RENDERING SPECULAR EFFECTS

Anton S. Kaplanyan
1 August 2016
SPECULAR ALIASING

- Specular highlight can be small
  - Gets brighter when gets smaller
  - Specular microdetails sparkle

- Specular aliasing
  - Emphasized on curved geometry
  - Does not go away even after many samples

- Pixel footprint spans a large area on the surface
  - Stretched on curved surfaces
RENDERING SPECULAR

- Rendering equation: scattering at surface point $x$
  
  $$L_o(x) = \int_{\Omega} f_r(x, \omega_i, \omega_o) L_i(x) \, d\omega_i$$

- Flux incident at the image pixel is
  
  $$I_j = \int_{\mathcal{F}} W_j \, L_o(x) \, G \, dx$$

- Want to integrate over the pixel footprint $\mathcal{F}$
MICROFACET BSDF

- The BSDF $f_r(x, \omega_i, \omega_o)$ is a scattering function.
- Cook-Torrance microfacet BSDF is commonly used:
  $$f_r(x, \omega_i, \omega_o) = \frac{G(\omega_i, \omega_o)F(\omega_i, \omega_o)D(h)}{4|n \cdot \omega_i||n \cdot \omega_o|}$$
- Shadowing-masking and Fresnel ($G$ and $F$) are $[0;1]$ bounded.
- The Normal Distribution Function (NDF):
  - Density of actively reflecting microfacets given $h$.
  - Unbounded values!
Three-Points Transport

- The half vector depends on three adjacent vertices: $x, l, e$
- The three-point transport with integration of NDF is

$$I_j \approx C_2 \int_{\mathcal{F}} D(h(x, l, e)) dx$$
VARIATION OF HALF VECTOR

Flat

Curved
Materials with Specular Microdetails
Joint work with Tobias Zirr

Glints

Snow / Sand

Brushed Metal
CONCEPT OF A MICRODETAIL

- Correlated clusters
  - Correlation both in NDF and on surface
  - Isotropic or anisotropic
  - Model with nested distributions
  - Glints, grooves in brushed metal, etc.

Results from [Yan et al.14]
Discrete Stochastic Microfacet Models [Jakob et al.14]
- Hierarchical search
- In spatial and half vector (slope) domain

Rendering Glints on High-Resolution Normal-Mapped Specular Surfaces [Yan et al.14]
- Hierarchical pruning of a filtered micronormal map
- In half vector domain, parallel light and eye rays
PREVIOUS WORK: REAL-TIME

Sparkly but not too Sparkly! A Stable and Robust Procedural Sparkle Effect

- SIGGRAPH AiRT’15, EGSR’16 (Studio Gobo)
- 3D grid of sparkle kernels
  - Based on “Gettin’ procedural” [Shopf10]
- Sparse sparkles & glints

Labs R&D: Rendering Techniques in Rise of the Tomb Raider

- SIGGRAPH AiRT’15 (Eidos Montreal)
- Simple procedural noise for sparkles
Nested NDF distribution

- \( D_m \) selects a slope of a single microdetail
- \( D_l \) defines shape of a microdetail

\( D_g \) is a resulting global NDF

Convolution of \( D_m \) and \( D_l \)
AUTHORING WITH BISCALE NDFS

- Powerful artistic control:
  - Local roughness $D_l$ controls detail appearance
  - Global roughness $D_g$ controls distant appearance
COHERENT STOCHASTIC PROCESS

*Stable texture-space* power-of-two grids and anisotropic filtering:

- One binomial draw per grid cell
- Trilinear interpolation
VIEW DEPENDENCY (SHIMMERING)

- Search space 4D: Also need subdivision of microdetail orientations
- Paraboloid half vector grid
- Seed binomial using a 4D index
- Perturb half vector partitioning using texture grid index to avoid simultaneous change of sparkles
PERFORMANCE

- GeForce GTX 980, 1080p
- Maximum anisotropy: 16x

<table>
<thead>
<tr>
<th>Scene</th>
<th>Polys</th>
<th>Isotropic footprint, ms</th>
<th>Grazing angle, ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full-screen pass</td>
<td>2</td>
<td>0.9</td>
<td>2.9</td>
</tr>
<tr>
<td>Snow</td>
<td>32k</td>
<td>2.5</td>
<td>4.0</td>
</tr>
<tr>
<td>Dress</td>
<td>100k</td>
<td>1.4</td>
<td>4.4</td>
</tr>
<tr>
<td>Car (grooves)</td>
<td>570k</td>
<td>2.5</td>
<td>3.9</td>
</tr>
<tr>
<td>Crytek Sponza</td>
<td>262k</td>
<td>3.0</td>
<td>5.9</td>
</tr>
</tbody>
</table>

- ALU variance:
  - 8-64 cells to shade, 412 static instructions, 204 within a loop for one cell
  - No texture fetches
EXAMPLE CODE & RESULTS

Example available online: https://www.shadertoy.com/view/ldVGRh
Specular Antialiasing
Joint work with Stephen Hill, Anjul Patney, and Aaron Lefohn
METHOD SUMMARY (SPOILERS!)

