Interactive Multiresolution Hair Modeling and Editing

Tae-Yong Kim and Ulrich Neumann

{taeyong | uneumann}@graphics.usc.edu

Computer Graphics and Immersive Technologies Laboratory
Integrated Media Systems Center
University of Southern California

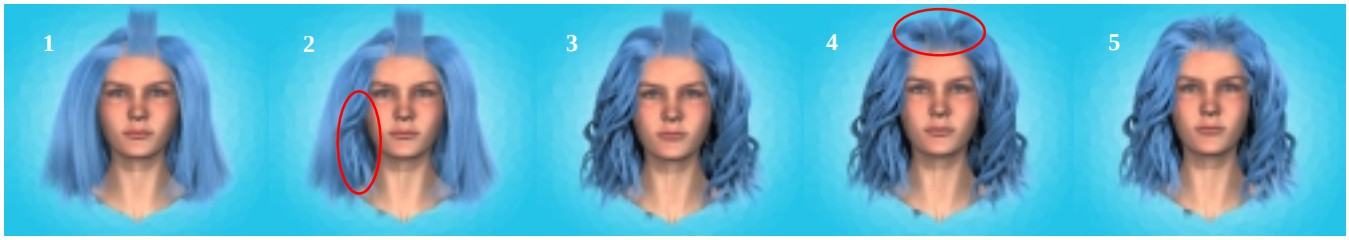


Figure 1. An example multiresolution hair design procedure. Each hair model results from interactive multiresolution editing operations. 1. The user roughly designs a hairstyle with about 30 high-level clusters. 2. One hair cluster (inside the ellipse) is subdivided and made curly. 3. The curly cluster is copied onto other clusters. 4. The bang hair cluster (inside the ellipse) is subdivided and refined. 5. Final hair model after further refinements.

Abstract

Human hair modeling is a difficult task. This paper presents a constructive hair modeling system with which users can sculpt a wide variety of hairstyles. Our Multiresolution Hair Modeling (MHM) system is based on the observed tendency of adjacent hair strands to form clusters at multiple scales due to static attraction. In our system, initial hair designs are quickly created with a small set of hair clusters. Refinements at finer levels are achieved by subdividing these initial hair clusters. Users can edit an evolving model at any level of detail, down to a single hair strand. High level editing tools support curling, scaling, and copy/paste, enabling users to rapidly create widely varying hairstyles. Editing ease and model realism are enhanced by efficient hair rendering, shading, antialiasing, and shadowing algorithms.

CR Categories and Subject Descriptions: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling – Modeling Package

Additional Keywords: hair modeling, multiresolution modeling, level of detail, hair rendering, generalized cylinders

1 INTRODUCTION

Hairstyle is a determining factor of a person's first impression when meeting someone [LaFrance 2001]. Thus, hair is an important aspect of personal identity, but hair modeling remains a major obstacle in realistic human face synthesis. The volumetric nature of hair is simply not captured by surface models, so it is often simplified or hidden by objects like hats.

Surface modeling methods can use high-density data acquired from range scanners. Such model acquisitions are not yet practical for hair since we lack suitable volumetric scanner technologies. The recent work by Grabli et al. [2002] attempts to automatically reconstruct hair models from photographs. Although promising, the approach is still in its inception stage and does not recover complete hair models. Our goal is an interactive hair modeling system that allows a user to easily and quickly design a wide range of hairstyles (Figure 1 illustrates an example hairstyling process).

Cluster (or wisp) hair models [Chen et al. 1999; Falk and Sand 2001; Kim and Neumann 2000; Plante et al. 2001; Watanabe and Suenaga 1992; Xu and Yang 2001; Yang et al. 2000] exploit the observed tendency of adjacent hair strands to form clusters due to static attraction or artificial styling. These models employ two-step manipulation. 1) The rough geometry of a hair cluster is modeled. 2) Details are added by rendering each hair strand or volume density. These models, however, lack stylistic variations in each hair strand due to a limited set of parameters, especially, for complex variations such as curly hair, as noted by Xu and Yang [2001].

Strand hair models [Anjyo et al. 1992; Daldegan et al. 1993; Hadap and Thalmann 2001; Lee and Ko 2001; Rosenblum et al. 1991] allow every hair strand to be explicitly designed. However, manual modeling of individual hair strands is extremely tedious. Designing just the key hair strands can consume five to ten hours [Thalmann and Hadap 2000]. Thus, strand hair models are often coupled with dynamics simulations. External parameters such as gravity, wind forces, and stiffness affect the global shape of the final hair model. However, the dynamics methods are often limited to relatively simple hairstyles and complex hairstyles are not easily modeled, even after repeated trial and error iterations of minutes or hours of simulation.

Problems arise from the complexity of real human hairstyles, often created by hairstylists with extensive efforts such as curling and combing (Figure 2). Structured and discontinuous clusters (e.g., braids, combing effects) are difficult to model with strand hair models, whereas stylistic strand variations are difficult to achieve with cluster/wisp hair models. Our *multiresolution hair*

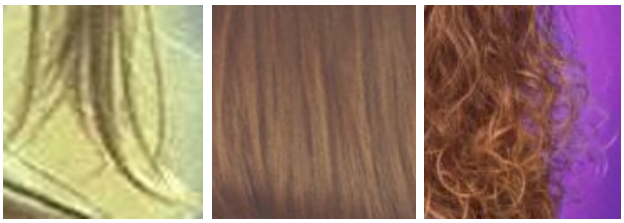


Figure 2. Photographs of human hair.

model aims at bridging this gap by allowing detailed local control as well as global control.

To model complex objects, researchers successfully employ multiresolution concepts for continuous curves, surface editing, and volume sculpting (see [Stollnitz et al. 1996] for examples). Their major benefit is the user's freedom to choose the appropriate level of detail for a desired model manipulation. In this spirit, we develop a Multiresolution Hair Modeling (MHM) system. However, hair models are inherently volumetric and contain a large number of disjoint hair strands. Thus, our treatment differs from other multiresolution techniques in applying the concept of multiresolution to hair. In our context, multiresolution manipulations are achieved with a hierarchy of generalized cylinders. MHM allows users to interactively move, scale, curl, and copy a portion of an evolving hair model at any level of detail.

Rendering is considered a separate offline process in existing hair modeling systems. However, such non-interactive processes can significantly slow down the modeling process, especially for complex hairstyles for which rendering effects such as self-shadowing are crucial. We aim at providing an interactive rendering capability, enabling users to get immediate visual feedback during modeling. To our knowledge, no reported hair modeling systems interactively render explicit hair models, complete with shading, antialiasing, and self-shadowing.

The contributions of our work lie in a novel multiresolution hair representation, interactive tools for editing hair, and efficient hair rendering methods, all aimed at decreasing a user's time and tedium for modeling complex hairstyles. The remainder of this paper is organized as follows. Section 2 reviews related work and section 3 provides a brief system overview. The multiresolution hair representation is detailed in section 4, and section 5 presents editing tools suited to the representation. Interactive rendering algorithms are presented in section 6, and results are presented and discussed in section 7. Sections 8 and 9 discuss implementation issues and future directions.

