13. Techniques for Rendering Filiform Objects

André M. LeBlanc and Russell Turner
Computer Graphics Lab
Swiss Federal Institute of Technology
Lausanne, Switzerland

13.1. Introduction

Filiform structures occur in the natural environment in various forms and embrace a wide variety of phenomena. They are, by definition, very thin threadlike objects that yield, in aggregate form, highly complex and somewhat nebulous volumes such as fur, hair and fibrous material. Larger and more detailed geometry such as grass, plants or tree branches may also exhibit the same filiform structure when seen at a greater distance. The finer geometry recedes as a result of distance and gradually becomes invisible to the eye. The larger structures, however, remain and define the aggregate form of the object. Another important source of filiform phenomena result from dynamic particle motion blurring. The trajectory of small, fast moving particles during a single animation frame is traced to yield a trailing filament.

This chapter divides the filament rendering problem into two camps. The first tries to resolve the problem by sampling the geometry in object space, on a pixel by pixel basis. These are essentially ray-tracing methods. The second approach uses image buffer techniques, which operate on the entire pixel array simultaneously. Here, individual objects are projected onto the pixels and composited, object by object, to arrive at a final picture.

13.2. Sampling geometry

The most obvious, brute-force approach to rendering filiform objects is to model each individual strand as a curved cylinder, and attempt to render them as surfaces. This immediately runs into several serious problems. For most practical scenes, a strand width is quite a bit less than the size of a pixel, resulting in a serious aliasing problem. Furthermore, the sheer number of cylinder primitives, together with the large amount of oversampling necessary to overcome aliasing will overwhelm most raytracing programs.

Therefore, it is not surprising that most successful attempts at rendering filiform objects with raytracing have been achieved by categorically avoiding explicit models of geometry. Instead, they rely on three-dimensional textures to provide the necessary detail and give only an illusion of geometry.

13.2.1. Fur using Raytracing

Miller (1988a, 1988b) rendered a furry caterpillar by raytracing an image made of explicit hairs. Each hair was modeled with triangles arranged in the shape of a pyramid. Oversampling was used to avoid aliasing but apparently no illumination or shadowing were used, only color mapping. Although the number of hairs was relatively small and their thickness large, this technique was nonetheless rather computationally intense.

13.2.2. Fur with Texels

A texel is a type of model intermediate between a surface texture and geometry. It is defined as a three-dimensional array, where each element holds information on density, local orientation of micro-geometry and a lighting model. This information is distributed freely in the array to model how light interacts with highly detailed geometry. Kajiya and Kay (1989) used this technique to render fur on a teddy bear. Texel arrays were modeled and laid out on the surface of the bear.

13.2.2.1. Texel Illumination Model for Hair

Kajiya and Kay (1989) defined an illumination model (Kajiya 1985) that follows directly from the underlying cylindrical nature of a hair strand. Figure 13.1 shows a segment of hair and the important vector quantities used in the model. The unit tangent vector, $\vec{t}$, represents the direction of the hair's axis. The light vector, $\vec{l}$, points in the direction of the light source. The reflection vector, $\vec{r}$, points in the direction of reflected light and is the direction of maximum specular reflection. The eye vector, $\vec{e}$, points in the direction of the viewer. The angles θ and φ are the angles from the tangent vector to the light vector and eye vectors respectively.

Diffuse Component

The diffuse component of the reflectance can be obtained by integrating the Lambert cosine law along the illuminated half of the cylinder to yield a simple function of the angle θ:

$$\text{Diffuse Reflectance} = K_D \sin(\theta) \qquad (13.1)$$

where K_D attenuates the reflected intensity as a function of the radius of the cylinder.

This does not, however, take into account self-shadowing, that is, the fact that half of the hair strand is in its own shadow.

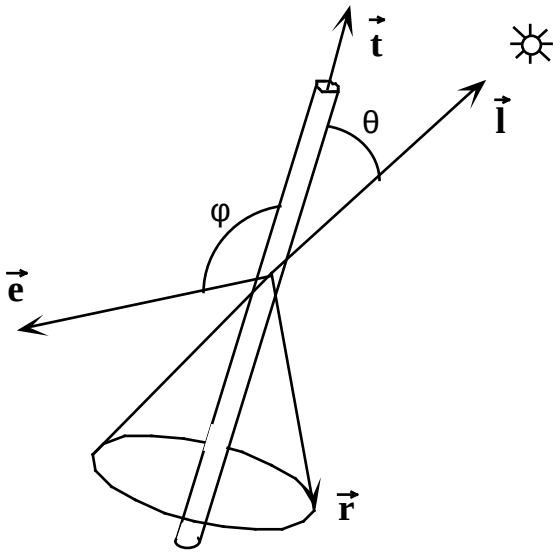

Figure 13.1. Vector quantities

Specular Component
The specular component is derived essentially by taking a Phong specular distribution and rotating it around the hair axis. This is motivated by making a symmetry-based argument. At any section along the length of the hair, there are surface normals pointing in all directions perpendicular to the tangent vector, $\vec{t}$. Therefore, the surface of the hair will reflect light in a cone-shaped 360 degree radial pattern formed by rotating the $\vec{r}$ vector around the hair axis. This cone represents the angles of maximum specular reflection. The Phong specular component is calculated by taking the angle between this cone and the eye vector, $\vec{e}$, which equals $\varphi + \theta - \pi$, and using this as the angle for the standard Phong equation:

$$\text{Specular Reflectance} = K_S \cos^n(\varphi + \theta - \pi) \qquad (13.2)$$

13.2.2.2. Rendering Texels

Texels are rendered using raytracing. A ray, emanating from the eye, intersects a texel array. From this point of intersection, the ray marches through the texel accumulating the reflected intensities from each visited array element. Shadowing is rendered by spawning a second ray towards the light at each element.

13.2.3. Fur with Hypertextures

Perlin and Hoffert (1989) employed volume densities to model soft furlike objects. A volume region is defined in space in which geometry is modeled as pseudo-random density functions. No illumination or shadowing was modeled.

13.3. Image Buffer Techniques

Instead of sampling geometry on a pixel by pixel basis, image buffer techniques approach the rendering problem on an object by object basis. The primary advantage of rendering a scene this way comes from the assurance that every object, however small it may be, will somehow contribute to the final image. There is no risk of undersampling geometry or accidently missing objects. In animations, this can limit the flickering effect caused by small objects that keep poping in and out of a sampling ray's path.

Image buffer techniques resemble, in some sense, the classic special effects photographic process by which individual objects are projected onto film and then composited together at a later stage into a final picture. The computer simply adds flexibility to this process and automates compositing operations.

Porter and Duff (1984; Duff 1985) defined a compositing algebra to combine and blend raster images. This compositing technique, sometimes called alpha-blending (Carpenter 1984), allows raster operations to be performed on a frame buffer according to an alpha channel. The alpha channel is an additional component of each pixel, along with the red, green and blue components, which can be used to control the amount by which an incoming image is blended with the one already present in the frame buffer.

Hidden surface removal via frame buffers is done by storing depth values at each pixel into a z-buffer. These values are compared with incoming pixels to keep the nearest ones and discard the farthest (hidden) ones.

One of the advantages of using alpha-blending and z-buffer techniques is that they are easily implemented in hardware and many advanced frame-buffer architectures incorporate alpha channel and z-buffer bitplanes. These machines support pixel compositing operations and hidden surface removal.

13.3.1. Hair using Z-Buffer

Csuri et al. (1979) rendered fur-like volumes by modeling each hair as a single triangle laid out on a surface. A z-buffer algorithm was used for hidden surface removal. No antialiasing or shadowing was done.

Watanabe and Suenaga (1989) modeled human hairs as connected segments of triangular prisms and were able to render a full head of straight human hair in a reasonably short time using a hardware z-buffer renderer with Gouraud shading. Although the hair model had a realistic number of hairs (more than a million primitives), the illumination model was quite simplistic and apparently no attempt was made to deal with aliasing. As a result, the images have a stiff, wire brush-like appearance.

13.3.2. Grass and Fire using Particle Systems

The term "particle systems" was first used by Reeves (1983) but now encompasses a range of techniques (Reeves and Blau 1985, Sims 1990, Smith 1984). Fundamentally, a particle system is an animation technique based on dimensionless points which represent very small

dynamic objects. Since the original purpose of the technique was to animate particles of fire in motion (for a sequence in the film *Star Trek II: The Wrath of Khan*), the points were displaced along their paths of motion during one animation frame to simulate motion blur, yielding thin three-dimensional line segments.

Later, the same technique was used to represent static images with thin filaments, such as grass (Reeves and Blau 1985). As a result, the term "particle" often refers, in fact, to a filament which may have a considerable length.

A particle system is rendered by painting each particle in succession onto the frame buffer. The particle's color contribution is calculated and combined with the pixel's existing color via alpha-blending.

13.3.2.1. Shadow Projection

Using particle systems and a frame buffer, (Reeves and Blau 1985) were able to render an antialiased, natural looking field of grass containing over half a million particles in a reasonable amount of CPU time. The technique, however, has several limitations. In particular, shadowing is limited, using a stochastic model for local self-shadowing and a simple texture map projected onto the grassy field for shadows of external objects.

This consists of projecting a two-dimensional shadow pattern onto an object from the point of view of each light. This method is limited to certain geometrical configurations where the three-dimensional nature of the shadow volume cast by objects is not significant.