- Accurate and stable filtering of specular highlight on the pixel footprint
- Simple, real-time, robust
- Compatible with common real-time methods
  - Deferred shading
  - Normal maps and filtering thereof (e.g., LEAN/CLEAN, vMF)
  - Support for Beckmann and GGX NDF models
- Requires models with high-quality shading normal
PREVIOUS WORK

Unstable

Stable

Stable

Stable

1 spp, α=0.01

[Vlachos15]

[Hill12]

Ours
The half vector depends on three adjacent vertices: \( x, l, e \)

The three-point transport with integration of NDF is

\[
I_j \approx C_2 \int_{\mathcal{F}} D(h(x, l, e))dx
\]

How does a half-vector \( h \) change w.r.t. \( x/l/e \)?

- Use first-order 2x2 derivative matrix \( M \) of \( h \) w.r.t. \( x/l/e \) [Jakob12]
TRANSFORMING THE PIXEL FOOTPRINT

- Obtain variation of slopes due to finite area of pixel footprint

- $M$ is a mapping of pixel footprint to half-vector domain!
  - First transform ray differentials into $\Delta x$ on surface
  - Then multiply by matrix $M$ to get vectors $\Delta h$ of $\varphi$ in slope domain
    \[
    \varphi \approx M(F)
    \]

- NDF filtering is then a 2D integration over the region $\varphi$!
CHANGE OF DOMAIN

We can filter the NDF based on the pixel footprint $\mathcal{F}$

$$\int_{\mathcal{F}} D(h(x)) dx = \int_{\mathcal{\partial}} D(h) \left| \frac{dh}{dx} \right| dh \approx \frac{|\mathcal{F}|}{|\mathcal{\partial}|} \int_{\mathcal{\partial}} D(h) dh$$
PRACTICAL NDF FILTERING

- How to compute $M$?
  - Benefit from quad shading on GPU!
    - Use $\ddx / \ddy$ to obtain the final value with finite differencing
    - Matrix $\varphi = dh/duv = M(F)$ is first-order change of $h$ induced by pixel footprint
    - Implicitly accounts for surface curvature with derivative of shading normal

- Robust temporal stability
  - Use an axis-aligned rectangle bounding the parallelogram
  - Aligned along $s$ and $t$ axes of the slope domain
  - Overfilters the NDF
void shade()
{
  ...
  // Compute plane-plane half vector in local shading frame (hpp)
  vec3 hppWS = hWS / dot(hWS, shadingNormal);
  vec2 hpp = vec2(dot(hppWS, shadingTangent), dot(hppWS, shadingBitangent));
}

ALGORITHM

Compute half-vector in slope domain
ALGORITHM

Compute half-vector in slope domain

Compute its ddx/ddy derivatives

```c
void shade()
{
    ...
    // Compute plane-plane half vector (hpp)
    vec3 hppWS = hWS / dot(hWS, shadingNormal);
    vec2 hpp = vec2(dot(hppWS, shadingTangent), dot(hppWS, shadingBitangent));
    // Use ddx/ddy, thanks to quad shading!
    mat2 dhduv = mat2(dFdx(hpp), dFdy(hpp));
    // Compute filtering rectangular region
    vec2 rectFp = min((abs(dhduv[0]) + abs(dhduv[1])) * 0.5f, 0.7f);
```
void shade() {

    // Compute plane-plane half vector (hpp)
    vec3 hppWS = hWS / dot(hWS, shadingNormal);
    vec2 hpp = vec2(dot(hppWS, shadingTangent), dot(hppWS, shadingBitangent));
    // Use ddx/ddy, thanks to quad shading!
    mat2 dhduv = mat2(dFdx(hpp), dFdy(hpp));
    // Compute filtering rectangular region
    vec2 rectFp = min((abs(dhduv[0]) + abs(dhduv[1])) * 0.5f, 0.7f);
    // Covariance matrix of pixel filter's Gaussian (remapped in roughness units)
    vec2 covMx = rectFp * rectFp * 2.f;
    roughness = sqrt(roughness * roughness + covMx); // Beckmann proxy convolution (for GGX)
RESULTS

1spp (rasterized)

512spp, ray-traced
RESULTS

1 sample/pixel 1 sample/pixel 128 samples per 1/4 ray tracing
no NDF filter NDF filtering
CONCLUSION

- NDF filtering for stable specular shading
  - Integrate highlight across pixel footprint on a local shading quadric
  - Preserve highlight energy, find difficult and small highlights
  - Compatible and orthogonal to other methods, simple and readily usable

- **Limitations**
  - Addresses only *shading* aliasing
    - No improvements for geometric aliasing
    - Can still alias with high-frequency bumpy geometry
  - Relies on *properly modeled* shading normals
TAKE HOME MESSAGE

Filtering of diffuse illumination is well-established

- Filter on surface, e.g., texture filtering

When filtering specular & glossy, always consider a 3-point transport

Specular constraint lives in local shading frame

- Half vector depends on all three vertices of the path
- Depends on the curvature of the surface
Thank you!

Q&A