2 Related Work

Since the pioneering work by Csuri et al. [1979], researchers have developed a number of hair modeling systems. These systems are categorized based on hairstyles to which they are best suited. Refer to [Parke and Waters 1996; Thalmann and Hadap 2000] for more comprehensive survey on human hair modeling and rendering research.

Smooth Hairstyle: Strand hair models are often limited to smooth hairstyles, due to the lack of mechanisms to simulate discontinuous clustering effects in their simplified physics simulation or interpolation methods. Anjyo et al. use cantilever beam dynamics and one-dimensional angular dynamics for hair modeling and animation [1992], later extended by Lee and Ko [2001] for interactive hairstyling. Rosenblum et al. produce hair motion by mass-spring simulation of individual hair strands [1991]. Exploiting the similarity between fluid and smooth hairstyles, Hadap and Thalmann develop an interactive hairstyling system [2000] and a dynamics system simulating hair/hair interaction [2001].

Watanabe and Suenaga [1992] propose a wisp model, later extended in an interactive hairstyling system by Chen et al. [1999]. In an integrated hair system by Daldegan et al. [1993], characteristic hair strands define the boundary of wisps. The common assumption that hair strands are parallel inside a wisp makes it hard to model rigorous strand variations such as curliness with random parameters to perturb wisps. Also, discontinuous clusters (e.g., due to combing) are not efficiently handled due to the underlying interpolation between key hair strands.

Clusters and Discontinuity: The cluster hair model by Yang et al. [2000] defines the boundary of a wisp with generalized cylinders, later used in an interactive hairstyling system in [Xu and Yang 2001]. Strand variations are implicitly modeled with ray tracing volume density. A similar method is used in the production of the movie 'Shrek' in which wisps are defined by control polygons [Falk and Sand 2001]. These methods handle bounded, man-made hairstyles such as braids, but stylistic strand variations are difficult to model. A method to model discontinuities in smooth hairstyles is reported in [Kim and Neumann 2000]. The dynamics system by Plante et al. [2001] simulates the interactions between discontinuous wisps in motion. However, the clustering is pre-determined before simulation and the distribution of strands inside a wisp is fixed.

Animal Fur or Short Hair: Explicit geometry is often avoided for short animal fur. Kajiya and Kay [1989] use volumetric texture (or texels) to mimic a group of hair strands, later extended by Neyret [1997] for multiresolution. An appearance-based model of distant animal fur is developed [Goldman 1997]. For real-time rendering of animal fur, concentric textures are used to sample the volumetric texture functions, augmented with large-scale geometry [Lengyel 2000; Lengyel et al. 2001]. A common assumption with animal fur is that the distribution of hair has repeated patterns. However, the assumption does not hold with longer human hair, as adjacent hair strands at the scalp split, curl, and move away from each other. Such styling effects require a significantly different modeling method from existing animal fur techniques [Lengyel et al. 2001].

3 Overview

Figure 3 illustrates the structure of a multiresolution hair model. A parametric patch on the scalp surface (section 4.2), defines the region where hair can originate. A hair model is constructed hierarchically, starting from a small set of generalized cylinders (GCs). A GC defines the boundary of each hair cluster, controlling a group of clusters or hair strands (see section 4.1 for our definition of a GC). The user interactively subdivides each hair cluster, adding more detail until the desired appearance is achieved. Subdivision steps (section 4.4) add new nodes to the hierarchy we call a hair tree. Editing operations such as curling, moving, copying, and selecting are applied to nodes of the hair tree (section 5). While the user edits a hairstyle, the system interactively visualizes the edited model using various rendering options (section 6).

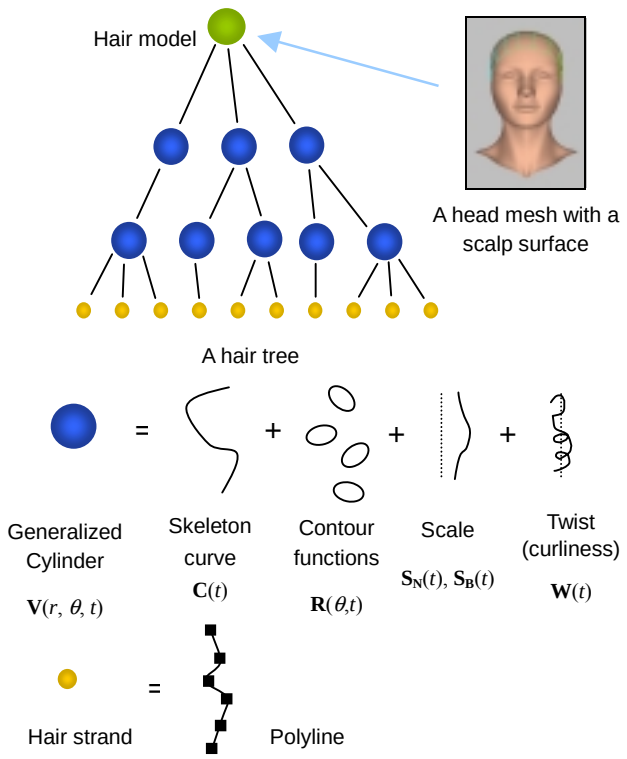


Figure 3. Graphical view of a multiresolution hair model.

4 Hair Cluster Representation

A GC defines the boundary of a hair cluster. In the following sections, we do not differentiate GCs from hair clusters and both terms are used interchangeably.

4.1 Generalized Cylinder

A GC is defined with a skeleton curve $\mathbf{C}(t)$ and a set of contours placed along the curve[1].

$$\mathbf{GC}(\theta, t) = \mathbf{C}(t) + \mathbf{R}(\theta, t) \quad (1)$$

where $\mathbf{C}(t)$ is a space curve parameterized by t and $\mathbf{R}(\theta, t)$ is a contour function centered at $\mathbf{C}(t)$. θ is an angle around the principal axis of a reference frame. With an additional parameter r, any point around the GC can be defined as

$$\mathbf{V}(r, \theta, t) = \mathbf{C}(t) + r\,\mathbf{R}(\theta, t) \quad (2)$$

Constructing a GC requires each contour to be properly aligned with reference frames. We use the frame generation method by Bloomenthal [1990]. The frame is represented with the tangent vector of the skeleton curve $\vec{\mathbf{T}}$, the principal normal $\vec{\mathbf{N}}$ and the binormal vector $\vec{\mathbf{B}}$ (Figure 4). Both $\vec{\mathbf{N}}$ and $\vec{\mathbf{B}}$ lie on the plane of the contour perpendicular to the skeleton curve. The pre-computed frames are interpolated to form a smoothly shaped GC. Let $\vec{\mathbf{T}}(t)$, $\vec{\mathbf{N}}(t)$, and $\vec{\mathbf{B}}(t)$ denote the interpolated frame vectors at t ($0 \le t \le 1$). The world coordinate of a point parameterized by (r,θ,t) is

$$\mathbf{V}(r, \theta, t) = \mathbf{C}(t) + r\,\mathbf{R}(\theta, t)\left\{\cos(\theta)\vec{\mathbf{N}}(t) + \sin(\theta)\vec{\mathbf{B}}(t)\right\} \quad (3)$$

[1] Similar definitions can be found in [Aguado et al. 1999; Xu and Yang 2000]. Alternatively, a GC can be viewed as rotational sweep of a profile curve [Kim et al. 1994].

