Rendering Dynamic Wet Fur Using Layered Textures

Paulo Silva[†] 坂東 洋介[†‡] 西田 友是[†]
東京大学[†] 東芝[‡]

1 Introduction

In computer graphics, textures are an important tool to model small scale detail. Fur falls is one such case. Due to its large numbers, fur can be hard to render in real-time, if done strand by strand.

There are several methods that make use of textures to render fur (see Sec. 2). Textures, by nature, work well in representing static detail. But, in real life surfaces covered by fur tend to be fluffy and soft. However, previous methods only render the fur texture as if it was static detail.

This paper describes a method to control the shape of fur represented by texture layers. Our target is to obtain an effect similar to wet fur clumping. Our implements fur manipulation completely on the GPU through texture coordinates warping. We use a 3D texture for fur geometry, a 2D texture for fur color, and a 2D texture for a mask representing fur clumping areas.

Our contribution is a GPU oriented method, for fur shape manipulation, that can render fur clumps similar to wet fur in real-time.

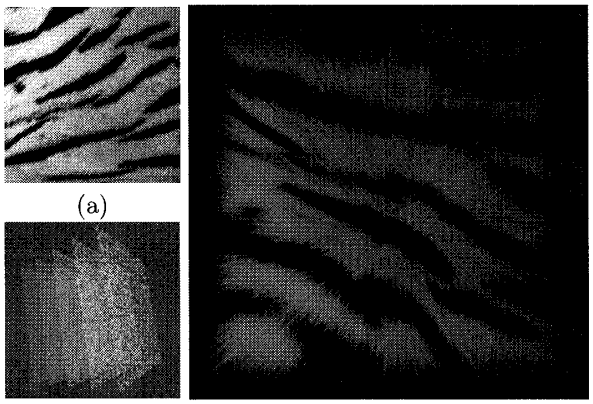
2 Related Work

Lengyel et al. [1] presented a real-time method using texture layers. This method can cover arbitrary surfaces by repeatedly pasting and blending patches of fur texture onto a base surface, until the surface is covered.

A method for rendering wet fur was proposed by Bruderlin [2]. This method concentrates primarily on the shape the fur takes when wet. However this method is not oriented for real-time rendering. Our method combines Bruderlin's method with Lengyel's method on a GPU implementation.

3 Our Method

Our method uses as input a mesh and a fur color texture (Fig. 1 (a)). Then, we create the fur geometry texture layers (Fig. 1 (b)) based on Lengyel et al.'s [1] method. Additionally, we create a clump mask texture (Fig. 2). This mask contains the clump regions information. That is, the center of the clump c, its radius r, and its wetness or clump-percent ρ (see Sec. 3.1). To render the fur layers we implemented a modified version of Lengyel et al.'s [1] method on the GPU. During rendering, in the fragment shader, we use the clamp mask to find if a strand is in a clump region. If so, we compute the displacement $\vec{d}$, this strand should suffer due to the clumping effect (see Sec. 3.2). Using this displacement we transform the texture coordinates of the color and fur textures. This creates the desired clumping as shown in Sec. 4.



(a) (b) (c)

Figure 1: Image (a) is the input color texture pattern, image (b) is the fur geometry texture layers, and (c) is the result of applying (a) to (b).

3.1 Clump Mask Generation

For the clump mask (Fig. 2), we create an additional texture. Here we assume a parameterization from the model to the texture is available. Then, to add a clump region, we use an idea similar to Bruderlin's [2]. For a clump located at surface parameter $c = (u, v)$ with radius r, and a clump-percent $\rho \in [0, 1]$, the center position on the model is computed from the parameterization, and written to the clump mask in the texture channels (R, G). Additionally we write (r, ρ) to texture channels (B, A).

3.2 Rendering Fur Clumps

The final fur clumps are rendered using the three above mentioned textures. First we check if the area to render is within a clump region. This is done by checking the clump mask (Fig. 2) using the model parameterization. Then, we compute a displacement $\vec{d}$ using Eq. 1.

$$\vec{d} = h\rho r(\vec{p} - \vec{c})/\|\vec{p} - \vec{c}\| \quad (1)$$

Paulo Silva[†], Yosuke Bando[†‡], Tomoyuki Nishita[†], The University of Tokyo[†], Toshiba[‡]

