Screen-Space Normal Distribution Function Caching for Consistent Multi-Resolution Rendering of Large Particle Data ... A CPU Prospect?

Mohamed Ibrahim¹, Patrick Wickenhäuser², Peter Rautek¹, Guido Reina², Markus Hadwiger¹

¹King Abdullah University of Science and Technology (KAUST),
²University of Stuttgart
Expanding Fluid Layer (30 million particles)
Expanding Fluid Layer (30 million particles)

Flat Surfaces

Gas distribution

Droplets
Standard Raycasting
Screen-Space Normal Distribution Functions (S-NDFs)
Screen-Space Normal Distribution Functions (S-NDFs)
Screen-Space Normal Distribution Functions (S-NDFs)

\[ P(\omega_m) = \frac{A'_{\perp}(\omega_m)}{\Omega'} \]
Screen-Space Normal Distribution Functions (S-NDFs)
Screen-Space Normal Distribution Functions (S-NDFs)
Screen-Space Normal Distribution Functions (S-NDFs)
Scale Consistency

Lod 0

Raycasting (1 ray)

Raycasting ($4^{lod}$ rays)

Our Approach
Linearly Filtering S-NDFs

\[ \frac{1}{4} \sum \begin{bmatrix} \text{image} \end{bmatrix} = \begin{bmatrix} \text{image} \end{bmatrix} \]
Caching S-NDFs

S-MIP

Query

Map/Allocate

Send

visible region

cache
Supported Operations

S-NDFs

+ S-MIP

⇒

Relighting

Navigational
### Performance Measures (GPU)

<table>
<thead>
<tr>
<th>data set</th>
<th>particle count</th>
<th>re-light [ms]</th>
<th>ray casting + S-MIP</th>
<th>S-NDF + S-MIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Laser ablation crown</td>
<td>518,000</td>
<td>1129.74</td>
<td>26.53</td>
<td></td>
</tr>
<tr>
<td>Copper/silver mixture</td>
<td>14,500,00</td>
<td>1309.35</td>
<td>19.37</td>
<td></td>
</tr>
<tr>
<td>Expanding fluid layer</td>
<td>30,000,000</td>
<td>1523.61</td>
<td>22.19</td>
<td></td>
</tr>
<tr>
<td>Laser ablation</td>
<td>48,000,000</td>
<td>1253.84</td>
<td>22.17</td>
<td></td>
</tr>
<tr>
<td>Instanced laser (5×)</td>
<td>5× 12,500,000</td>
<td>1524.25</td>
<td>18.95</td>
<td></td>
</tr>
<tr>
<td>Large laser ablation</td>
<td>199,940.704</td>
<td>2203.66</td>
<td>16.87</td>
<td></td>
</tr>
</tbody>
</table>
Implementation Details

GPU ⇒ dataset ⇒ sampling ⇒ S-NDF accumulation ⇒ visible region ⇒ Hybrid Implementation?
OSPRay

- Rendering API
- Platform Independent
- Visualization-oriented
- CPU implementation

OSPRay — A CPU Ray Tracing Framework for Scientific Visualization.
Wald, I et al., IEEE Scientific Visualization 2016
OSPRay ... Performance

Diffuse Path Tracing Performance of Embree vs. NVIDIA OptiX Prime.

Wald, I. et al., ACM SIGGRAPH 2014
OSPRay ... Integration

RTCBoundsFunc  RTCIntersectFunc  RTCOccludedFunc
OSPRay + S-NDFs + S-MIP?

gpu
⇒
sampling
⇒
S-NDF accumulation
⇒
visible region
⇒
single pass!

Dataset
OSPRay + S-NDFs + S-MIP?

- Large data
- Large cache
- Finer S-NDFS
- 4D S-NDFS
- Fast traversal
- Multiple samples
- Guided sampling

Tools:
- Paraview
- MegaMol
- VMD
- Visit

---
Thank you

http://www.vccvisualization.org
Application Scenarios
Implementation ... Device Abstraction

• Ideal implementation differs according to target:
  • PCI-card based on Xeon Phi coprocessor
  • MPI-parallelism
• OSPRay supports device abstraction
  • API calls are seen as a stream of commands.
    • Each, internally, routed to one of multiple possible backend devices.
Implementation ... Device Abstraction

• Local Device
  • Executes all rendering right in the same process as a the visualization application.

• COI Device
  • Offload to first-generation Xeon Phi Knights Corner coprocessor cards.

• MPIDevice
  • Support to MPI-parallel rendering

• All devices build on top of a shared rendering infrastructure that implements the actors.
Software Infrastructure

• Build on top of
  • Intel SPMD Program Compiler (ISPC)
  • Embree ray tracing kernels

• Embree
  • Building and traversing acceleration structures.
  • Automatically selects acceleration structure and traversal kernels best suited for a given CPU.
Software Infrastructure

• ISPC
  • All throughput sensitive operations that require vectorization (rendering, shading, primitive intersection)
  • Allows transparent targeting of multiple vector instruction set architectures (ISA).
    • Intel SSE4, AVX, AVX2, etc...

<table>
<thead>
<tr>
<th>OSPRay API (ospray.h)</th>
<th>Local Device</th>
<th>COI Device</th>
<th>MPI Device</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>COI</td>
<td></td>
<td>MPI</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OSPRay Core (shared) (Geoms, Volumes, Renderers, …)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C++</td>
</tr>
</tbody>
</table>

| CPU ISAs (Xeon/Xeon Phi) |