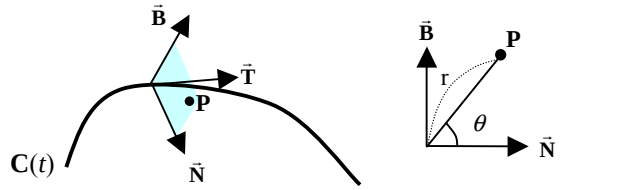


Figure 4. Definition of a reference frame along the space curve $\mathbf{C}(t)$. The parametric coordinate for a point $\mathbf{P}$ is given as (r,θ,t), where θ is defined as the angle around the tangent vector $\vec{\mathbf{T}}$, starting from the principal normal $\vec{\mathbf{N}}$.

Representation of Contour: $\mathbf{R}(\theta)$ is a contour shape function dependent on θ. $\hat{\mathbf{R}}(\theta)$ is defined as an offset distance function from a circle with a global radius s ($\mathbf{R}(\theta) = s + \hat{\mathbf{R}}(\theta)$). N pairs of angles and offset radius values control the shape of a contour along with the global radius s. The current implementation uses $\theta_i = 2i\pi / N$ and $\hat{\mathbf{R}}(\theta)$ is computed by linear interpolation of the angle / offset pairs. By definition, the contour is star-shaped, which we find flexible enough to represent the boundary of hair clusters. The offset function can efficiently represent star-shaped contours with a small set of parameters (currently, N = 32 is used). The continous contour function $\mathbf{R}(\theta, t)$ is linearly interpolated from a discrete number of contours.[2]

Auxiliary Functions: Adding auxiliary scale and twist terms completes our description of GC. The scale terms $\mathbf{S_N}(t)$ and $\mathbf{S_B}(t)$ let the user easily control the stretching and shrinking of GCs along the two axis vectors $\vec{\mathbf{N}}$ and $\vec{\mathbf{B}}$, while direct modification of the contour functions allows fine detailed control over the shape. The twist term $\mathbf{W}(t)$ controls the curliness of a hair cluster. Adding these terms yields

$$\mathbf{V} = \mathbf{C}(t) + r\,\mathbf{R}(\hat{\theta}, t)\left\{\cos(\hat{\theta})\mathbf{S}_N(t)\vec{\mathbf{N}}(t) + \sin(\hat{\theta})\mathbf{S}_B(t)\vec{\mathbf{B}}(t)\right\} \quad (4)$$

where $\hat{\theta} = \theta + \mathbf{W}(t)$.

4.2 Scalp Surface

A *scalp surface* is a parametric patch defining the region of a head where the user can place hair. We use a tensor-product Catmull-Rom spline patch that the user wraps over the head model. The current system allows a user to interactively specify each control point of the spline patch. The scalp surface is constructed once for each head model and it is useful in many ways. 1) The user can easily place and move the GC contour on the scalp mesh (Figure 5). 2) Sampling and assigning hair strands become simpler (section 4.3). 3) The subdivision of hair clusters is reduced to a 2D problem (section 4.4). 4) The scalp parameterization provides the user with convenient means to select clusters in 2D space (section 5.3).

Scalp space (middle figure in Figure 5) is spanned by u and v, the parametric coordinates of the surface. *World space* is specified by the 3D world coordinate frame. We denote the contour function in scalp space as $\mathbf{R}^s(\theta)$ and contours in world space as $\mathbf{R}^w(\theta)$. Scalp space contours are used for hair strand generation and for the initialization of multiple world space contours that define the shape of a GC. Appendix A describes how to generate an initial GC using the scalp space contour.

[2] There exist more rigorous methods to define and interpolate the contours of GCs [Aguado et al. 1999]. Our definition of contours aims at simplicity and computational efficiency.

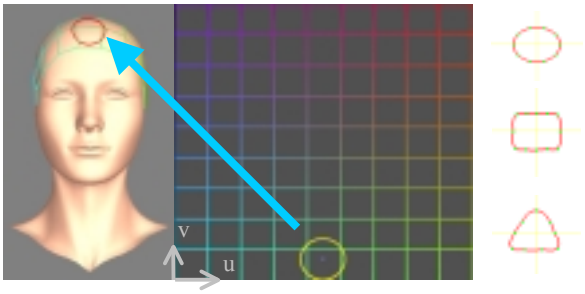


Figure 5. Placing a contour on the scalp. The user selects a contour (right) and interactively places it in 2D scalp space (yellow circle in the middle), which defines its 3D position (red circle on the left). The grid is color coded to help the user match corresponding positions.

4.3 Generation of Hair Strands

Hair strands are represented as polylines (a set of connected line segments). Creating hair strands independently within each GC (Figure 6a) can cause visually distracting hair density variation since we allow arbitrarily shaped scalp space contours that can overlap each other (Figure 7). Rather than trying to prevent the overlaps, we sample the root positions of hair strands first, and assign the strands to their *owner* GCs (Figure 6b).

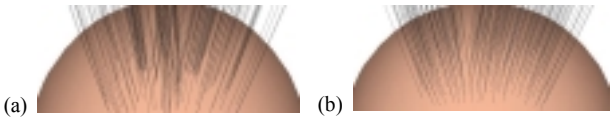


Figure 6. (a) Independent hair generation for each cluster causes uneven hair density in overlapping regions. (b) By assigning pre-sampled hair strands to clusters, the overlap problem is solved.

Initially, (or whenever the total number of hair strands changes), the root position (u_h,v_h) of each hair strand is uniformly sampled on the scalp grid. For each hair strand, we determine if there exists a GC that owns the strand (the details of the ownership decision are deferred until section 4.4). If there is such a GC, the parametric coordinate (r_h, θ_h) of the strand is computed (see appendix B for details). Given (r_h, θ_h), the hair strand is created by connecting points computed with equation 4, incrementing t from 0 to 1.

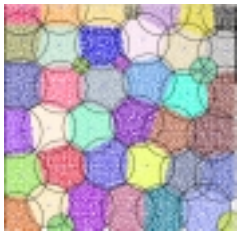


Figure 7. Scalp space contours can overlap each other. Colored dots illustrate the root positions of strands. Each color indicates the owner cluster of hair strands.

Note that equation 4 assumes that each GC contour lies in the plane perpendicular to the skeleton curve. Since the scalp shape is curved, planar contours can misalign the root positions of strands. To handle the problem, the root position ($t = 0$) of each hair strand is directly evaluated on the scalp, using

$$\mathbf{V}(r_h,\theta_h,0) = \mathbf{S}(u_h,v_h) \quad (5)$$

where $\mathbf{S}(u,v)$ is the parametric scalp patch. However, this modification can still cause unnatural bending along hair strands if other points of polyline segments are directly evaluated from equation 4 (Figure 8a). Similar problems can arise from self-intersections in GCs of high curvature. We interpolate a Catmull-Rom spline curve for each hair strand to guarantee smoothness (Figure 8b). The number of spline control points is

Figure 8. (a) Straight line connections can cause unnatural bending around the root position and in highly curved region. (b) Using spline interpolation at strand-level, smoothness is guaranteed.

the same as that of the skeleton curve of the strand's owner GC. The polyline segments of each hair strand are evaluated from the spline interpolation.

