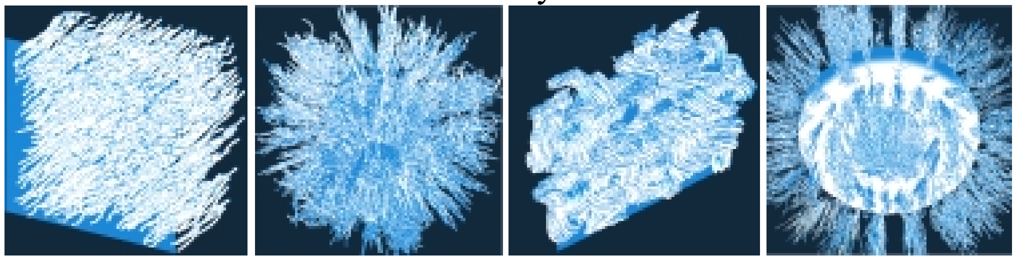
Modeling Hair and Fur with NURBS

Anna Sokol
ansokol@cs.sunysb.edu
Computer Science Department
SUNY Stony Brook



Abstract: *This paper is presenting a framework for modeling hair and fur using NURBS surfaces. This framework will simulate different types of hair from very curvy to straight, from very short to very long, and from very dense to bolding. Each hair strand is modeled using a twisted NURBS cylindrical surface. The hair is modeled on top a plain, a torus, and a sphere.*

1 Introduction

Realistic hair simulation has been a huge problem due to its incredible complexity. For example, a human head can have over 100,000 individual strands. The simulation of long and/or curly hair creates even more complexity.

While movies like *Monsters, Inc.* and *Final Fantasy* have been able to generate very realistic looking hair and fur, they weren't done anywhere near real time. Modeling, styling, simulating, and animating hair remains a slow, tedious, and often a painful process for animators.[1]

In this paper, I propose to model hair relatively quickly using twisted NURBS cylindrical surfaces. The hair is rendered in almost real time. It can render 5000 individual hair strands in 5 seconds. The hair can look short or long, curly or straight, and messy or combed.

2 Related Work

Hair modeling is a very active area of research and numerous approaches exist. For example, modeling hair using level-of-detail representation[1] where hair is rendered as strands, clusters, or individual strands based on how visible the individual hair strands happen to be. They used butterfly subdivision to represent each cluster or strand and the strips are not necessarily rendered at all.

Another approach is to model hair as 2D strips. Each hair strip, modeled by one patch of parametric surfaces in particular NURBS, represents a group of hair strands. A variety of shapes may be defined for each strip. For the rendering, they apply alpha-mapping on the tessellated polygons to achieve a realistic visual effect.[2]

Techniques for real-time rendering of fur and hair graphics were presented in [Lengyel 2000; Lengyel et al. 2001; NVIDIA 2001]. However, these techniques do not work well or are not applicable for rendering long, wavy or curly hair.[1]

An integrated system for modeling, animating and rendering hair is described in [Dald93]. It uses an interactive module called HairStyler [Thal93] to model the hair segments that represents the hairstyle.[2]

To reduce the overall computation time, strands of hair that are near each other or move in a similar fashion, are bundled together as a group or as a wisp [Kurihara et al. 1993].[1]

Other modeling methods include trigonal prisms with 2D hair distribution maps, volumetric models, single triangle laid out on the surface, a pyramid of triangles for one strand, connected segments of triangular prisms, and volume densities controlled with pseudo-random functions.[11]

3 Technique

Twisted NURBS cylindrical surfaces where used to model individual hair strands. Randomization was used to model clusters of hair strands. The hair was modeled on a plain, a sphere, and a torus. The rest angle range, the number of partitions per hair strand, the length of each partition, the number of hair strands per cluster, and the number of clusters were dynamically user selected.

3.1 NURBS surface

A NURBS surface of degree p in the u direction and degree q in the v direction has the form:

$$S(u,v) = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j}P_{i,j}}{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j}}$$

The $\{P_{i,j}\}$ from a bi-directional control net, the $\{w_{i,j}\}$ are the weights, and the $\{N_{i,p(u)}\}$ and $\{N_{j,q(u)}\}$ are the non-rational B-spline basis functions defined on the knot vectors:

$$U = \{\underbrace{0,\ldots,0}_{p+1}, u_{p+1},\ldots,u_{r-p-1},\underbrace{1,\ldots,1}_{p+1}\}$$

$$V = \{\underbrace{0,\ldots,0}_{q+1}, v_{q+1},\ldots,v_{s-q-1},\underbrace{1,\ldots,1}_{q+1}\}$$

where $r = n+p+1$ and $s = m+q+1$.

