High Quality Surface Splatting on Today’s GPUs

Mario Botsch, Alexander Hornung, Matthias Zwicker, Leif Kobbelt
Introduction

- Point-based representations suitable for massive data sets
- Rendering is important for interactive applications
  - High performance
  - High visual quality
Current PBR methods

- High quality
  - Projectively correct rasterization
  - Phong shading
  - EWA anti-aliasing
- ... or high performance
Current PBR methods

- Trade-off performance against quality
- Limited by available GPU features
- Exploit latest GPUs’ features to get both
Overview

- Related Work
- Deferred Shading
- EWA Filter Approximation
- Results
Perspectively Correct Rendering
Per-Pixel Phong Shading

Flat Shading

Gouraud Shading

Phong Shading
No filtering

EWA filtering

Anti-Aliasing

Section 3.17, we introduced the problem of aliasing and discussed some basic
principles for generating antialiased 2D primitives. Here we examine aliasing in more
detail so that we can understand when and why it occurs, laying the groundwork for
applying anti-aliasing into the visible-surface and shading algorithms covered in the
subsequent chapters. Additional material may be found in [CROW77b; CROW81]; an
explicit set of examples is included in [BLIN89a; BLIN89b].
<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>✗</td>
<td>✓</td>
<td>✓</td>
<td>1M</td>
</tr>
<tr>
<td>NV30 Splatting PG ‘03</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>27M</td>
</tr>
<tr>
<td>Persp. Accurate GI ‘04</td>
<td>(✓)</td>
<td>✗</td>
<td>✓</td>
<td>5M</td>
</tr>
<tr>
<td>Phong Splatting PBG ‘04</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>6M</td>
</tr>
<tr>
<td>NV40 Splatting PBG ‘05</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Overview

• Related Work
• Deferred Shading
• EWA Filter Approximation
• Results
Phong Shading for PBR

- Interpolate normal vectors
  - No connectivity like for meshes

- Assign linear normal field
  - Limited to static geometries

- Splat normals into framebuffer
  - Deferred shading
3-Pass Shading

Visibility
Splatting

Shading
Blending

Normalization
3-Pass Deferred Shading

Visibility Splatting

Attribute Splatting

Normalization Shading
Deferred Shading

- Compute lighting for each *image pixel*, not for each *generated fragment*

- Splats mutually overlap
  - \#fragments \approx 7 \cdot \#pixel

- Rasterization shader is bottleneck
  - Keep it small by deferring shading

→ Performance does hardly depend on surface shader complexity
Deferred Shading

- Clear separation between rasterization and surface lighting / shading
- Simplifies shader development
  - Same shader as for meshes
Required GPU Features

• Render attributes to several buffers
  • Multiple render targets (MRT)

• Accumulate at high precision
  • Floating point arithmetic
  • Floating point buffers
  • Floating point textures
  • Floating point blending
Floating Point Precision

8 bit ubyte clamped to [0,1]

16 bit float un-clamped
Overview

- Related Work
- Deferred Shading
- EWA Filter Approximation
- Results
EWA Anti-Aliasing

Screen-space low-pass filter

Object-space reconstruction filter

In 1.7, we introduced the problem of aliasing and discussed some basic ideas for generating anti-aliased 2D primitives. Here we examine aliasing in more detail so that we can understand when and why it occurs, laying the groundwork for suppressing aliasing into the visible-surface and shading algorithms covered in the later chapters. Additional material may be found in [CROW79, CROW81]; no sample set of examples is included in [BLOM88a, BLOM89].
EWA Filtering

• Combine both filters into one
  • Choose both as Gaussians
  • Affine approximation to projection

• GPU implementation in GI ‘04
  • Complex computations
  • About 5M splats/sec

➡ Screen-space filter often omitted
No filtering

Object-Space Filter Only

ALIASING AND ANTIALIASING

Section 3.17, we introduced the problem of aliasing and discussed some basic techniques for generating anti-aliased 2D primitives. Here we examine aliasing in more detail that we can understand when and why it occurs, laying the groundwork for incorporating anti-aliasing into the visible-surface and shading algorithms covered in this chapter. Additional material may be found in [CROW77b; CROW81]; a further set of examples is included in [BLIN88a; BLIN89b].