13.3.3. Shadow-Buffers and Pixel-Blending

Kajiya and Kay (1989) demonstrated the importance of shadows in rendering realistic looking hair. Hair presents a wide variety of shadow effects due to the more complex geometry and the interaction between the hairless objects (the background scene) and the hair. To render naturalistic hair, shadows must be cast not only from hair onto hair, but also from the hair onto the hairless objects and from the hairless objects onto the hair.

LeBlanc et al. (1991) introduced a hybrid rendering method that enables shadows of hair to be incorporated into any scene. This method makes use of both raytracing and image buffer techniques to generate images in a multi-step pipeline — hairless objects are handled by raytracing and hair is incorporated among the raytraced objects as a post-processing step. In both steps, shadowing is handled with shadow buffers.

13.3.3.1. Shadows with Shadow Buffers

Shadow buffers are created by projecting a scene onto a pixel array from the point of view of the light source (Williams 1978). Only one piece of information, the depth or z value, is stored for each pixel. This results is a two-dimensional array which records the regions of space which are not visible from the light source and therefore in shadow. One shadow buffer is necessary for each light source together with the projection matrix used to create it.

With this information, it is possible to render an image with shadows by taking a point on a surface and transforming it into the shadow buffer's coordinates system. If the projected depth is greater than the depth in the shadow buffer, the pixel is considered to be in shadow for that light source.

Reeves et al. (1987) proposed a technique to produce softer shadows (penumbra), called percentage closer filtering. In this technique, (as in Williams 1978), a point on a surface is transformed to the shadow buffer's coordinates system and its depth is computed. It is then compared with a *region* of depth values in the shadow buffer to yield a value that indicates the percentage of shadowing for that region. This results in shadows with soft edges which resemble penumbra. The degree of softness can be varied to simulate a diffuse light source by enlarging or reducing the area of the sample region.

Rendering Shadows

Rendering hair with shadows requires that shadow buffers of the hairless objects be generated for each light source (Figure 13.2a, see color section). These shadow buffers will handle all shadows coming from the hairless objects.

To cast shadows from the hair onto the scene requires an augmented shadow buffer that also includes hair. This is achieved quickly by using a hardware-based line-drawing algorithm together with z-buffer hidden surface removal. Hair segments are fed to the graphics hardware resulting in a depth-map, as seen from a light source (Figure 13.2b, see color section).

The two shadow buffers are then combined into one. At each pixel, the depth values are compared and the lower value is retained, resulting in a scene-and-hair composite shadow buffer (Figure 13.2c, see color section). The composite scene-and-hair shadow buffer will then be used to determine the shadowing percentages, resulting in shadows cast by all objects in the scene, including hair.

The composite scene-and-hair shadow buffer is then fed into the raytracing renderer, generating a scene with hair shadows but no hair (Figure 13.2d, see color section). As discussed below in section 13.3.3.2, hair with shadows will then be blended into the scene to create a final image with full shadowing (Figure 13.2f, see color section).

Lighting Model with Shadows

Taking the hair intensity equation defined by Kajiya and Kay (1989) (see Section 13.2.2.1), the diffuse and specular components, together with ambient and shadowing components result in the following reflected intensity equation:

$$I = K_A + \sum_i S_i L_i (K_D \sin(\theta) + K_S \cos^n(\varphi + \theta - \pi)) \qquad (13.3)$$

where I is the total reflected intensity, L_i the intensity of light i, S_i the percentage of shadowing from L_i's composite scene-and-hair shadow buffer and K_A an ambient term.

13.3.3.2. Pixel Blending

A strand of hair, which is basically a long curved cylinder, is represented as a three-dimensional curve (polyline of segments) together with a radius. The complete hairstyle data

structure therefore consists of a very large number (often over one million) of three-dimensional line segments together with a scalar thickness value.

The next task is to render the individual hairs into the hairless image (Figure 13.2d, see color section) in a manner such that aliasing artifacts are avoided. In a normal viewing configuration, the thickness of a human hair is much less than half a pixel width and tens or hundreds of hair strands contribute to the intensity of a single pixel. Antialiasing is accomplished using a pixel blending technique which is similar to a method developed for particle system rendering.

The Aliasing Problem

A straightforward solution to the aliasing problem is to treat each pixel as a square and then consider that portion of the hair strand passing through the projection of the pixel. The light intensity contributed by that segment of the hair strand within the boundaries of the pixel projection is considered to be that hair's contribution to the pixel intensity.

Therefore, to correctly render a single pixel, the light intensity contributed by every hair passing through the projection of the pixel must be blended together properly, along with the underlying hairless image, to form the final pixel intensity. This process is called pixel blending.