Varying the length of each hair strand increases the model's natural appearance. A random sample η is taken from a uniform distribution from 1-k to 1 ($0 \le 1\text{-}k \le \eta \le 1$), where k denotes the degree of tip length variation. Adding the length variation, we evaluate the spline controls points of each hair strand using ηt instead of t for equation 4.

4.4 Subdivision

When a parent cluster is subdivided, contours and skeleton curves of its child clusters are created. The contours of child clusters are inherited from those of the parent cluster unless the user specifies alternative shapes. Inherited contours are scaled by $\rho / \sqrt{N}$, where N is the user-specified number of child clusters and ρ controls the degree of overlap between child clusters. For world space contours, ρ is set to 1.0. For scalp space contours ρ is set to a value larger than 1.0 (typically $1.3<\rho<1.5$), to ensure that the subdivided region is fully covered by the contours of child clusters.

To place the child clusters on the scalp, we first sample random positions inside the contour of their parent. This often produces uneven distributions of contour positions (Figure 9a). We use the position relaxation algorithm by Turk [1991] to improve the distribution (Figure 9b). Given these positions, skeleton curves are created using the method described in section 4.3.

Figure 9. Subdivision of contours and tiling of the scalp space. (a) Random positioning (ρ=1.4). (b) Position relaxation. (c) An example tiling of the scalp surface after a few subdivision steps.

Deciding the Owners of Hair Strands After child clusters are positioned on the scalp, hair strands are re-assigned to the new clusters. As shown in Figure 9c, contours of the child clusters can overlap with each other or contours of existing clusters. In overlapping regions, we assign hair strands to a cluster with the closest center position in scalp space, similar to computing the centroidal voronoi diagram [Hausner 2001] of the center positions of all the clusters on the scalp (Figure 10a). However, simply computing the voronoi diagram may be problematic when precise controls for hair grouping are desired (e.g., for parting).

Our hair ownership decision algorithm shown below as a pseudo code allows the user to design and tune the tiling and subdivision patterns if necessary (Figure 10b). For each hair strand, the algorithm traverses the hair tree starting from root

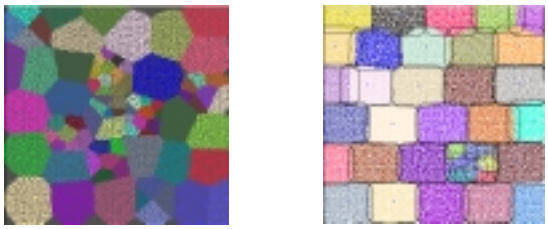


Figure 10. (a) Hair strand assignments based on contours shown in Figure 9c. The root positions of hair strands are drawn as colored dots. (b) User controlled tiling and subdivision. Note the clear borders between contours.

clusters. The closest cluster containing the strand is chosen at each depth. The iteration is repeated until it reaches a leaf cluster. If there is only one such leaf cluster, the strand is assigned to the cluster regardless of the distance to other clusters.[3] When two or more leaf clusters contain the same strand due to overlaps, the cluster with a center position closest to the strand is selected. When a hair strand is contained in a non-leaf cluster, but not contained in any of its descendents[4], the strand is assigned to the closest descendent leaf cluster. Note that we allow only leaf clusters to own hair strands, and for each hair strand, there is at most one owner. Thus, we can maintain consistent hair density and limit the total number of hair strands; both features prove useful while level-of-detail manipulations control the appearance of the overall hair model (section 6).

```
function FINDOWNEROFAHAIRSTRAND (HairStrand H)
O ← NULL ,  C ← first root cluster,  done ← FALSE
while (!done and C != NULL) do
    done ← TRUE
    forall {S | S is a sibling of C or C itself } do
        CHECKFORINCLUSION(H,S)
        if S contains H and its center is closer to H than O
            O ← S
            done ← FALSE
    if (!done)
        C ← O.FIRSTCHILD
if (O is not a leaf cluster)
    forall { L | L is a descendent of O and a leaf node } do
        if L's center is closer to H than O
            O ← L
return O
```

5 INTERACTIVE MANIPULATION

The core of MHM lies in the user's ability to edit any node in the hair tree. The user controls the position, contours, scale, and twist of any GC, affecting the hair model at multiple levels of detail (section 5.1). A sub-tree in the hair tree can be used as a user-defined *style template*. The copy/paste tool (section 5.2) transfers a user-designed style from one cluster to other clusters. Section 5.3 describes a set of selection methods that exploit the scalp surface and the hair tree. During interactive manipulation, hair/head penetration is detected and avoided (section 5.4).

5.1 Editing Properties of Hair Clusters

The user interactively manipulates GC parameters such as scale, twist, and skeleton shape (equation 4). When editing leaf clusters, the system simply reassigns the parameters and update hair strands. When editing a non-leaf cluster, all of its descendent clusters must follow the shape changes to preserve their relative position and orientation (Figure 11).

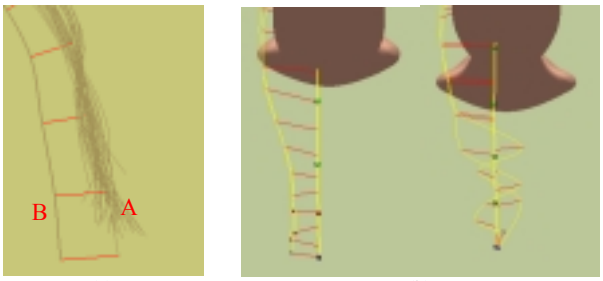
(a) (b) (c) (d)

Figure 11. a) The user picks the control point (inside the blue circle) of a high-level cluster. The user can b) move c) scale d) twist the region around the control point.

When a non-leaf cluster is edited, all the descendant clusters are *bound* to the cluster. Let **V** be the cluster before editing. Assume that the user changes **V** to **V'**. Then each descendant cluster is updated as follows. 1) For each control point **P** of the skeleton curve, we find the parametric coordinate (r,θ,t) of **P** with regard to the cluster **V** such that $\mathbf{P} = \mathbf{V}(r,\theta,t)$ (*bind*). 2) The new position **P'** is recalculated using $\mathbf{P'} = \mathbf{V'}(r,\theta,t)$ (*update*).

The bind procedure is the inverse transform of equation 4, *i.e.* $(r,\theta,t) = \mathbf{V}^{-1}(\mathbf{P})$. Details are given in appendix C. The bind process is performed whenever the user initiates an action (for example, selects a hair cluster and starts editing it), whereas the update process occurs at every step of the user interaction. Hair strands are considered *statically* bound to a cluster when their owner is decided. The bind/update method is similar to *Wires* [Singh and Fiume 1998], but our method compensates for the axial scaling and twist terms, and no falloff term is used.

Temporary/Local Binding To manipulate a group of clusters with no common ancestor (e.g., attaching a root cluster to another root cluster), the user selects a group of clusters and binds the clusters to an arbitrary *binder* cluster. Then, these selected clusters are temporarily bound to the binder cluster instead of their parent clusters until the user nullifies the setting. This option is especially useful to control root clusters (e.g., to make a pony tail style), or to locally control a set of clusters that stem from two disjoint root positions (Figure 12).