3.2 Individual strands

Each individual strand of hair will be modeled as a twisted NURBS cylindrical surface with n control points, degree 2 and 20×20 sudivision. Each of these cylinders contains a certain thickness and a length. A circle determines the thickness of the cylindrical strand. First, the control points of that circle, which will contain the thickness of the individual strand, are created. Then the v knots for these circle control points are created.

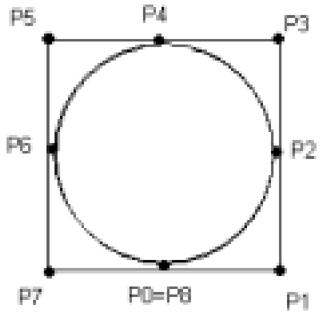


Figure 1: The thickness of each strand with 9 control points

The twisting of the hair strand is determined by randomizing the rest angle range down the length of the cylinder. The u knots are based on how many parts that hair contains.

After all the control points are determined for each hair strand, that hair strand is rendered as a twisted NURBS cylindrical surface. One end of the hair strand is *rooted* on the underlining surface. Otherwise the hair would have been flying of every which way.

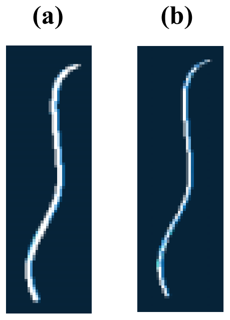


Figure 2: Individual strand seen as

(a) wireframe and (b) smooth.

3.3 Clusters

For each cluster, individual hair strands are created on the fly. If the hair is considered messy, the randomized twisting is created individually for every hair strand. If the hair is considered combed, the randomized twisting is created only once for the entire cluster.

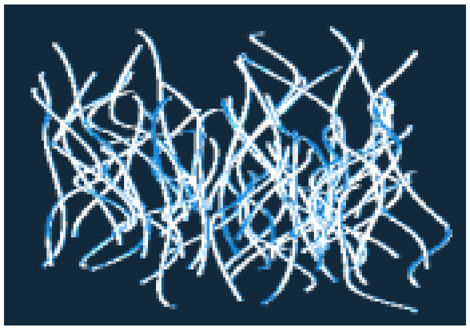


Figure 3: Messy hair in one cluster

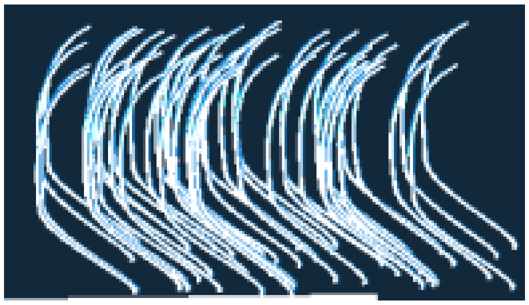


Figure 4: Combed hair in one cluster

4 Algorithm

The underlining shapes used here include a plain, a torus, and a sphere. The root of each hair strand is rooted on the surface. The torus and the sphere are rendered using NURBS surfaces with degree 2 and 20×20 subdivision. The plain on the other hand is rendered only through polygonal representation.