A hair strand's intensity contribution to the pixel is equal simply to the intensity of the hair, I, times the fractional area of the pixel that the hair occupies. The total color value of the pixel P therefore is:

$$P = F_B B + \sum_i F_j I_j \qquad (13.4)$$

where B is the intensity of the pixel background image, F_B is the fraction of pixel area covered by the background (hairless) image, I_j is the hair intensity of hair j, and F_j is the fraction of pixel area covered by hair j.

Pixel Blending

If the pixel intensity is initialized to the background value, then each hair strand's intensity can be blended into the pixel in back-to-front order in proportion to its projected surface area. The intensity of the pixel P_n after blending in the nth hair strand is therefore:

$$P_0 = B ,$$

$$P_n = (1 - A_n)P_{n-1} + A_n I_n \qquad (13.5)$$

where B is the background image intensity, I_n is the intensity of the nth hair and A_n is the area of the hair strand's projection.

Rendering Segments

The pixel blending based algorithm requires that, for each pixel, the hair strands are blended in back-to-front order. Also, only visible hair strands should be blended, that is, strands

obscured by hairless objects in the background image are not visible and therefore should not be blended.

This requires depth information of each pixel in the hairless image (Figure 13.2e) and the depth of each strand. A common z-buffer hidden surface removal is used to discard hidden hair pixels. The remaining strands' pixels are blended into the image.

Rendering a hair strand in this manner takes advantage of standard antialiased line-rendering algorithms and z-buffer capabilities which have been implemented in hardware in many commercially available graphics hardware systems. It is therefore a relatively fast method for rendering antialised hairs with full shadowing.

References

Carpenter L (1984) The A-buffer, an Antialiased Hidden Surface Method, *Proc. SIGGRAPH '84, Computer Graphics,* Vol.18, No3, pp. 103-108.

Csuri C, Hakathorn R, Parent R, Carlson W, Howard M (1979) Towards an interactive high visual complexity animation system, *Proc. SIGGRAPH '79, Computer Graphics,* Vol.13, No2, 1979, pp. 289-299.

Duff T (1985) Compositing 3-D Rendered Images, *Proc. SIGGRAPH '85, Computer Graphics,* Vol.19, No3, pp. 253-259.

Kajiya JT (1985) Anisotropic Reflection Models, *Proc. SIGGRAPH '85, Computer Graphics,* Vol.19, No3, pp. 15-21.

Kajiya JT, Kay TL (1989) Rendering Fur with Three Dimentional Textures, *Proc. SIGGRAPH '89, Computer Graphics,* Vol.23, No3, pp. 271-280.

LeBlanc A, Turner R, Thalmann D (1991) Rendering Hair using Pixel-Blending and Shadow-Buffers, *Journal of Vizualisation and Computer Animation,* Vol.2, No3.

Miller GSP (1988a) From Wire-Frame to Furry Animals, *Proc. Graphics Interface '88,* pp. 138-146.

Miller GSP (1988b) The Motion Dynamics of Snakes and Worms, *Proc. SIGGRAPH '88, Computer Graphics,* Vol.22, No4, pp. 169-178.

Perlin K, Hoffert E (1989) Hypertexture, *Proc. SIGGRAPH '89, Computer Graphics,* Vol.23, No3, pp. 253-262.

Porter T, Duff T (1984) Compositing Digital Images, *Proc. SIGGRAPH '84, Computer Graphcis,* Vol.18, No3, pp. 253-259.

Reeves WT (1983) Particle Systems - A Technique for Modeling a Class of Fuzzy Objects, *Proc. SIGGRAPH '83, Computer Graphics*, Vol.17, No3, pp. 359-376.

Reeves WT, Blau R (1985) Approximate and Probabilistic Algorithm for Shading and Rendering Structured Particle Systems, *Proc. SIGGRAPH '85, Computer Graphics,* Vol.19, No3, pp. 313-322.

Reeves WT, Salesin DH, Cook RL (1987) Rendering Antialiased Shadows with Depth Maps, *Proc. SIGGRAPH '87, Computer Graphics*, Vol.21, No4, pp. 283-291.

Sims K (1990) Particle Animation and Rendering Using Data Parallel Computation, *Proc. SIGGRAPH '90, Computer Graphics,* Vol.24, No4, pp. 405-413.

Smith AR (1984) Plants, Fractals and Formal Languages, *Proc. SIGGRAPH '84, Computer Graphics,* vol.18, No3, pp. 1-10.

Watanabe Y, Suenaga Y (1989) Drawing Human Hair Using Wisp Model, *Proc. Computer Graphics International '89,* pp. 691-700.

Williams L (1978) Casting Curved Shadows on Curved Surfaces, *Proc. SIGGRAPH '78, Computer Graphics,* Vol.12, No3, pp. 270-274.