Practical Real-Time Hair Rendering and Shading

Thorsten Scheuermann*
ATI Research, Inc.

Introduction

We present a real-time algorithm for hair rendering using a polygon model, which was used in the real-time animation *Ruby: The Double Cross*, appearing in this year's SIGGRAPH animation festival. The hair shading model is based on the Kajiya-Kay model, and adds a real-time approximation of realistic specular highlights as observed by Marschner et al. We also describe a simple technique to render the semi-transparent hair model in approximate back-to-front order. Instead of executing a spatial sorting step on the CPU at run-time, we render the opaque and transparent hair regions in separate passes to resolve visibility.

Hair model

The hair model we use is built with layered 2D polygon patches which provides a simple approximation of the volumetric qualities of hair. The polygonal model is preferable to line rendering since it lowers the load on the vertex processor, and simplifies the problem of sorting the geometry from back-to-front for rendering.

The main textures on the model are a base map used to represent the fine structure of a bundle of hair, and a set of opacity maps used to give the 2D patches a less uniform appearance.

Hair shading

Diffuse lighting. For the diffuse lighting component, we use a scaled and biased $N \cdot L$ term: $diffuse = \max(0, 0.75 \times N \cdot L + 0.25)$. This brightens up the side of the hair facing away from the light and helps to create a softer overall look.

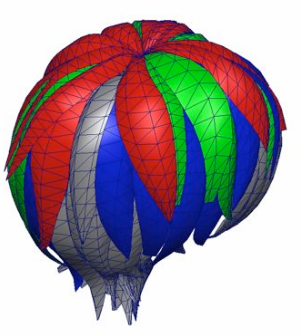
Specular lighting. Our basic approach for computing the specular component uses the Kajiya-Kay model. More recently, Marschner et al. reported that hair has two distinguishable highlights. The first one is from light reflecting off the surface, and is shifted towards the tip of the hair. The second highlight is from light that is transmitted into the hair strand and reflected back out towards the viewer. The color of this highlight is modulated by the hair's pigment color, and is shifted towards the hair root. It also has a noisy sparkling appearance.

To approximate the observations by Marschner et al., we compute two separate specular terms per light source. The two terms have different specular colors, different specular exponents and are shifted in opposite directions along the hair strand. We modulate the secondary highlight with a noise texture to achieve a very inexpensive approximation of the sparkling appearance observable in the secondary highlight in real hair.

To shift the specular terms, we perturb the hair tangent used in the lighting calculation along the hair patch normal direction: $T' = \text{normalize}(T + s \times N)$. A positive or negative value for the shift parameter s shifts the highlight towards the hair tip or root, respectively. In order to break up the uniform look of the highlights over the hair patches, we look up s in a texture. An alternative to looking up s per pixel would be to look up T in a "tangent map" analogous to the commonly used normal maps.

Ambient occlusion. As a simple approximation for hair self-shadowing, we store an ambient occlusion term per vertex. This term is computed in a pre-processing step.

*e-mail: thorsten@ati.com



For the final pixel shader output, we add the diffuse and specular terms and modulate the sum by the base texture and the ambient occlusion term.

Rendering with approximate depth sorting

To achieve correct blending of semi-transparent portions of the hair, we must render the hair model in roughly back-to-front order. During pre-processing, we sort the hair patches by their distance from the head and store the draw order in a static index buffer. At run-time, we render the model using four passes:

During the first pass, we prime the Z buffer for the opaque regions of the hair model. We use alpha-test to mask out the transparent parts, turn off back-face culling, disable writing to the color buffer, and render using a very simple pixel shader that only returns the hair's alpha channel.

In the following passes, we use the full hair pixel shader. For the second pass we set the Z test to equal so that the same opaque pixels as in the first pass get shaded. For the third pass we disable Z writing, enable front-face culling, and set the Z test to less. This draws all back-facing transparent hair regions. Finally, we re-enable Z writing, set the Z test to less, and enable back-face culling, so that all front-facing transparent regions are rendered.

Early-Z Culling. The reason we initialize the Z buffer in the first pass is to avoid using alpha-testing in the subsequent passes, because enabling this render state disables early-Z culling on the graphics processor. Taking advantage of early-Z culling to skip pixel shader execution on pixels that fail the Z test is a considerable performance win, and the extra rendering pass is more than offset by the gains of early-Z culling in the following three passes.

Our hair rendering scheme has the advantage of not requiring a spatial sorting step on the CPU at run-time. However, we assume that the hair model only undergoes very moderate animation that doesn't move the hair patches much relative to each other. If this assumption doesn't hold, it is possible to use a more robust CPU-based spatial sorting scheme.

References

KAJIYA, J. and KAY, T. 1989. Rendering Fur With Three Dimensional Textures. In *Computer Graphics (Proceedings of ACM SIGGRAPH 89),* 23, 3, ACM, 271-280.

MARSCHNER, S.R., JENSEN, H.W., CAMMARANO, M., WORLEY, S. and HANRAHAN, P. 2003. Light Scattering from Human Hair Fibers. *ACM Transactions on Graphics,* 22, 3, 780-791.