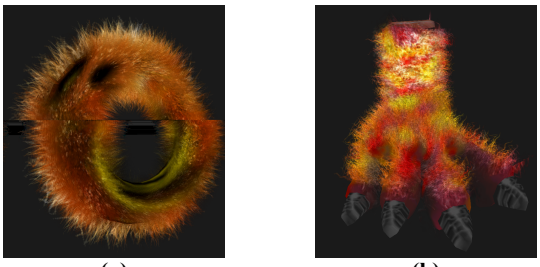
User Customizable Real-Time Fur

John Isidoro
Boston University and ATI Research
JIsidoro@ati.com

Jason L. Mitchell
ATI Research
JasonM@ati.com

Introduction

Recent advances in real-time fur rendering have enabled the development of more realistic furry characters. In this sketch, we outline a number of advances to the shell and fin based fur rendering technique by Lengyel et al [2001] using the pixel and vertex shader capabilities of modern 3D hardware.



(a) **(b)**

Figure 1 – Furry torus and animal claw

Shell and Fin extrusion using Vertex Shaders

Vertex shaders allow for more flexibility in fur rendering by performing the shell and fin extrusion in hardware. For shells, each shell vertex is moved in the direction of the normal a fraction of the fur height. For each of the fins, two of the four vertices are extruded in the same manner. The fin's v texture coordinate is used to choose which vertices get extruded. One of the nice things about extrusion in a vertex shader is that it allows for the fur length to be set using a vertex shader constant. A possible application of this is to have characters in a video game have shrinking or growing hair during the course of a game.

Per Pixel Anisotropic Lighting on Fins and Shells

One interesting property of stochastically generated fur is the distribution of hair directions throughout the texture. Due to the changing directions of curly hairs, it is no longer sufficient to derive hair direction for lighting from only the albedo texture. Instead, the hair direction is encoded per pixel along side the opacity value when the 3D shell and 2D fin fur textures are generated. When rendering the fur, the per-pixel direction can be used as a direction of anisotropy for Heidrich and Seidel's [1999] strand anisotropy technique. This allows individual hairs to capture diffuse and specular lighting. This helps to break up the homogeneity of the fur, producing a more realistic rendering.

Hair Color Sourcing (Shells)

Another technique we use to create more realistic fur is to apply an albedo texture that produces color variations for different regions of the fur. The alpha channel of this texture can also be used as an opacity value to produce bald spots on regions of the model. However, when the fur texture contains very wavy or slanted fur, each hair will change color over its length due to the fact the texture coordinate used to fetch from the albedo texture is directly tied to the hair's position. A better approach is to fetch the same texel from the albedo map over the length of the hair. This way, when there are sharp edges in the albedo map, the hairs from both sides of the edge will intermingle, giving a more realistic appearance. For example, the hairy foot in Figure 1b does not have fur on the claws. The hairs originating near the claws are able to bend over the claws while remaining visible and colored correctly.

Color Sourcing for Shells

To implement color sourcing for shells, another 3D texture is generated in parallel with the direction/opacity texture. This texture contains a (u,v) offset for each pixel with the hair which represents how far the (u,v) position of the hair is from the u,v position of the albedo of the hair. In the pixel shader used for rendering the shells, the albedo texture lookup is perturbed by this (u,v) so that each hair fetches from only one position in the albedo map. This way, each hair's albedo is the same over the length of the hair.

Color Sourcing for Fins

Color sourcing for the fin texture is performed in a similar manner, but requires additional per vertex data in order to perform the correct math. Because the fin fur texture is 2D, only a u offset is required per pixel. However, the fin fur texture coordinates are different from the albedo map texture coordinates, and an albedo map (u,v) needs to be computed from the fur texture's u offset. To do this, the albedo map du, dv for each fin is encoded per vertex, and interpolated over the polygon within the pixel shader. In the pixel shader, the fur texture's u offset is multiplied by the albedo map overall fin du, dv in order to get the hair's (u,v) offset for the albedo map.

Controlling Hair Density Using the Albedo Map

Another parameter that can be specified over the surface of the object using an albedo map is hair density. In order to do this, an additional texture channel is generated in parallel with the other fur texture channels. In this channel is a "thinning" value from 0.0 to 1.0 which specifies the percentage of the fur texture left to be generated. The first hair generated gets a value of 1.0, and the last hair generated gets a value of 0.0. In the pixel shader, a pixel is only written if its thinning value is less than or equal to the albedo map's density value (usually stored in the alpha channel of the albedo map). This density value causes a percentage of the hairs to not be drawn, thus thinning the fur.

Hole Filling for Mipmap Generation

Simply down-sampling the color-encoded direction and offset textures causes incorrect results due to the unfilled pixels in the texture. In order to get around this, each unfilled pixel in the texture is replaced by a weighted average of the filled pixels within a small kernel radius. This results in a texture which varies smoothly, and can be down-sampled without producing artifacts in the lighting or albedo map offsets.

Future Enhancements

Since both the fins and shells are extruded along the direction of the vertex normal in the vertex shader, it is possible to animate the fur by perturbing the normals. This perturbation could be performed using a vertex tweening sequence or procedurally inside a vertex shader.

References

HEIDRICH, W. AND SEIDEL, H.-P. Realistic, Hardware-accelerated Shading and Lighting. In *Proceedings of SIGGRAPH 1999*, ACM Press / ACM SIGGRAPH, New York. 171-178.

LENGYEL, J., PRAUN, E., FINKELSTEIN, A. AND HOPPE, H. Real-Time Fur over Arbitrary Surfaces. In *ACM 2001 Symposium on Interactive 3D Graphics*, 2001.