ALIASING AND ANTIALIASING

Section 3.17, we introduced the problem of aliasing and discussed some basic techniques for generating anti-aliased 2D primitives. Here we examine aliasing in more detail that we can understand when and why it occurs, laying the groundwork for incorporating anti-aliasing into the visible-surface and shading algorithms covered in this chapter. Additional material may be found in [CROW77b; CROW81]; a further set of examples is included in [BLIN88a; BLIN89b].
EWA Filtering

- minification
- magnification
- anisotropic
  - minification-magnification

reconstruction filter
low-pass filter
resampling filter
⊗ =
Approximate EWA Filtering

- Reconstruction filter
- Low-pass filter
- Our approximation
- EWA filter
Approximate EWA Filtering

- **Reconstruction filter radius**
  \[ r_{3D} := \sqrt{u^2 + v^2} \]
  \[ r_{3D} \leq 1 \]

- **Screen-space filter radius**
  \[ r_{2D} := \frac{d(x,y)}{\sigma} \]
  \[ r_{2D} \leq 1 \]

- **Combined filter**
  \[ r := \min\{r_{3D}, r_{2D}\} \]
  \[ r \leq 1 \]
  \[ w := \text{Gauss} (r) \]
Approximate EWA Filtering

- Restrict minimum projected splat size to $2\sigma \times 2\sigma$ pixels
  - Ensure enough fragments for AA
  - Done in vertex shader

- Combine minimum of radii
  - Done in fragment shader
  - 3 additional instructions only
Approximate EWA Filtering

• Simple approximation to exact EWA
  • Efficient implementation
  • Removes (most) aliasing

• Generates much more fragments
  • \#fragments \approx 30 \cdot \#pixel
  • Deferred shading!
Overview

- Related Work
- Deferred Shading
- EWA Filter Approximation
- Results
Approximate EWA Filtering

No filtering  Object-Space  Object-Space  Screen-Space
Approximate EWA Filtering

Object-Space

Object-Space
FSAA

Object-Space
Screen-Space
Phong Shading
Timings & Comparison

- Models from 100k to 14M splats
- Different surface shaders (GeForce 6800 Ultra)

<table>
<thead>
<tr>
<th>Shader Type</th>
<th>Instructions</th>
<th>Msplats/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phong</td>
<td>8</td>
<td>22.8</td>
</tr>
<tr>
<td>NPR</td>
<td>48</td>
<td>21.3</td>
</tr>
<tr>
<td>Phong + Shadow</td>
<td>23</td>
<td>15.7</td>
</tr>
<tr>
<td>NPR + Shadow</td>
<td>61</td>
<td>15.0</td>
</tr>
<tr>
<td>PG ’03</td>
<td></td>
<td>25.0</td>
</tr>
<tr>
<td>PBG ’04</td>
<td></td>
<td>5.2</td>
</tr>
</tbody>
</table>
## Comparison

<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>✗</td>
<td>✓</td>
<td>✓</td>
<td>1M</td>
</tr>
<tr>
<td>NV30 Splatting PG ‘03</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>27M</td>
</tr>
<tr>
<td>Persp. Accurate GI ‘04</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>5M</td>
</tr>
<tr>
<td>Phong Splatting PBG ‘04</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>6M</td>
</tr>
<tr>
<td>NV40 Splatting PBG ‘05</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>23M</td>
</tr>
</tbody>
</table>
Conclusion

- Latest GPUs offer important new features
  - Multiple render targets
  - Floating point pipeline
- Allows for fast & high quality splatting
  - Deferred shading
  - EWA approximation