ADVANCED SKIN SHADING WITH FACEWORKS

Nathan Reed — NVIDIA
March 24, 2014
DIGITAL IRA

• Tech demo
• Collaboration with Dr. Paul Debevec at USC
• Lots of other inspiring work on skin & eyes
  • [Penner10], [Jimenez12], [Jimenez13]
THE FACEWORKS API

- Middleware lib
- Integrate advanced skin shading in your game engine
- Goal: enable quality that matches Digital Ira
- This is a preview!
FEATURES

• Subsurface scattering (SSS)
  • One-pass solution based on [Penner10]
  • Both direct and ambient light

• Deep scatter
  • Based on thickness from shadow map [Green07]
  • Direct light only, so far

• D3D11 only, so far
SSS COMPONENTS

- Geometric curvature
- Normal maps
- Shadow edges
- Ambient
SSS 1/4: GEOMETRIC CURVATURE

- Precompute curvature per vertex (1 float)
  - Use bind pose of mesh
- Precompute a LUT (lookup texture)
  - For a given range of curvatures
  - Based on diffusion profile [d’Eon07]
- In PS, sample LUT based on curvature & N·L
SSS 2/4: NORMAL MAPS

• In PS, sample the normal map twice
  • Once as usual
  • Once with higher mip level
• Precompute mesh UV scale
• Combine lighting from both normals & curvature
SSS 3/4: SHADOWS

- Wide shadow filter
  - PCF, VSM, ESM, etc.
  - Not contact hardening - want consistent filter radius
- Precompute a LUT
  - Sharpens shadows
  - Adds red glow in shadowed area
- In PS, sample LUT based on shadow result
SSS 4/4: AMBIENT

- In PS, generate 3 normal vectors
  - Combine blurred & unblurred normals
- Eval ambient light for each normal
- Combine 3 lighting values
DEEP SCATTER

• Estimate thickness from shadow map
  • Helper functions for standard depth maps
  • Inward normal offset to fix silhouette edge issues
  • Poisson disk filter

• Apply falloff function using thickness & N·L

• Multiply by texture for veins, bones etc.
RECAP

Precomputed
• Per-vertex curvature
• Per-mesh UV scale
• Curvature LUT
• Shadow LUT

Pixel Shader
• Interpolate curvature
• Sample normal twice
• Wide shadow filter
• Eval ambient 3 times
• Estimate thickness
INTEGRATION

- C API & .dll, link into your engine & tools
  - Precompute data
  - Set graphics state per draw
- HLSL API, #include into your shaders
  - Evaluate lighting in PS
INTEGRATION

- Look up bind points using shader reflection
- Create context object
- Per draw: fill out config struct, set state
PERFORMANCE (lower is better)

<table>
<thead>
<tr>
<th>No FaceWorks</th>
<th>SSS</th>
<th>SSS + Deep (1 tap)</th>
<th>SSS + Deep (8 taps)</th>
<th>SSS + Deep (32 taps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>1.29</td>
<td>1.37</td>
<td>1.52</td>
<td>2.21</td>
</tr>
</tbody>
</table>
FUTURE WORK

• Ambient-light deep scatter
• Specular model for skin
  • Including occlusion
• Eye rendering
• Customizable diffusion profiles
• Support more APIs/platforms
  • OpenGL, Mobile, Consoles
INTEGRATION WALKTHROUGH

• C API & .dll, link into your engine & tools
  • Precompute data
  • Set graphics state per draw
• HLSL API, #include into your shaders
  • Evaluate lighting in PS
INTEGRATION WALKTHROUGH

• In tools:
  • Precompute curvature & UV scale for each model
  • Look up bind points for constant buffers etc. in shaders
  • Generate curvature & shadow LUTs
    ▪ Can also just be done once and checked in
INTEGRATION WALKTHROUGH

• In engine:
  • Create a context
  • Fill out config struct
  • Set state before draw call
INTEGRATION WALKTHROUGH

• In pixel shader:
  • Get mip level for blurred normal
  • Sample normal twice
  • Evaluate direct light
  • Evaluate shadow
  • Get 3 normals for ambient
  • Evaluate ambient light
  • Estimate thickness
  • Evaluate deep scatter
FEEDING THE RENDERER

• Need high-quality models and textures for good results!
  • LPS head: 18K tris (before tess), 4K textures
• Normal map should not be softened
  • Need strong bumps for specular
  • The shader will soften the diffuse
• Head scans are great, if within your means
EXTRA GOODIES IN THE SAMPLE APP

• Physically-based shading
  • Blinn-Phong NDF, Schlick-Smith visibility, Schlick Fresnel
  • 2-lobe specular [Jimenez13]

• IBL with preconvolved cubemaps [Lagarde11]

• Filmic tonemapping [Hable10]

• Adaptive tessellation
  • More polygons where curvature is high
REFERENCES

• Hable, “Uncharted 2 HDR Lighting”, GDC 2010
• Jimenez & von der Pahlen, “Next-Generation Character Rendering”, GDC 2013
• Lagarde, “Adopting a physically based shading model”, blog post (2011)
• Penner, “Pre-Integrated Skin Shading”, SIGGRAPH 2010