Course: 34

Photorealistic Hair Modeling, Animation, and Rendering

Tuesday, Half Day, 8:30 am - 12:15 pm

Level: Intermediate

The full range of hair simulation problems and practical solutions, both novel research ideas and time-tested industrial practices. This course examines hairstyling, hair-hair interactions, and hair rendering using graphics hardware.

Prerequisites

Familiarity with the fundamentals of computer graphics, numerical linear algebra, differential equations, numerical methods, rigid-body dynamics collision detection and response, physics-based illumination models, and fluid-dynamics is strongly recommended, but not necessary.

Topics

Overview of the hair-research field, multi-resolution editing and fluid streamlines for hairstyling, hair stiffness dynamics as rigid multibody serial chains, collision detection via OBB trees and adaptively sampled distance fields, techniques for hair-hair interaction using fluid dynamics or sparse-guide hairs, advanced hair illumination models, volumetric shadows, and hair rendering using (programmable) graphics hardware.

Organizer

Nadia Magnenat-Thalmann

MIRALab, University of Geneva

thalmann@miralab.unige.ch

Lecturers

Armin Bruderlin

Sony Pictures Imageworks

armin@imageworks.com

Sunil Hadap

PDI/Dreamworks

shadap@pdi.com

Tae-Yong Kim

Rhythm & Hues Studio

tae@rhythm.com

Ulrich Neumann

University of Southern California

uneumann@graphics.usc.edu

Yizhou Yu

University of Illinois at Urbana-Champaign

yyz@cs.uiuc.edu

Steve Worley

Founder of Worley Laboratories

Steve@worley.com**DESCRIPTION OF THE COURSE**

Topic 1: Introduction 30 min

8:30 State of the Art - Nadia Magnenat-Thalmann

8:45 Quest for realism - Ulrich Neumann

Topic 2: Hair Shape Modeling and Dynamics

9:00 A Multiresolution Technique for Hair Styling -Tae-Yong Kim and Fluid flow based Hair Modeling - Sunil Hadap

9:35 Hair Dynamics -Sunil Hadap and Yizhou Yu

10:15 Break

Topic 3: Hair Rendering

10:30 Hair microstructure and Illumination- Steve Worley

11:00 Algorithms for hardware accelerated Hair Rendering-Tae-Yong Kim

Topic 4: Case studies

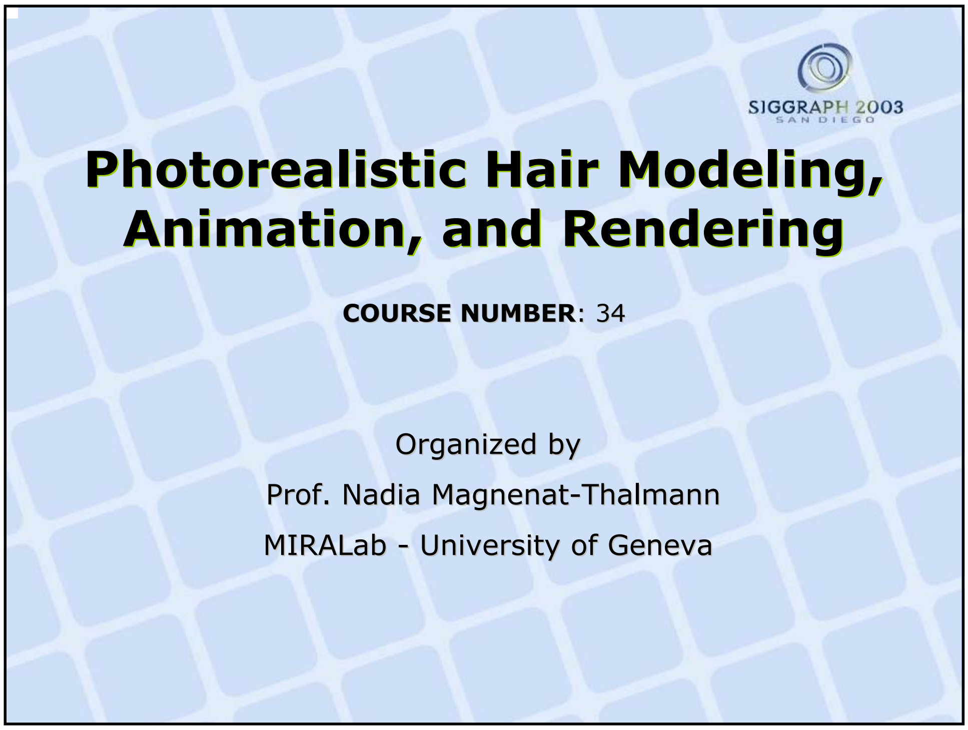
11:20 Cultural Heritage applications at MIRALab-Nadia Magnenat-Thalmann

