Practical Real-Time Hair Rendering and Shading

Thorsten Scheuermann

ATI Research
Outline

- Art assets
  - Hair model
  - Textures
- Shading
  - Kajiya-Kay
  - Marschner
- Depth sorting
  - Early-Z culling optimization
- Demo
Hair Model - Geometry

- Several layers of patches to approximate volumetric qualities of hair
- Per-vertex ambient occlusion term to approximate self-shadowing
Reasons for using a polygonal hair model:

• Lower geometric complexity than line rendering
  – Makes depth sorting faster
  – Pretty much a necessity for real-world use on current graphics hardware

• Integrates well into our art pipeline
Hair Model - Textures

- **Base texture**
  - Stretched noise
  - Hair color set in a shader constant
- **Alpha texture**
  - should have fully opaque regions
- **Specular shift texture**
- **Specular noise texture**
Hair Lighting: Kajiya-Kay

- Anisotropic strand lighting model
- Use hair strand tangent \( T \) instead of normal \( N \) in lighting equations
- Assumes hair normal to lie in plane spanned by \( T \) and view vector \( V \)
- Example: Specular \( N \cdot H \) term

\[
\sin(T, H)^{specularity} = \frac{\sqrt{1 - \text{dot}(T, H)^2}}{\text{specularity}}
\]
Hair Lighting: Marschner

- Based on measurements of hair scattering properties
- Observations
  - Primary specular highlight shifted towards hair tip
  - Secondary specular highlight
    - Colored
    - Shifted towards hair root
    - Sparkling appearance
- For simplicity we’re trying to match these observations phenomenologically
Hair Shader Implementation

Vertex Shader
- Just passes down tangent, normal, view vector, light vector, ambient occlusion term

Pixel Shader
- Diffuse Lighting
- Two shifted specular highlights
- Combines lighting terms
Diffuse Lighting Term

• Kajiya-Kay diffuse term $\sin(T, L)$ looks too bright without proper self-shadowing
• Instead, use scaled and biased $N \cdot L$ term:

$$\text{diffuse} = \max(0, 0.75 \times N \cdot L + 0.25)$$

• Brightens up areas facing away from the light when compared to plain $N \cdot L$ term
  – Simple subsurface scattering approximation
• Softer look
Shifting Specular Highlights

- Move tangent along patch normal direction to shift specular highlight along hair strand
- Assuming $T$ is pointing from root to tip:
  - Positive shift moves highlight towards tip
  - Negative shift moves highlight towards root
- Look up shift value from texture to break up uniform look over hair patches
Specular Strand Lighting

- Specular strand lighting using half-angle vector
- Compute two highlights with
  - Different colors
  - Different specular exponents
  - Differently shifted tangents
- Modulate secondary highlight with noise texture
- Specular highlights are attenuated by scaled and biased $N \cdot L$ term to account for lack of true self-shadowing
Combination of Lighting Terms

$$\text{finalColor} = (\text{diffuse} + \text{specular}_1 + \text{specular}_2) \ast \text{baseTexture} \ast \text{ambientOcclusion}$$
Comparison

Kajiya-Kay  Marschner et al.  Photograph  Our shader

(Left three pictures from [Marschner03])
Approximate Depth Sorting

- Back-to-front rendering order necessary for correct alpha-blending
- For a head with hair this is very similar to rendering from inside to outside
- Use static index buffer with inside to outside draw order
  - Computed at pre-process time
  - Sort connected components (hair strand patches) instead of individual triangles
Sorted Hair Rendering Scheme

- **Pass 1**: render opaque parts
  - Visibility resolved using z buffer
- **Pass 2**: render transparent back-facing parts
- **Pass 3**: render transparent front-facing parts
- Use alpha testing to distinguish between opaque and transparent parts in each pass
Taking Advantage of Early-Z Culling

- Early-Z culling allows skipping pixel shader execution for fragments that fail Z test
- Helps scenes with high depth complexity like the layered hair model
- Unfortunately early-Z culling is incompatible with alpha testing on our target hardware
- Replacing alpha tests with Z tests enables early-Z culling which speeds up hair rendering
Optimized Scheme: Pass 1

Prime Z buffer with depth of opaque hair regions

- Enable alpha test to only pass opaque pixels
- Disable backface culling
- Enable Z writes, set Z test to Less
- Disable color buffer writes
- Use simple pixel shader that only returns alpha

- No benefits of early-Z culling in this pass, but shader is very cheap anyway
Optimized Scheme: Pass 2

Render opaque regions

- Start using full hair pixel shader
- Disable backface culling
- Disable Z writes
- Set Z test to **Equal**
  - Z test passes only for fragments that wrote to Z in pass 1, which are the opaque regions

- This and subsequent passes don’t require alpha testing and thus benefit from early-Z culling
Optimized Scheme: Pass 3

Render transparent back-facing parts

• Cull front-facing polygons
• Disable Z writes
  – Z order isn’t necessarily correct
• Set Z test to Less
Optimized Scheme: Pass 4

Render transparent front-facing parts

• Cull back-facing polygons
• Enable Z writes
• Set Z test to Less

• Enabling Z writes prevents incorrect depth order at expense of possibly culling too much
• Covers up potential depth order artifacts of previous pass
Demo
Pros and Cons

Pros:
• Low geometric complexity
  – Reduced load on vertex engine
  – Makes depth sorting faster
• Easy fall-backs for lower-end hardware

Cons:
• Sorting scheme assumes little to no animation in hair model
  – Things like dangling pony tails need to be handled separately
  – Sort geometry at run-time to overcome this
• Not suitable for all hair styles
Conclusion

- Art assets
  - Polygon hair model
  - Textures
- Shading
  - Diffuse term
  - Two specular terms
  - Ambient occlusion term
- Approximate depth sorting
- Early-Z culling optimization