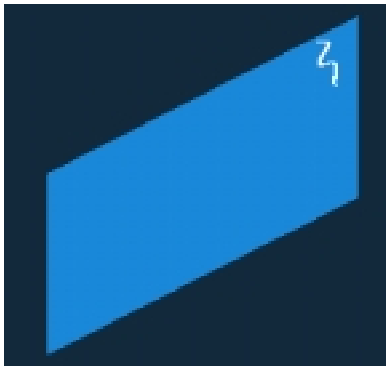


Figure 5: The plain

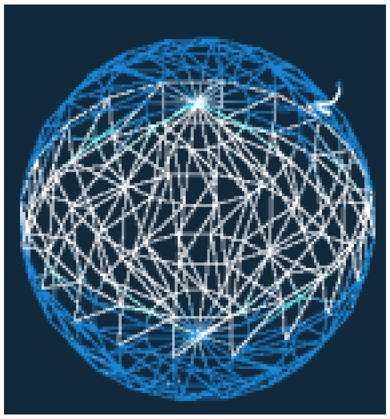


Figure 6: The sphere

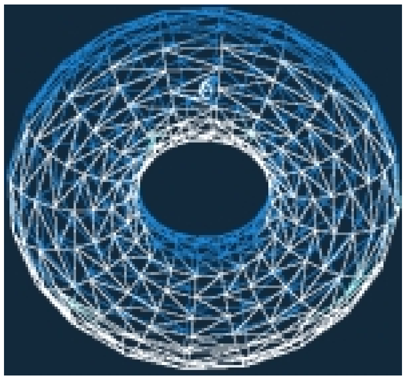


Figure 7: The torus

In the case of the plain, the hair root has y=0 and y is incremented randomly by the length range for each part of the hair. Therefore, the hair will always grow upward. The root x and z locations are determined randomly within the range of the cluster. The twist and curl are determined by the randomization of the rest angle range for the x and z values.

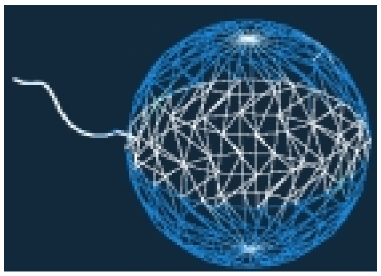


Figure 8: An individual strand attached on surface vertex.

Since the torus and the sphere are created using NURBS surfaces determining the root and the direction of the hair strands is trickier. The vertices of the subdivided NURBS surface are used as roots. For each hair strand this vertex is randomly selected from all the vertices in the subdivided surface. The direction of the hair strand is determined by the normal of the vertex that happens to be that hair strand's root. Figure 8 shows the algorithm for determining the direction and twist of a hair strand. The parts represent the selected amount of parts per hair strand. The restAngle is the rest angle range that has been selected. The length is the length range that has been selected. The surf[j].nx represent the x coordinate of the underlining surface j.

```
For i=1 to parts {
if (abs(surf[j].nx)<0.0001)
    xc[i]=random(restAngle)+xc[0]
else
   xc[i]=surf[j].nx*random(length)+xc[i-1];
if (abs(surf[j].ny)<0.0001)
    yc[i]=random(restAngle)+yc[0];
else
    yc[i] =surf[j].ny*random(length)+yc[i-1];
if (abs(surf[j].nz)<0.0001)
    zc[i] = random(restAngle)+zc[0];
else
    zc[i] = surf[j].nz*random(length)+zc[i-1];
}
```

Figure 9: The twist and direction of the hair on a NURBS surface.

```
for (i=0 … parts)
{
  for (j=0 … 9)
  {
      cp[i*9+j].x =xc[i] + origcp[j].x;
      cp[i*9+j].y =yc[i] + origcp[j].y;
      cp[i*9+j].z =zc[i] + origcp[j].z;
      cp[i*9+j].w = origcp[j].w;
  }
}
```

Figure 10: Building the hair strand control points.

6 Results

This implementation uses OpenGL and C++ on a Windows 2000 PC. The interface used is FLTK. It was compiled under Microsoft Visual Studio C++. The following results took about 5 seconds to render each because they use 50 hair strands per 10^2 clusters with 10 parts per hair strand, in essence 5000 individual hair strands. The less hair strands, clusters, and parts the faster the rendering time.

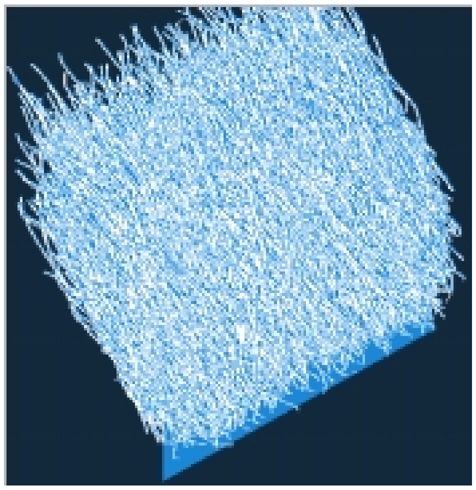


Figure 11: Messy hair on a plain with 50 strands per cluster, 10^2 clusters, length range = 40, 10 hair parts, and rest angle range = 40

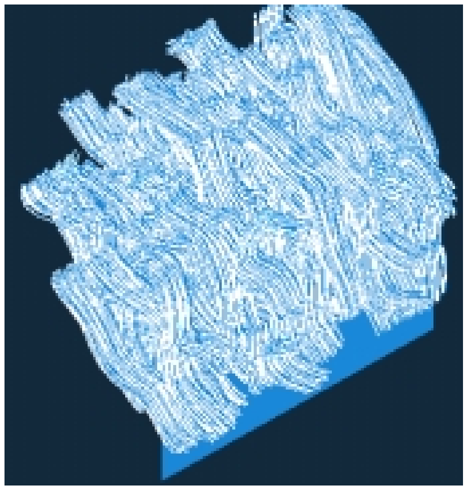


Figure 12: Combed hair on a plain with 50 strands per cluster, 10^2 clusters, length range = 40, 10 hair parts, and rest angle range = 40

