Outline

- Point Based Rendering
  - Surface splatting
- This paper’s unique contributions
  - Multi-Pass deferred shading
  - Elliptical Weighted Average approximation
Point Based Rendering

- Laser scanning range data, huge
- Mesh: need connectivity, triangulation
- PBR: Rendering directly with point set

Courtesy of Stanford “The Digital Michelangelo Project”
Point-based surface representation

No connectivity

Surface Splatting
Volume and Surface
PBR - Splat

- Volume splatting to Surface splatting
- Interactive Application, trade-offs
  - High performance
  - High visual quality
<table>
<thead>
<tr>
<th>Method</th>
<th>Persp. Correct</th>
<th>Phong Shading</th>
<th>Anti-Aliasing</th>
<th>Splats/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>EWA Splatting</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>1M</td>
</tr>
<tr>
<td>NV30 Splatting PG '03</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>27M</td>
</tr>
<tr>
<td>Persp. Accurate GI '04</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>5M</td>
</tr>
<tr>
<td>Phong Splatting PBG '04</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>6M</td>
</tr>
<tr>
<td>NV40 Splatting PBG '05</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Previous Work

- Well designed data structure: QSplat
- Pre-processing time: 1 hour/ 10M points
- Represent each splat by an alpha-textured quad or triangle
  - Expensive, 3~4 times in size
- Representing each splat by one OpenGL vertex + pixel shader for rasterization
  - Correct depth values for correcting blending result: depth map
Procedures

• Splat rasterization
• Hardware deferred shading, using MRT
  • Visibility pass - depth map
  • Attribute pass - color value & normal vector
• Shading pass - final color with MRTs
**Splat Rasterization** *(BSK04)*

Figure 2: *Defining a texture function on the surface of a point-based object.*
Figure 4: Rendering by surface splatting, resampling kernels are accumulated in screen space.
Splat Rasterization

- A OpenGL vertex $c \rightarrow d^2d$ image space square (vert.s)
- Pixel shader test pixel(x,y) in $d^2d$ in or out of projected elliptical splat contour, discard outsiders
- Local raycast to get point on splat surface (with local parameter $(u, v)$), test against
  $$u^2 + v^2 = (u_j^T(q - c_j))^2 + (v_j^T(q - c_j))^2 \leq 1$$
- Calculate and accumulate weighting factor using
  $$w(x, y) = h \left( \sqrt{u^2 + v^2} \right)$$
- Pixel depth, normal, color determined in each pass
HW-Accelerated Deferred Shading

Visibility Splatting

Avoid holes

Generate Normal map

Normalization Shading
Normals

- Interpolate normal vectors
- No connectivity like for meshes
- Assign linear normal field
- Static geometries
- Splat wrt. normals, generate normal map in multi-pass rendering
Normalization Shading

Final normalization step to fix saturated lighting
EWA approximation on screen space

- A heuristic approximating EWA screen-space filter
- Generate enough fragments for anti-aliasing purpose, done in vertex shader
- Limit projected splat to be at least \((2\rho)(2\rho)\) pixels
- Combine minimum, done in fragment shader
- \#frag/#pixel increases to about 30, need DS
Approximate EWA Filtering

- **Reconstruction filter radius**
  \[ r_{3D} := \sqrt{u^2 + v^2} \]

- **Screen-space filter radius**
  \[ r_{2D} := \frac{d(x, y)}{\sigma} \]

- **Combined filter**
  \[ r := \min\{r_{3D}, r_{2D}\} \]
  \[ w := \text{Gauss}(r) \]
  \[ r \leq 1 \]
  \[ r_{2D} \leq 1 \]
  \[ r_{3D} \leq 1 \]
No filtering
Object space
Object and screen space
Some Results
Phong shading
Figure 5: From left to right: The Phong-shaded octopus model and NPR-shaded renderings of the dinosaur model, the Igea artifact, and the massive Lucy dataset. All models are rendered with shadow mapping enabled and hence require one additional visibility rendering pass for the shadow map generation.
Speed

