fps, 31 fps
- 5 fps

Hardware - accelerated global illumination by image space photon mapping
Subsampling

- Joint-bilateral upsampling on normal and depth
- Subsample *incident* radiance, *not* pixel color (no diffuse surface assumption)
Diffuse Interreflection Video
Game Demo: Ironworks

Hardware-Accelerated Global Illumination by Image Space Photon Mapping
Morgan McGuire  David Luebke
Williams College  NVIDIA Corporation
Caustics Video
ISPM Properties

• **General** physically-based method
  • Caustics, color bleeding, contact shadows, etc.

• **As fast as** single-phenomenon GPU algorithms
  • (but lower quality for *that* phenomenon)
  • 1M poly scenes 1920x1080 @ 30 fps

• **Simple** implementation
  • ~600 statements in GLSL and C++
Theme

*Exploit the strengths of graphics pipeline, without being fettered to it*

Rasterizer, ROP, Z Cull, Z compression:

exploit single center of projection

ROP, texture:

massive blending capability ("hidden" FLOPS)

*NVIDIA GPUs: CUDA + Acceleration Engines like OptiX*
Thank You!

Source code:

ISPM: http://graphics.cs.williams.edu
G3D: http://g3d-cpp.sf.net

- Onos and NS Theme from *Natural Selection 2* by Unknown Worlds Entertainment
- Nexus 6 from *Tremulous*
- Bunny from The Stanford 3D Scanning Repository
- Ironworks from *Quake Live* by id Software
- Sponza Atrium by Marko Dabrovic

Special thanks to Qi Mo, Evan Hart, Kefei Lei, Daniel Fast, Eric Enderton, and anonymous HPG09 reviewers
Backup Slides
(from HPG presentation)
Hardware accelerated global illumination by image space photon mapping

Problems

Heuristic Solutions

Jensen

ISPM

Sources of error

Surface point to be shaded

Gather kernel extent

Photons and scatter kernels:

- Green: Good estimator
- Red: Poor estimator
Hardware-accelerated global illumination by image space photon mapping.
Hardware accelerated global illumination by image space photon mapping
Radiance Estimate

\[ L_o(\vec{s}, \vec{\omega}_o) = \int f(\vec{x}, \vec{\omega}_i, \vec{\omega}_o) \ast L_i(\vec{s}, \vec{\omega}_i) \ast \max(0, \vec{\omega}_i \cdot \vec{n}) d\vec{\omega}_i \]

\[ \Delta L_o(\vec{s}, \vec{\omega}_o) = f(\vec{s}, \vec{\omega}_i, \vec{\omega}_o) \ast \Phi_i \ast \max(0, \vec{\omega}_i \cdot \vec{n}) \ast \kappa(\vec{x} - \vec{s}, \vec{n}_p) \]

\[ \kappa(\vec{x} - \vec{s}, \vec{n}_p) = \text{texture1D(gaussian, } t) \]

\[ t = \frac{|\vec{x} - \vec{s}|}{r_{xy}} \left(1 - \left| \frac{\vec{x} - \vec{s}}{|\vec{x} - \vec{s}|} \cdot \vec{n}_p \right| \frac{r_{xy} - r_z}{r_z} \right) \]
Assumptions

- Point light
- Pinhole camera

Limitations

- Clipping at near plane – annoying, but ignorable/avoidable
- 4x more expensive than direct illumination (heavy fill consumption)
- Consistent, but biased (like photon mapping)
Future Work

• Tracing on GPU
• Fill rate
• Shaping photon volume
• Game-specific optimizations and analysis
  • The Cornell box is harder to render than Natural Selection 2!
# Performance Details