(a) (b)

Figure 12. In this figure, only skeleton curves are shown. (a) Temporary binding. The cluster A is temporarily attached to cluster B. (b) Two root clusters are locally bound to another (binder) cluster and selected control points (red) are twisted around the binder cluster.

[3] In contrast, a voronoi diagram based approach may assign the strand to another cluster even if the hair strand does not originate from the region bounded by the cluster.

[4] This case occurs when the contours of child clusters do not perfectly cover the contour of the parent cluster even after the position relaxation step. Note that we set ρ to a value larger than 1.0 to minimize this artifact.

5.2 Copy and Paste

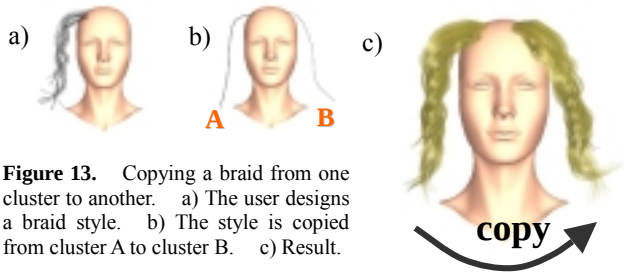


Figure 13. Copying a braid from one cluster to another. a) The user designs a braid style. b) The style is copied from cluster A to cluster B. c) Result.

We can treat a hair cluster and its descendent clusters as a user-defined style template. The copy process illustrated in Figure 13 transfers a *style* from one cluster to another, by copying the sub-tree parameters. Copying is similar to the editing process in section 5.1. When the style of cluster A is copied to cluster B, the descendent clusters of cluster A are first bound to A, but these clusters are updated with cluster B as their parent. These updated clusters replace existing descendents of cluster B. Contours are scaled by the ratio between the size of contours of cluster A and B. Other style parameters such as scale and twist are simply duplicated. Note that any node in the hair tree can be copied to another node regardless of depth; making every edit the user makes a potential style template (Figure 14).

Figure 14. A braid style resulting from multiple copy/paste operations at multiple scales.

5.3 Selection Methods

When a hair model is subdivided into many small clusters, it becomes difficult to select some portions of the model due to its volumetric nature (Figure 15). The following selection methods are available. 1) The user picks a control point of a cluster or specifies a region with a sphere. 2) The user selects clusters in scalp space with standard 2D selection methods (for example, dragging a rectangle). 3) The user picks a cluster and traverses the hair tree. Operations such as 'pick first child node' or 'pick next sibling' are mapped to the keyboard, which proves useful in manipulating the multiresolution model. 4) Selection based on depth is provided (e.g., 'display only the clusters of depth < 2').

Figure 15. The complexity of a multiresolution hair model. Contours (green) and skeleton curves (yellow) of 1340 leaf clusters are shown.

5.4 Avoiding Hair-Head Penetration

It is visually distracting if hair strands penetrate the head. During editing, hair strands and skeleton curves are tested for intersection. We use the closest-point-transform (CPT) algorithm [Mausch 2000]. The CPT provides a fast look-up table solution with the preprocessing of the distances and gradients to the closest points on the head mesh. Let D(**p**) be the distance from a point **p** to the closest point on the mesh and $\nabla(\mathbf{p})$ be the gradient of the distance. Penetration is avoided by altering each point **p** inside the head mesh (D(**p**) < 0) with equation 6.

$$\mathbf{p}_{new} = \mathbf{p} - \mathbf{D}(\mathbf{p}) \cdot \nabla(\mathbf{p}) \qquad (6)$$

6 INTERACTIVE RENDERING

Hair models in our framework are explicitly rendered; every hair strand is drawn as polylines in OpenGL. This section describes methods tailored to render such explicit models for interactive modeling purpose.

Shading Model: The lighting calculation provided in OpenGL is disabled and shading is calculated in software, using the anisotropic shading model by Kajiya and Kay [1989]. The shaded color is computed at each point of the line segments and colors are interpolated with OpenGL. Other shading models such as that in [Goldman 1997] could be equally applicable.

Self-shadowing: Self-shadowing is an essential cue to depict volumetric hair (Figure 17a and Figure 17b). We use our opacity shadow maps algorithm [Kim and Neumann 2001], a fast approximation of deep shadow maps [Lokovic and Veach 2000]. Since shadows are view-independent, they can be computed once and cached for reuse while the user interactively changes views.

Antialiasing: Since hair strands are very thin, it is important to draw them smoothly with correct filtering. The antialiased line drawing option in OpenGL alone is not sufficient since the correct result depends on the drawing order [McReynolds 1997]. Our visibility ordering algorithm, inspired by [Levoy and Whitted 1985], determines the drawing orders for polyline segments of hair strands based on the distance from the camera (Figure 16).

First, the bounding box of all the segments is sliced with planes perpendicular to the camera. Each *bin*, a volume bounded by a pair of adjacent planes, drawn as a color bar in Figure 16, stores indices of segments whose farthest end point is contained by the bin. After other objects (e.g., a head mesh) are drawn, the depth buffer update is disabled. Then, the segments are drawn as antialiased lines such that the ones indexed by the farthest bin are drawn first. Although simple, the method is fast and converges to exact ordering as hair strands are drawn with more segments. Since we keep relatively dense line segments for each hair strand, the algorithm produces visually satisfactory results.

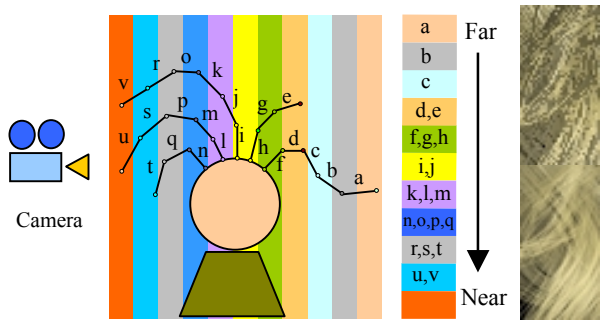


Figure 16. For smooth drawing, polyline segments are sorted by the distance from the camera and drawn as antialiased OpenGL lines with the user controlled alpha values. The small images on the right show close-ups for an aliased image (top) and an antialiased image (bottom).

Figure 17. Effects of shadows and level of detail. Images were captured during interactive user sessions. (a) A: front lighting, B: back lighting. (b) A: without shadows, B: with shadows. (c) A: 1,200,000 segments (20000 strands, 60 segments per strand, and $\alpha = 0.3$), B: 100,000 segments (5000 strands, 20 segments per strand, and $\alpha = 1.0$). The model A is 12 times more complex than model B, hence the rendering time is 12 times slower for A than B.

In the interactive modeling framework, the viewpoint does not change much from frame to frame. This coherence enables us to perform the visibility ordering periodically. In contrast, depth buffer based super-sampling methods (e.g., accumulation buffer [McReynolds 1997]) must compute visibility at every frame. In addition, the alpha values of segments can control the perceived thickness of hair strands. As strands become thinner, super-sampling methods would require more samples while alpha value changes suffice in the visibility-ordered hair model.