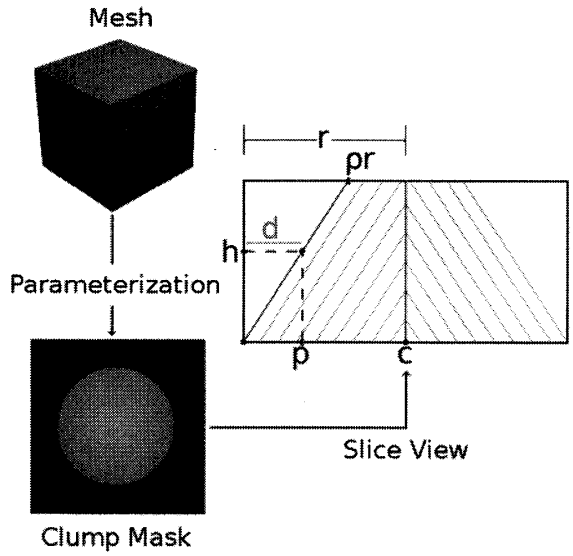


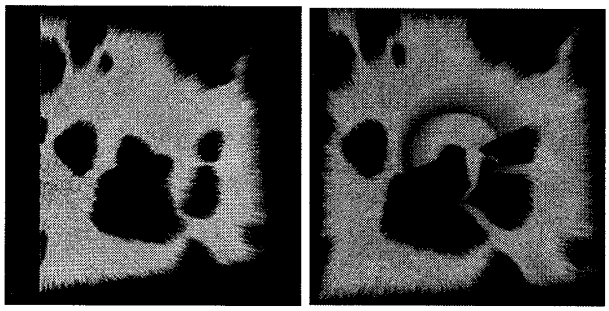
Figure 2: Using the mesh parameterization, and a clump mask we compute the displacement $\vec{d}$. The black area in the clump mask represents dry fur. The colored area represents the distance to center of the clump. In the slice view we can see: $\vec{d}$ is the displacement vector, $\vec{p}$ is the current position in the clump mask, $\vec{c}$ is the center of the clump, h is the height of the current texture layer, r is the radius of the clump, and ρ is the clumping percentage.

where, $h \in [0,1]$ is the current height, ρ is the wetness, r is the clump radius (we only consider circular clumps), $\vec{p}$ is the current pixel position on the clamp texture, and $\vec{c}$ is the center of the clump. The displacement $\vec{d}$ is illustrated on the slice view in Fig. 2. Then, we add $\vec{d}$ to the texture coordinates of the color, fur and clump textures. If the ρ read from the clump texture (Fig. 2) at this new position is not 0 (means the location is wet), we draw the fur using the displaced coordinates. Otherwise we discard the fragment because it no longer belongs to a clump region.

4 Results

In Fig. 3 (a) we can see a rendering of normal fur, which is similar to previous methods. In Fig. 3 (b) we can see an example using one clump region centered on the mesh.

Table 1 shows the performance statistics of our method. The resolution of the wet mask and fur texture layers is 256×256. The system used was an *Intel Quad Core 3GHz*, with 2GB of main memory and a *Nvidia GeForce 8800 GT* GPU with 512MB of memory. Since the dry fur rendering is equivalent to previous methods, we can conclude that our method consumes a minimal amount of all the rendering process. Finally, the performance of our method in frames per second, is linear in the number of layers, screen resolution, and changes very little for different fur densities.



(a) (b)

Figure 3: Combining the dry fur in (a) with the clumping mask Fig. 2, we obtain our final clumping fur effect seen in (c).

Table 1: Frames per second for several screen resolutions, number of layers, and clump states. The fur density for rows 1 to 8 is $256^2/2$ strands, and for rows 9 and 10 is $256^2/8$ strands. The fur density needs to be less or equal to the fur layers resolution.

#Layers	Screen	Speed (fps)	Clump
16	800×600	613	no
64	800×600	165	no
16	1680×1050	231	no
64	1680×1050	64	no
16	800×600	570	yes
64	800×600	155	yes
16	1680×1050	220	yes
64	1680×1050	61	yes
16	800×600	634	no
16	800×600	593	yes

5 Conclusions and Future Work

We have presented a method to create clumps of fur, that simulate a wet fur effect. Our method leverages previous methods for rendering fur based on texture layers, adding a simple and GPU friendly technique, which provides high frame rates.

As future work, we think our method can serve as a rendering base for other techniques, such as fur and water interaction.

References

[1] J. Lengyel, E. Praun, A. Finkelstein, and H. Hoppe, "Real-time fur over arbitrary surfaces," *In Proc. of Simposium of Interactive 3D Graphics'01*, pp. 227–232, 2001.

[2] A. Bruderlin, "A method to generate wet and broken-up animal fur," *The Journal of Visualization and Computer Animation*, vol. 11, no. 5, pp. 249–259, 2000.