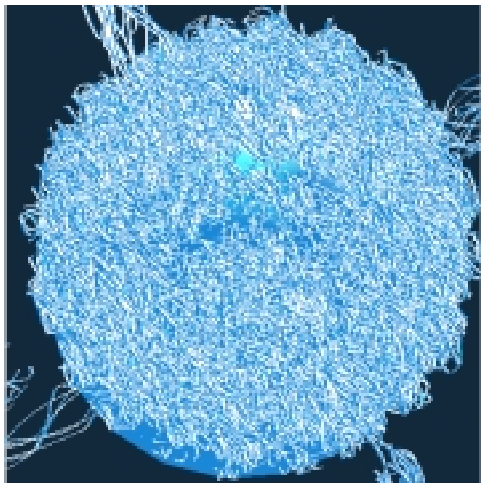


Figure 13: Messy hair on a sphere with 50 strands per cluster, 10^2 clusters, length range = 40, 10 hair parts, and rest angle range = 40

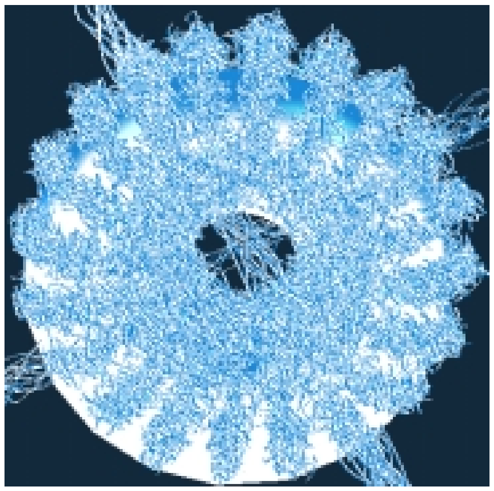


Figure 15: Messy hair on a torus with 50 strands per cluster, 10^2 clusters, length range = 40, 10 hair parts, and rest angle range = 40

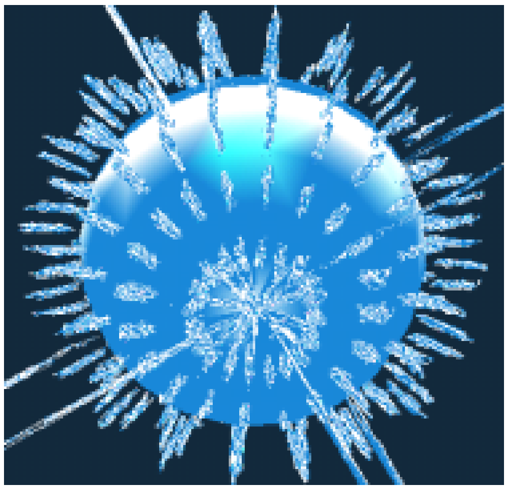


Figure 14: Combed hair on a sphere with 50 strands per cluster, 10^2 clusters, length range = 40, 10 hair parts, and rest angle range = 40

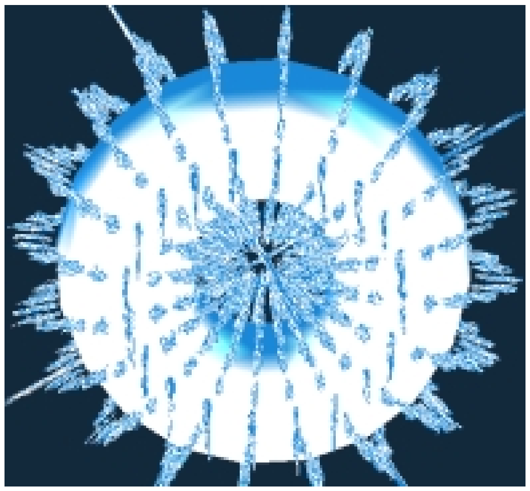


Figure 16: Combed hair on a torus with 50 strands per cluster, 10^2 clusters, length range = 40, 10 hair parts, and rest angle range = 40

7 Conclusion

In this paper is presenting a framework for modeling hair and fur using NURBS surfaces. This framework simulates different types of hair from very curvy to straight, from very short to very long, and from very dense to bolding. Each hair strand is modeled using a twisted NURBS cylindrical surface. The hair is modeled on top a plain, a torus, and a sphere. Texture mapping renders everything black.