Level of Detail and Interactive Rendering: With explicit hair models, the results of rendering (e.g., shadows, colors, etc.) can be cached, allowing users to interactively view the model during editing. Users can control three parameters - alpha values, the number of hair strands, and the number of segments per strand -, to adjust the speed of rendering (Figure 17c). These parameters give users choices in the tradeoff between speed of rendering and image quality. Increasing the number of hair strands and the number of segments per strand contributes to improved rendering quality, whereas decreasing these parameters allows the user to interactively examine the hair model at higher frame rates.

7 RESULTS

Our system assists users in creating a variety of complex models (Figure 18 ~ 22). Depending on the complexity, it takes from 10~20 minutes to a few hours to model each hairstyle. The most time-consuming step in modeling is the positioning of initial root clusters. Typically users design 10 to 30 root clusters, which consumes about 70 ~ 80 percent of the total modeling time.

Hairstylists often use curling and combing as their means to promote clustering effects, adding visual richness due to shadows. Likewise, we observe that adding more hair strands does not necessarily enhance the visual complexity of a hair model. As the hair volume is filled with more strands, room for shadows diminishes. Thus, we find that, to model a natural hairstyle, the *structural* aspects are as important as individual strand details. Note that the spaces between clusters after subdivision amplify shadows between clusters, enhancing the perceived complexity of the model. Shadowing is less salient for smooth hairstyles. However, subdividing smooth hair models can also add visually interesting combing effects (Figure 19).

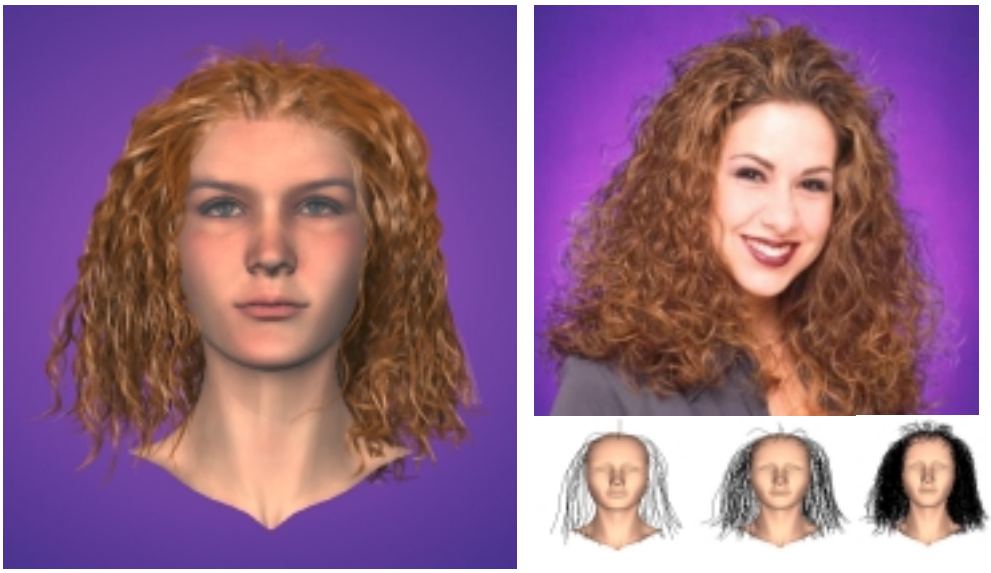


Figure 18. Designing a complex hairstyle. The hair model (left) consisting of 940 clusters at three levels of detail was created after the photograph shown in the middle (image courtesy of http://www.hairboutique.com). The small images show the skeleton curves at each level of detail.

Figure 19. The hairstyle (top) is modeled after the image (bottom, http://www.hairboutique.com)

8 IMPLEMENTATION AND DISCUSSION

Our implementation runs on a system with an Intel PIII 700 Mhz CPU, 512 MB memory, and nVidia Quadro 2 Pro graphics card. Table 1 shows time measurements for a hair model of 10,000 hair strands and 40 segments per each strand. For shadow calculation, 40 opacity maps of 200 x 200 pixels were used.

Shading	1.2 second	**Visibility Ordering**	0.43 second
Hair update	5000 strands per second	**Shadow Calculation**	5.88 seconds*

Table 1. Time measurements for components of MHM system. *Measurement per light source

During the interactive rendering sessions, a typical setting is to use about 150,000 segments (e.g., 5000 hair strands with 30 segments per strand) with alpha value of 0.5 in a window size of 512 by 512. The system interactively (> 5 frames per second) renders the edited model with antialiasing. The frame rate will only get better with progress in CPU and graphics hardware performance. Shadow computation remains a bottleneck in rendering and we hope to further accelerate the shadow calculation with 3D texture hardware [Kim and Neumann 2001].

As reported in [Singh and Fiume 1998], the closest point on a curve becomes ambiguous as a point moves away from the curve or if the curve is of a high curvature. This can cause the control points of skeleton curves to *drift* during the bind/update procedure. However, the bind/update procedure is only used for rough editing of the hair model, while the user eventually *corrects* the drift by refining child clusters. This problem could be obviated by fitting the entire skeleton curves of the bound clusters, not just the control points, to the skeleton curve of the binder cluster.

In the bind/update procedure, *every* descendent cluster is bound to the edited cluster, not just immediate child clusters. It is tempting to store the positions of child clusters as offsets to their parent cluster to speed up the bind operation. However, there are more time-consuming operations such as hair strands updates, temporary binding, and penetration avoidance. The offset approach could be inefficient for these operations that require the world coordinates of each cluster. For example, if all the root clusters are subdivided three times, the offset approach will incur three times more GC computations than the current approach.

To speed up the model updates, we use caching schemes such as tabulation of sine functions and curve approximation to find the closest point on a curve. Thus, memory requirement is currently high (200 MB at maximum). Considering that the data structure of the hair tree itself is compact (4KB per each GC), the memory usage could be reduced by further optimizations and faster CPUs. The current bottleneck in the model update is the curve evaluation for each strand (table 1). For efficiency, we tag hair clusters that the user is currently editing and update hair strands only for these clusters. The spline curve drawing in OpenGL may speed up the process if combined with programmable vertex shaders for local shading calculation. However, that would require an alternative antialiasing algorithm since our visibility algorithm requires every strand to be represented as polyline segments.

When the root positions of hair strands are sampled, the hair density can vary over the curved scalp surface. Although this is not a serious problem, we could provide an additional density map on the scalp that the user can paint. The scalp surface may also be used as an atlas for texture maps of other parameters such as colors, thickness, etc.

As with any multiresolution approach, the benefit of MHM is maximized with complex models, such as natural curly hairs. The major benefit of MHM lies in a user's capability to rapidly add structural details and variations, given initial templates. Providing a gallery of rough style templates would greatly speed up the production of various hairstyles.

We use GCs as our cluster representation mainly due to its modeling efficiency (e.g., axial scaling/twist) and also because a hair strand can be treated as a very thin GC. However, GCs may not be best suited to modeling global shapes for smooth hair or short hair. Existing methods may suffice for smoothly varying hairstyles (e.g., [Anjyo et al. 1992; Hadap and Thalmann 2000]) and short hair/animal fur (e.g, [Goldman 1997; Lengyel 2000; Lengyel et al. 2001]). Thus, it may be worth investigating methods to fit root clusters into other control primitives (e.g., polygonal models) or using MHM as a *backend* system to refine the results from other modeling methods.

