Real-Time Appearance Preserving Out-of-Core Rendering with Shadows

Michael Guthe, Pavel Borodin, Ákos Balázs and Reinhard Klein

University of Bonn
Computer Graphics Group
Out-of-Core HLOD Rendering

+ **Adjustable Balance: CPU ⇐⇒ GPU**

+ **Guaranteed screen space error**
  - No popping artifacts along silhouettes

G. Varadhan, D. Manocha, *Out-of-Core Rendering of Massive Geometric Environments*, 2002

University of Bonn • Computer Graphics Group
M. Guthe, P. Borodin, Á. Balázs and R. Klein

EGSR 2004
Out-of-Core HLOD Rendering

– **Not appearance preserving**
  - Blurry, small features are missing
  - Severe popping artifacts in shading

Motivation

- High quality rendering of gigabyte sized models (with realistic shadows)
General Pipeline

1. Segmentation
2. Simplification
3. HLOD Selection & Culling
4. Rendering

Preprocessing

Runtime
Normal Map Textures

- Parameterization of whole model not possible
  - Parameterization for each cell
  - Popping artifacts between LODs

Appearance Preserving Simplification

- **Exact view parameters required**
  ⇒ Not directly applicable to HLOD rendering

Appearance Preserving Simplification

- Linear normal interpolation assumed (e.g. Phong shading, Environment Maps)
  - Error measure: distance to next point with same (interpolated) normal $\vec{n}_{\text{int}}$
Appearance Preserving Simplification

- Error as orthogonal combinatination:

\[ \varepsilon^2 = \varepsilon_{\text{geo}}^2 + \varepsilon_{\text{norm}}^2 \]

⇒ Modified Hausdorff error algorithm
Appearance Preserving Simplification

**Approximation with maximum curvature**

\[ \mathcal{E}_{\text{norm}} \approx \frac{\arccos(\vec{n} \cdot \vec{n}_{\text{int}})}{\kappa_1} \]

\[ \kappa_1 \approx \max \left( \frac{\arccos(\vec{n}_1 \cdot \vec{n}_2)}{\|P_1 - P_2\|}, \frac{\arccos(\vec{n}_1 \cdot \vec{n}_3)}{\|P_1 - P_3\|}, \frac{\arccos(\vec{n}_2 \cdot \vec{n}_3)}{\|P_2 - P_3\|} \right) \]

- For small angles: \[ \arccos(\vec{n}_a \cdot \vec{n}_b) \approx \|\vec{n}_a - \vec{n}_b\| \]
Appearance Preserving Simplification

- Antialiasing before simplification
  - Smooth normal when all adjacent edges smaller than $\varepsilon$
  - More efficient simplification
Hybrid Point Polygon Simplification

- Transition between triangles and points in one direction only
  ⇒ Efficiency of simplification depends on manually selected transition point

Point Generation

When to replaced triangles by points?

- Transition from points to triangles is complicated
  - Point generation after hierarchical simplification
- Points renders twice as fast as point sized triangles
  - At most 3 additional points per triangle
Compression

- Loading can become bottleneck

- Compression algorithms with high decompression speed required
  - Compression of triangles with Cut-Border machine
  - Compression of points with cloud compression
    (M. Botsch et al., *Efficient High Quality Rendering of Point Sampled Geometry*, 2002)
General Pipeline

1. Segmentation
2. Simplification
3. HLOD Selection & Culling
4. Rendering
Hard Shadows

  - Needs additional CPU for shadow generation
Shadow Algorithms

Soft Shadows

  - **Realistic shadows**
    - Realtime only for simple scenes

  - **Realtime for moderately complex scenes**
    - Plausible, but not realistic Shadows
Shadow Algorithms

Shadow caster LOD selection required

picture from: G. Varadhan, D. Manocha, Out-of-Core Rendering of Massive Geometric Environments, 2002
General Pipeline