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Ironworks (Car)</td>
<td>1</td>
<td>161,612</td>
<td>141,402</td>
<td>20,120</td>
<td>20</td>
<td>2</td>
<td>8,448</td>
<td>1</td>
<td>1520x1080</td>
<td>4x4</td>
<td>7,162</td>
<td>3,450</td>
<td>14.2</td>
<td>0.1</td>
<td>6.2</td>
<td>17.2</td>
<td>8.1</td>
<td>121.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>161,612</td>
<td>141,402</td>
<td>20,120</td>
<td>20</td>
<td>2</td>
<td>8,448</td>
<td>3</td>
<td>1520x1080</td>
<td>4x4</td>
<td>8,846</td>
<td>5,134</td>
<td>14.2</td>
<td>0.1</td>
<td>6.3</td>
<td>24.8</td>
<td>10.2</td>
<td>121.8</td>
</tr>
<tr>
<td>Ironworks (Water)</td>
<td>8</td>
<td>121,962</td>
<td>66,734</td>
<td>55,228</td>
<td>1</td>
<td>1</td>
<td>26,896</td>
<td>1</td>
<td>1520x1080</td>
<td>4x4</td>
<td>37,806</td>
<td>18,852</td>
<td>8.9</td>
<td>0.1</td>
<td>13.5</td>
<td>49</td>
<td>8.2</td>
<td>159.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>121,962</td>
<td>66,734</td>
<td>55,228</td>
<td>1</td>
<td>1</td>
<td>40,000</td>
<td>1</td>
<td>1520x1080</td>
<td>4x4</td>
<td>56,689</td>
<td>28,273</td>
<td>9</td>
<td>0.1</td>
<td>19.7</td>
<td>72.7</td>
<td>16.9</td>
<td>159.9</td>
</tr>
<tr>
<td>NS2</td>
<td>13</td>
<td>73,831</td>
<td>58,319</td>
<td>14,762</td>
<td>1</td>
<td>4</td>
<td>20,356</td>
<td>3</td>
<td>1520x1080</td>
<td>6x6</td>
<td>12,228</td>
<td>7,143</td>
<td>12.5</td>
<td>0.1</td>
<td>8</td>
<td>21.6</td>
<td>9.6</td>
<td>139.9</td>
</tr>
<tr>
<td>(Cargo Bay PVS)</td>
<td></td>
<td>73,831</td>
<td>58,319</td>
<td>14,762</td>
<td>1</td>
<td>4</td>
<td>20,356</td>
<td>1</td>
<td>1520x1080</td>
<td>6x6</td>
<td>10,004</td>
<td>4,919</td>
<td>12.6</td>
<td>0.1</td>
<td>7.6</td>
<td>16.1</td>
<td>8.3</td>
<td>139.9</td>
</tr>
<tr>
<td>(Circular PVS)</td>
<td></td>
<td>404,773</td>
<td>255,153</td>
<td>147,620</td>
<td>10</td>
<td>4</td>
<td>20,356</td>
<td>3</td>
<td>1520x1080</td>
<td>6x6</td>
<td>12,578</td>
<td>7,489</td>
<td>17.2</td>
<td>0.1</td>
<td>8.3</td>
<td>29.9</td>
<td>21.0</td>
<td>95.7</td>
</tr>
<tr>
<td>(Full Map)</td>
<td>9</td>
<td>1,190,405</td>
<td>880,403</td>
<td>310,000</td>
<td>21</td>
<td>30</td>
<td>95,948</td>
<td>1</td>
<td>1520x1080</td>
<td>6x6</td>
<td>97,982</td>
<td>61,000</td>
<td>150.2</td>
<td>0.7</td>
<td>67</td>
<td>284.4</td>
<td>14.5</td>
<td>6.5</td>
</tr>
<tr>
<td>Glass Sphere</td>
<td>11</td>
<td>8,034</td>
<td>4,022</td>
<td>4,012</td>
<td>2</td>
<td>5</td>
<td>26,640</td>
<td>2</td>
<td>1,520x1080</td>
<td>6x6</td>
<td>19,343</td>
<td>10,941</td>
<td>22</td>
<td>0.6</td>
<td>9.8</td>
<td>8.9</td>
<td>3.8</td>
<td>199.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8,034</td>
<td>4,022</td>
<td>4,012</td>
<td>2</td>
<td>5</td>
<td>137,780</td>
<td>3</td>
<td>512x512</td>
<td>3x3</td>
<td>108,569</td>
<td>66,447</td>
<td>2.5</td>
<td>0.8</td>
<td>36.1</td>
<td>31.0</td>
<td>21.4</td>
<td>199.0</td>
</tr>
<tr>
<td>Objects in Box</td>
<td>6</td>
<td>15,642</td>
<td>9,826</td>
<td>9,816</td>
<td>3</td>
<td>2</td>
<td>27,620</td>
<td>3</td>
<td>1520x1080</td>
<td>8x8</td>
<td>12,985</td>
<td>7,319</td>
<td>9.0</td>
<td>0.3</td>
<td>7.6</td>
<td>4.0</td>
<td>11.3</td>
<td>163.0</td>
</tr>
<tr>
<td>Sponsor Atrium</td>
<td>11</td>
<td>67,414</td>
<td>66,934</td>
<td>480</td>
<td>1</td>
<td>1</td>
<td>13,636</td>
<td>1</td>
<td>1520x1080</td>
<td>4x4</td>
<td>23,244</td>
<td>11,308</td>
<td>8.8</td>
<td>0.1</td>
<td>8.4</td>
<td>56.7</td>
<td>12.9</td>
<td>195.2</td>
</tr>
<tr>
<td>Glass Bunny</td>
<td>7</td>
<td>67,414</td>
<td>66,934</td>
<td>480</td>
<td>1</td>
<td>1</td>
<td>6,240</td>
<td>1</td>
<td>1520x1080</td>
<td>4x4</td>
<td>10,573</td>
<td>5,136</td>
<td>8.8</td>
<td>0.1</td>
<td>5.4</td>
<td>28.2</td>
<td>7.2</td>
<td>195.2</td>
</tr>
<tr>
<td>Caustic Ring</td>
<td>11</td>
<td>812</td>
<td>412</td>
<td>400</td>
<td>1</td>
<td>1</td>
<td>101,124</td>
<td>5</td>
<td>1520x1080</td>
<td>1x1</td>
<td>27,917</td>
<td>5,020</td>
<td>1.4</td>
<td>0.1</td>
<td>21.3</td>
<td>6.4</td>
<td>3.4</td>
<td>200</td>
</tr>
<tr>
<td></td>
<td>812</td>
<td>412</td>
<td>400</td>
<td>1</td>
<td>1</td>
<td>324,500</td>
<td>5</td>
<td>1520x1080</td>
<td>1x1</td>
<td>85,919</td>
<td>15,630</td>
<td>1.4</td>
<td>0.3</td>
<td>65.5</td>
<td>20.1</td>
<td>9.5</td>
<td>200</td>
<td>10.9</td>
</tr>
<tr>
<td></td>
<td>812</td>
<td>412</td>
<td>400</td>
<td>1</td>
<td>1</td>
<td>101,124</td>
<td>1</td>
<td>1520x1080</td>
<td>1x1</td>
<td>27917</td>
<td>5020</td>
<td>7.6</td>
<td>0.1</td>
<td>22.1</td>
<td>14.3</td>
<td>197.5</td>
<td>24.3</td>
<td></td>
</tr>
</tbody>
</table>

**Key:**
- **Bold:** Corresponds to a figure
- **Highlight:** Changed from first row
- **Highlight:** Bottleneck

Hardware-accelerated global illumination by image space photon mapping
Trace-bound

139k polys
36k photons
16 fps

Trace: 25 ms
Fill-bound

176k polys
8 objects
11 lights
100k photons

156 Mpix fill—80x overdraw!
10 fps