Figure 20. Curly ponytail.

Figure 21. Short spikes.

9 CONCLUSION

Human hair modeling is often an arduous task. The interactive multiresolution hair modeling (MHM) system presented in this paper strives to ease the process. The system evolved from a quest to find a fundamental hair representation flexible enough to encompass the wide range of human hairstyles. MHM extends the scope of hair models ranging from smooth shapes and short hair, to complex cases such as curly hair, braids and spiky clusters. However, avenues remain for future extensions.

Currently, the hair model is defined for a specific head model. It would be useful to transfer hairstyles from one head model to another. The scalp surface abstraction and the copy/paste tool may provide good starting points. Scattered data interpolation techniques such as Radial Basis Functions may also provide reasonable adaptation of the hair model to different head meshes. Completely automating the process may require sophisticated dynamics to handle both hair/hair and hair/head interactions.

In our framework, the user implicitly *designs* the hair/hair interactions in the form of multiresolution clusters. Extending MHM to support animation seems feasible in many ways. At the highest level, we could provide the user with kinematics or dynamics controls over root clusters. Alternatively, strand level animation techniques could be employed to animate the lowest level strands. However, in reality when hair moves, hair/hair interactions cause hair clusters to change dynamically, from large coherently moving clusters to independently moving strands, and vice versa. Simulation of such dynamic clustering effects is a challenging problem. The preservation of user-defined style raises another issue. Currently, we do not know of any animation technique that can handle this complexity.

The fluid flow model by Hadap and Thalmann [2001] and the layered wisp model by Plante et al. [2001] approach the hair/hair interaction problem at discrete levels, the former at strand level and the latter at cluster level. Simulating dynamic clustering effects may involve combining those methods in a continuous manner. Ultimately, hair clusters should split and merge dynamically, automatically, and often partially. We anticipate that the dynamic clustering problem could be formulated as that of dynamically updating the hair tree, consistently changing the hair strand ownership. The rich volumetric models generated by MHM hint that realistic hair motion may be efficiently simulated in a multiresolution framework, in the sprit of the 'simulation level of detail systems' [O'Brien et al. 2001].

ACKNOWLEDGEMENTS

This work received funding from the Annenberg Center and the National Science Foundation through its ERC funding of the Integrated Media Systems Center at USC. We recognize the holistic support from our colleagues in the USC CGIT laboratory. Special thanks go to Jun-Yong Noh, Douglas Fidaleo, and Clint Chua for proofreading the initial manuscript and video editing. We deeply thank Hiroki Itokazu for the head model and Bret StClair for his wonderful textures. We also thank J. P. Lewis for his numerous contributions and Dr. LaFrance for kindly showing us her unpublished manuscript. Many thanks go to Sean Mausch for his CPT software and Laehyun Kim for directing us to this algorithm. We also appreciate the anonymous reviewers and referee for their many valuable comments and suggestions.

Figure 22. More hairstyles.

References

AGUADO, A. S., MONTIEL, E., ZALUSKA, E. 1999. Modeling Generalized Cylinders via Fourier Morphing. *ACM Transactions on Graphics*. 18(4), 293-315.

ANJYO, K., USAMI, Y., AND KURIHARA, T. 1992. A Simple Method for Extracting the Natural Beauty of Hair. In *Computer Graphics (Proceedings of ACM SIGGRAPH 92)*, 26(4), ACM, 111-120.

BLOOMENTHAL, J. 1990. Calculation of Reference Frames Along A Space Curve. In A. Glassner, editor, *Graphics Gems*, Academic Press, 567-571.

CHEN, L., SAEYOR, S., DOHI, H., AND ISHIZUKA, M. 1999. A System of 3D Hairstyle Synthesis Based on the Wisp Model. *The Visual Computer, 15(4)*, 159-170.

CSURI, C., HAKATHORN, R., PARENT, R., CARLSON, W., AND HOWARD, M. 1979. Towards an Interactive High Visual Complexity Animation System. In *Computer Graphics (Proceedings of ACM SIGGRAPH 79)*, 13(4), ACM, 288-299.

DALDEGAN, A., THALMANN, N. M., KURIHARA, T., AND THALMANN, D. 1993. An Integrated System for Modeling, Animating and Rendering Hair. In *Computer Graphics Forum (Proceedings of Eurographics 93)*, 211-221.

FALK, R. AND SAND, L. R. (ORGANIZERS) 2001. "Shrek": The Story Behind The Screen. *ACM SIGGRAPH Course Note 19.*

GOLDMAN, D. 1997. Fake Fur Rendering. In *Proceedings of ACM SIGGRAPH 97*, ACM Press / ACM SIGGRAPH, New York, 127 – 134.

GRABLI, S., SILLION, F. X., MARSCHNER, S. R., AND LENGYEL, J. E. 2002. Image-Based Hair Capture by Inverse Lighting. In *Proceedings of*

Graphics Interface 2002, to appear.

HADAP, S. AND THALMANN, N. M. 2000. Interactive Hair Styler Based on Fluid Flow. *Eurographics Workshop on Computer Animation and Simulation*, 87-100.

HADAP, S. AND THALMANN, N. M. 2001. Modeling Dynamic Hair as a Continuum. In *Computer Graphics Forum (Proceedings of Eurographics 2001*), 329 – 338.

HAUSNER, A. 2001. Simulating Decorative Mosaics. In *Proceedings of ACM SIGGRAPH* 2001, ACM Press / ACM SIGGRAPH, New York, 573-580.

KAJIYA, J. AND KAY, T. 1989. Rendering Fur with Three Dimensional Textures. In *Computer Graphics (Proceedings of ACM SIGGRAPH 89)*, 23(4), ACM, 271-280.

KIM, M., PARK, E., AND LEE, H. 1994. Modeling and Animation of Generalized Cylinders with Variable Radius Offset Space Curves. *The Journal of Visualization and Computer Animation*, 5(4), 189-207.

KIM, T. AND NEUMANN, U. 2000. A Thin Shell Volume for Modeling Human Hair. In *Computer Animation 2000*, Philadelphia, IEEE Computer Society, 121-128.

KIM, T. AND NEUMANN, U. 2001. Opacity Shadow Maps. *Rendering Techniques 2001*, Springer, 177-182.

LAFRANCE, M. 2001. First Impressions and Hair Impressions. Unpublished manuscript. Yale University, Department of Psychology, New Haven, Connecticut. http://www.physique.com/sn/sn_yale-study2.asp

LEE, D.-W. AND KO, H.-S. 2001. Natural Hairstyle Modeling and Animation. *Graphical Models*, 63(2), 67-85.

LENGYEL, J. E. 2000. Realtime Fur. *Rendering Techniques 2000*, Springer, 243-256.

LENGYEL, J. E., PRAUN, E., FINKELSTEIN, A., AND HOPPE, H. 2001. Real-Time Fur over Arbitrary Surfaces. *ACM Symposium on Interactive 3D Techniques 2001*, 227-232.

LEVOY, M. AND WHITTED, T. 1985. The Use of Points as a Display Primitive. *Technical Report 85-022*, Computer Science Department, University of North Carolina at Chapel Hill, January.