1. Segmentation
2. Simplification
3. HLOD Selection & Culling
4. Rendering

Preprocessing

Runtime
Shadow Generation

Point light sources

\[ \varepsilon_h = \frac{\varepsilon_r d_l}{d_l + d_p} \]
Shadow Generation

Area light sources

Intensity change of $\gamma_i$ leads to:

$$\varepsilon_i = \frac{\gamma_i s_l d_p}{d_l + d_p}$$

$$\varepsilon_c = \varepsilon_h + \varepsilon_i$$

$$\varepsilon_p = \gamma_l s_p$$
Shadow Generation

- Only geometric approximation
- $\epsilon_r$ has to be known
  - Select receiver LODs in first pass
  - Select caster LODs with view aligned BB of each selected receiver hierachy level
    - For visible cells $\epsilon_c \leq \epsilon_r$
    - No self shadowing artifacts

picture from: G. Varadhan, D. Manocha, Out-of-Core Rendering of Massive Geometric Environments, 2002
Shadow Caster Prefetching

- Prefetching of shadow casters required
- Viewer movement similar to light source movement
  - Standard techniques
## Results

### Models and their HLOD size on disk

<table>
<thead>
<tr>
<th>model</th>
<th>#triangles</th>
<th>app. pres.</th>
<th>geometric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dragon</td>
<td>871,414</td>
<td>4.9 MB</td>
<td>2.4 MB</td>
</tr>
<tr>
<td>Happy Buddha</td>
<td>1,087,716</td>
<td>7.5 MB</td>
<td>3.5 MB</td>
</tr>
<tr>
<td>David 2mm</td>
<td>8,254,150</td>
<td>35.5 MB</td>
<td>20.2 MB</td>
</tr>
<tr>
<td>Lucy</td>
<td>28,055,742</td>
<td>148.4 MB</td>
<td>66.7 MB</td>
</tr>
<tr>
<td>David 1mm</td>
<td>56,230,343</td>
<td>321.3 MB</td>
<td>138.6 MB</td>
</tr>
<tr>
<td>St. Matthew</td>
<td>372,422,615</td>
<td>1566.2 MB</td>
<td>649.6 MB</td>
</tr>
</tbody>
</table>
## Results

### Average frame rates of different models

<table>
<thead>
<tr>
<th>model</th>
<th>app. pres.</th>
<th>geometric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dragon</td>
<td>99 fps</td>
<td>122 fps</td>
</tr>
<tr>
<td>Happy Buddha</td>
<td>81 fps</td>
<td>117 fps</td>
</tr>
<tr>
<td>David 2mm</td>
<td>64 fps</td>
<td>155 fps</td>
</tr>
<tr>
<td>Lucy</td>
<td>55 fps</td>
<td>110 fps</td>
</tr>
<tr>
<td>David 1mm</td>
<td>57 fps</td>
<td>114 fps</td>
</tr>
<tr>
<td>St. Matthew</td>
<td>53 fps</td>
<td>93 fps</td>
</tr>
</tbody>
</table>

Athlon 2800+ with Radeon 9800XT
Results

Comparison of image quality

geometric  geom. with $\frac{1}{4}$ error  app. preserving
(same number of triangles)
Comparison of image quality

geometric

geom. with $\frac{1}{4}$ error

app. preserving

(same number of triangles)
## Results

### Average frame rates with shadow

<table>
<thead>
<tr>
<th>model</th>
<th>no shadow</th>
<th>point light</th>
<th>area light</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dragon</td>
<td>99 fps</td>
<td>75 fps</td>
<td>20 fps</td>
</tr>
<tr>
<td>Happy Buddha</td>
<td>81 fps</td>
<td>70 fps</td>
<td>20 fps</td>
</tr>
<tr>
<td>David 2mm</td>
<td>64 fps</td>
<td>55 fps</td>
<td>21 fps</td>
</tr>
<tr>
<td>Lucy</td>
<td>55 fps</td>
<td>40 fps</td>
<td>15 fps</td>
</tr>
<tr>
<td>David 1mm</td>
<td>57 fps</td>
<td>45 fps</td>
<td>17 fps</td>
</tr>
<tr>
<td>St. Matthew</td>
<td>53 fps</td>
<td>37 fps</td>
<td>13 fps</td>
</tr>
</tbody>
</table>

Athlon 2800+ with Radeon 9800XT
Results

Screen shots from camera path

with hard shadow                   with soft shadow
Demo
Acknowledgements

- The models used in this talk were provided by the Digital Michelangelo Project.
- This work was partially funded by the BMBF under the project OpenSG PLUS and by the European Union under the project of RealReflect.