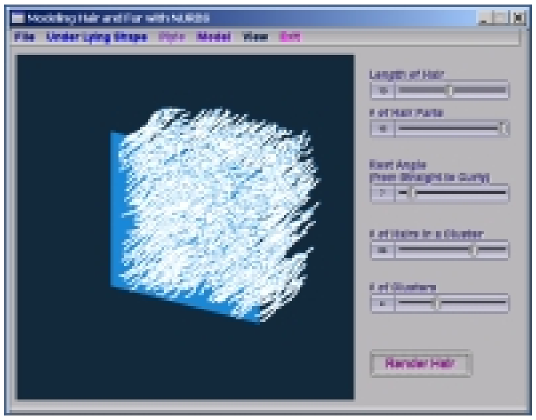


Figure 17: The user interface

8 References

1. Kelly Ward, Ming C. Lin, Joohi Lee, Susan Fisher, and Dean Macri. Modeling Hair Using Level-of-Detail Representations. *Computer Animation and Social Agents*, 2003.
2. Koh, C. K., and Huang, Z. 2001. A Simple Physics Model to Animate Human Hair Modeled in 2D Strips in Real Time. *Proceedings of Eurographics Workshop 2002,* 127-138.
3. Yang Guang and Huang Zhiyong, A Method of Human Short Hair Modeling and Real Time Animation, *IEEE*, 2002
4. Tsuneya Kurihara, Ken-ichi Anjyo, and Daniel Thalmann, Hair Animation with Collision Detection, *Models and Techniques in Computer Animation, Springer-Verlag, Tokyo*, pp.128-138, 1993
5. Leslie Piegl and Wayne Tiller, Curve and Surface Constructions using rational B-splines, *Computer-Aided Design*, 19(9), 485-498, 1987
6. K. Anjyo, Y. Usami, and T. Kurihura. A Simple Method For Extracting The Natural Beauty Of Hair, *SIGGRAPH* (92), pp. 111-120 (1992).
7. W. Böhm. Insert New Knots into B-spline Curves, *Journal of Computer Aided Design,* 12 (4), pp. 199-201 (1980).
8. L. H. Chen, S. Saeyor, H. Dohi, and M. Ishizuka. A System of 3D Hair Style Synthesis Based on the Wisp Model, *The Visual Computer*, 15 (4), pp. 159-170 (1999).
9. E. Cohen, T. Lyche, and R. Risenfeld. Discrete B-Splines and Subdivision Technique in Computer-Aided Geometric Design and Computer Graphics, *CGIP*, 14 (2), pp. 87-111 (1980).
10. A. Daldegan, T. Kurihara, N. Magnenat Thalmann, and D. Thalmann. *An Integrated System for Modeling, Animating and Rendering Hair*, Proc. Eurographics (93), Computer Graphics Forum, Vol.12, No3, pp.211-221 (1993).
11. C. K. Koh and Z. Huang, Real-time Animation of Human Hair Modeled in Strips, *Computer Animation and Simulation*, Springer-Verlag, pp.101-110 (2000).
12. R. E. Rosenblum, W E. Carlson, and I. E. Tripp. Simulating the Structure and Dynamics of Human Hair: *Modeling, Rendering and Animation*. The Journal of Visualization and Computer Animation, 2 (4), pp. 141-148 (1991).
13. C. C. Tanner, C. J. Migdal, and M. T. Jones. The Clipmap: A Virtual Mipmap, *SIGGRAPH (98),* pp. 151-158 (1998).
14. N. Magnenat Thalmann and A. Daldegan. Creating Virtual Fur and Hair Styles for Synthetic Actors. *In Communicating with Virtual Worlds*, Springer-Verlag, Tokyo, pp. 358-370 (1993).
15. J. Lengyel, Real-time fur. *Proc. Of Eurographics Workshop on Rendering,* 2000
16. J. Lengyel, E. Praun, A. Finkelstein, and H. Hoppe, Real-time fur over arbitrary surfaces, *Proc. of ACM Symp. on Interactive 3D Graphics,* 2001
17. NVIDIA. Final fantasy technology demo 2001. *http://www.nvidia.com* 2001
18. NVIDIA. Technical brief. *http://developer.nvidia.com/docs/lo/1451/SUPP/accuview.final.pdf*