<table>
<thead>
<tr>
<th>Model</th>
<th>#splats</th>
<th>Overdraw</th>
<th>Phong-SM</th>
<th>NPR-SM</th>
<th>Phong+SM</th>
<th>NPR+SM</th>
<th>[BK03]</th>
<th>[BSK04]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Balljoint</td>
<td>137k</td>
<td>5.9 / 7.2</td>
<td>20.1</td>
<td>15.5</td>
<td>13.3</td>
<td>11.9</td>
<td>24.8</td>
<td>4.5</td>
</tr>
<tr>
<td>Max</td>
<td>655k</td>
<td>7.9 / 15.4</td>
<td>22.0</td>
<td>20.4</td>
<td>14.9</td>
<td>14.5</td>
<td>27.0</td>
<td>5.9</td>
</tr>
<tr>
<td>David Head</td>
<td>1.1M</td>
<td>6.4 / 14.4</td>
<td>23.9</td>
<td>22.6</td>
<td>16.3</td>
<td>15.8</td>
<td>27.1</td>
<td>5.6</td>
</tr>
<tr>
<td>David Head</td>
<td>4.0M</td>
<td>6.9 / 37.7</td>
<td>26.0</td>
<td>25.3</td>
<td>17.5</td>
<td>17.2</td>
<td>31.3</td>
<td>5.5</td>
</tr>
<tr>
<td>David</td>
<td>8.3M</td>
<td>7.0 / 202</td>
<td>22.6</td>
<td>22.0</td>
<td>16.2</td>
<td>15.5</td>
<td>19.6</td>
<td>4.6</td>
</tr>
<tr>
<td>Lucy</td>
<td>14M</td>
<td>19.3 / 242</td>
<td>22.6</td>
<td>22.1</td>
<td>15.9</td>
<td>15.2</td>
<td>20.2</td>
<td>5.0</td>
</tr>
</tbody>
</table>

Table 1: This table shows the performance of our rendering approach in million splats per second for a 512 × 512 window using a GeForce 6800 Ultra GPU. We give timings for several different shaders (Phong shading, NPR shading, with and without shadow mapping) and compare to the fast but low-quality splatting of [BK03], and the high-quality but expensive Phong splatting [BSK04]. Due to per-pixel Phong shading and anti-aliasing, the quality of our method is superior even to [BSK04], while the rendering performance is still comparable to [BK03]. The third column shows the average number of fragments contributing to resulting image pixels, without and with our anti-aliasing technique. Since the latter generates significantly more fragments for complex models, the acceleration provided by the deferred shading approach is even more important.
# Quality

<table>
<thead>
<tr>
<th></th>
<th>Persp. Correct</th>
<th>Phong Shading</th>
<th>Anti-Aliasing</th>
<th>Splats/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>EWA Splatting</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>1M</td>
</tr>
<tr>
<td>NV30 Splatting PG ‘03</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>27M</td>
</tr>
<tr>
<td>Persp. Accurate GI ‘04</td>
<td>(✓)</td>
<td>X</td>
<td>✓</td>
<td>5M</td>
</tr>
<tr>
<td>Phong Splatting PBG ‘04</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>6M</td>
</tr>
<tr>
<td>NV40 Splatting PBG ‘05</td>
<td>✓</td>
<td>✓</td>
<td>(✓)</td>
<td>23M</td>
</tr>
</tbody>
</table>
Drawbacks

• Flexibility of z-buffering
• Need expensive two render passes for visibility splatting and attribute blending
Conclusion

• New GPU features
• MRT - Multiple Render Target
• High precision floating point pipe
  • Arithmetic, buffers, textures, blending
• High performance and high quality splatting
  • Deferred shading
• EWA approximation
Questions?
References

• http://people.csail.mit.edu/matthias/
• http://graphics.cs.cmu.edu/projects/adpewa/
• http://www-i8.informatik.rwth-aachen.de/publications/publications.html