LOKOVIC, T. AND VEACH, E. 2000. Deep Shadow Maps, In *Proceedings of SIGGRAPH 2000*, ACM, New York, 385-392.

MAUCH, S. 2000. A Fast Algorithm for Computing the Closest Point and Distance Transform. Submitted for publication in the *Jounral of SIAM SISC*. http://www.acm.caltech.edu/~seanm/software/cpt/cpt.pdf

MCREYNOLDS, T. (Organizer) 1997. Programming with OpenGL: Advanced Techniques, *ACM SIGGRAPH Course Note 11*.

NEYRET, F. 1997. Modeling, Animating, and Rendering Complex Scenes Using Volumetric Textures. *IEEE Transaction on Visualization and Computer Graphics*, 4(1), 55-70.

O'BRIEN, D., FISHER, S., AND LIN M. 2001. Automatic Simplification of Particle System. In *Computer Animation 2001*, Seoul, Korea, IEEE Computer Society, 210-219.

PARKE, F. AND WATERS, K. 1996. Computer Facial Animation. *A K Peters*.

PLANTE, E., CANI, M.-P., AND POULIN, P. 2001. A Layered Wisp Model for Simulating Interactions Inside Long Hair. *Eurographics Workshop on Computer Animation and Simulation 2001,*139–148.

ROSENBLUM, R., CARLSON, W., AND TRIPP E. 1991. Simulating the Structure and Dynamics of Human Hair: Modeling, Rendering and Animation. *The Journal of Visualization and Computer Animation*, 2(4), 141-148.

SINGH, K. AND FIUME, E. 1998. Wires: A Geometric Deformation Technique. In *Proceedings of ACM SIGGRAPH 98*, ACM, New York, 405 - 415.

STOLLNITZ, E. AND DEROSE, T. D. AND SALESIN, D. H. 1996. Wavelets for Computer Graphics. *Morgan Kaufmann*.

THALMANN, N. M. AND HADAP, S. 2000. State of the Art in Hair Simulation. *International Workshop on Human Modeling and Animation*, Korea Computer Graphics Society, 3-9.

TURK, G. 1991. Generating Textures on Arbitrary Surface Using Reaction Diffusion. In *Computer Graphics (Proceedings of ACM SIGGRAPH 91)*, 21(4), 289-298.

WATANABE, Y. AND SUENAGA, Y. 1992. A Trigonal Prism-Based Method for Hair Image Generation. *IEEE Computer Graphics and Application*, 12(1), 47-53.

XU, Z. AND YANG, X. D. 2001. V-HairStudio: An Interactive Tool for Hair Design. *IEEE Computer Graphics and Applications*, 21(3), 36-43.

YANG, X. D., XU, Z., YANG, J., AND WANG, T. 2000. The Cluster Hair Model. *Graphical Models*, 62(2), 85-103.

Appendix

A. Generation of Generalized Cylinder from Scalp Surface

When the user places a 2D contour on the scalp, the corresponding GC is created. First, the root position $\mathbf{P_0}$ of the skeleton curve is calculated as $\mathbf{P_0} = \mathbf{C}(0) = \mathbf{S}(u_c, v_c)$, where u_c and v_c denote the center of the scalp space contour. Let $\mathbf{N_0}$ be the surface normal at $\mathbf{P_0}$. Initially, the skeleton curve is formed as a straight line (Each control point $\mathbf{P_i}$ of the skeleton curve is given as $\mathbf{P_i} = \mathbf{P_0} + iL / (N - 1)\ \mathbf{N_0}$, where N is the number of control points) and L is the length of the skeleton curve. As a next step, we convert $\mathbf{R^s}(\theta)$, the scalp space contour to $\mathbf{R^w}(\theta)$ in world space. Let $\mathbf{R_i}$ be the 3D position of each sample point of the contour (recall from section 4.1 that each contour is represented with the sample values $\mathbf{R}(\theta_i)$).

$$\mathbf{R_i} = \mathbf{S}(u_c + \mathbf{R^s}(\theta_i)\cos(\theta_i), v_c + \mathbf{R^s}(\theta_i)\sin(\theta_i))$$

To make the contour planar, we project $\mathbf{R_i}$ to the plane formed by $\mathbf{P_0}$ and $\mathbf{N_0}$. Let $\hat{\mathbf{R}}_i$ be such a projected point. Then, $\mathbf{R^w}(\theta_i)$ is the distance from the center point $\mathbf{P_0}$ to $\hat{\mathbf{R}}_i$ in 3D space.

$$\mathbf{R^w}(\theta_i) = \left\| \hat{\mathbf{R}}_i - \mathbf{P_0} \right\|$$

The contour function is then decomposed into the global scale s and offset function as defined in section 4.1. The number of contours is the same as the number of control points of the skeleton curves. These contours are simply copied at each control point of the skeleton curves.

B. Inclusion Test for a Hair Strand

A hair strand is given its location on the scalp by (u_h, v_h). Let the center of the contour $\mathbf{R^s}(\theta)$ of each cluster be represented by (u_c, v_c). Let $\Delta u = u_h - u_c$ and $\Delta v = v_h - v_c$. Then, the angle θ_h is given as

$$\theta_h = \cos^{-1}\left(\frac{\Delta u}{\sqrt{\Delta u^2 + \Delta v^2}} \right)$$

$$\text{and } \theta_h = 2\pi - \theta_h \text{ if } \Delta u < 0 \qquad \text{(B.1)}$$

A hair strand is contained by a cluster if $\mathbf{R}^s(\theta_h) \le \sqrt{\Delta u^2 + \Delta v^2}$

A bounding box of each contour is used to speed up the test. Also, the inverse cosine function is tabulated. If a hair strand is assigned to a cluster, the parametric coordinate for the root position of the hair strand is given as (r_h, θ_h), where $r_h = \dfrac{\sqrt{\Delta u^2 + \Delta v^2}}{\mathbf{R}^s(\theta_h)}$.

C. Binding a Point P to a Cluster V

Given a point $\mathbf{P}$ and a cluster $\mathbf{V}$, the parametric coordinate (r, θ, t) of the point with respect to the cluster is found as follows. First, t is chosen as the value that minimizes the Euclidean distance between point $\mathbf{P}$ and the skeleton curve $\mathbf{C}(t)$. Let $\mathbf{P_C}$ be such a point that $\mathbf{P_C} = \mathbf{C}(t)$. Then θ is given as the angle between a vector connecting $\mathbf{P}$ and $\mathbf{P_C}$ and the principle normal $\mathbf{N}(t)$. The angle should be corrected for the scaling terms $S_N(t)$, $S_B(t)$. Let the projection of the vector $\overrightarrow{P_C P}$ on $\mathbf{N}(t)$, $\mathbf{B}(t)$ be P_N, P_B. Then, the angle θ is given using equation B.1, by letting $\Delta u = P_N / S_N(t)$, and $\Delta v = P_B / S_B(t)$. After correcting for the twist term $\mathbf{W}(t)$, $\theta = \theta - \mathbf{W}(t)$. The parameter r is the ratio between the Euclidean distance between $\mathbf{P}$, $\mathbf{P_C}$ and $\mathbf{R}(\theta, t)$.

$$r = \left\| P - P_C \right\| / \mathbf{R}(\theta, t).$$