11:35 Productions Hair/Fur Pipeline at Imageworks- Armin Bruderlin

Topic 5: Questions and Discussions

12:00 Audience and Speakers

SIGGRAPH 2003
SAN DIEGO

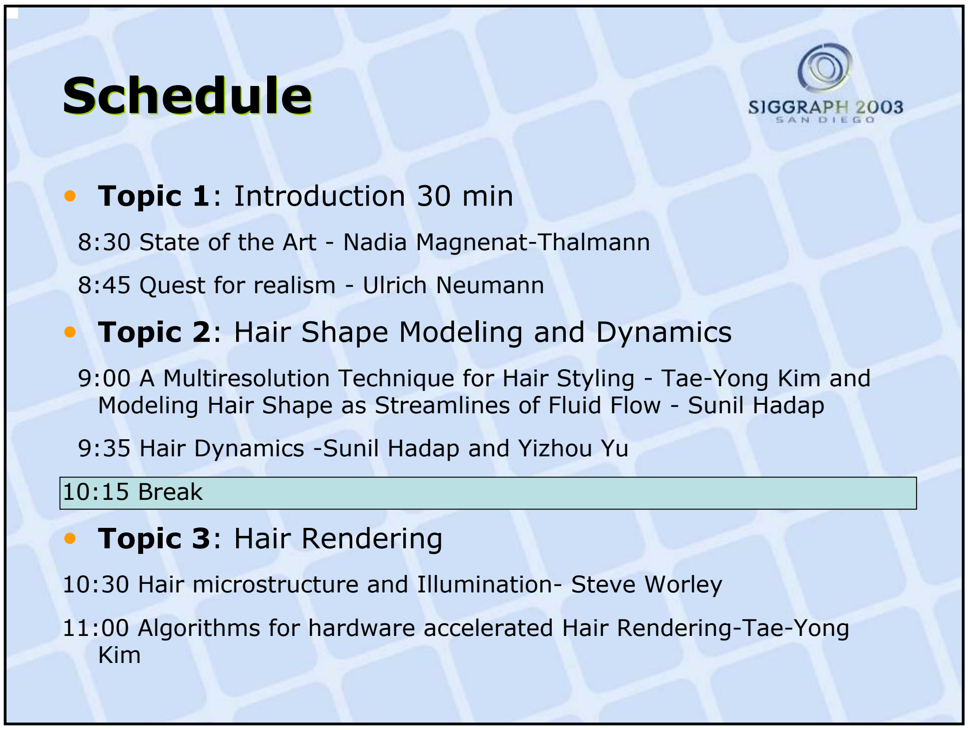


SIGGRAPH 2003
SAN DIEGO

Photorealistic Hair Modeling, Animation, and Rendering

COURSE NUMBER: 34

Organized by
Prof. Nadia Magnenat-Thalmann
MIRALab - University of Geneva



Schedule

SIGGRAPH 2003 SAN DIEGO

- **Topic 1**: Introduction 30 min

8:30 State of the Art - Nadia Magnenat-Thalmann

8:45 Quest for realism - Ulrich Neumann

- **Topic 2**: Hair Shape Modeling and Dynamics

9:00 A Multiresolution Technique for Hair Styling - Tae-Yong Kim and Modeling Hair Shape as Streamlines of Fluid Flow - Sunil Hadap

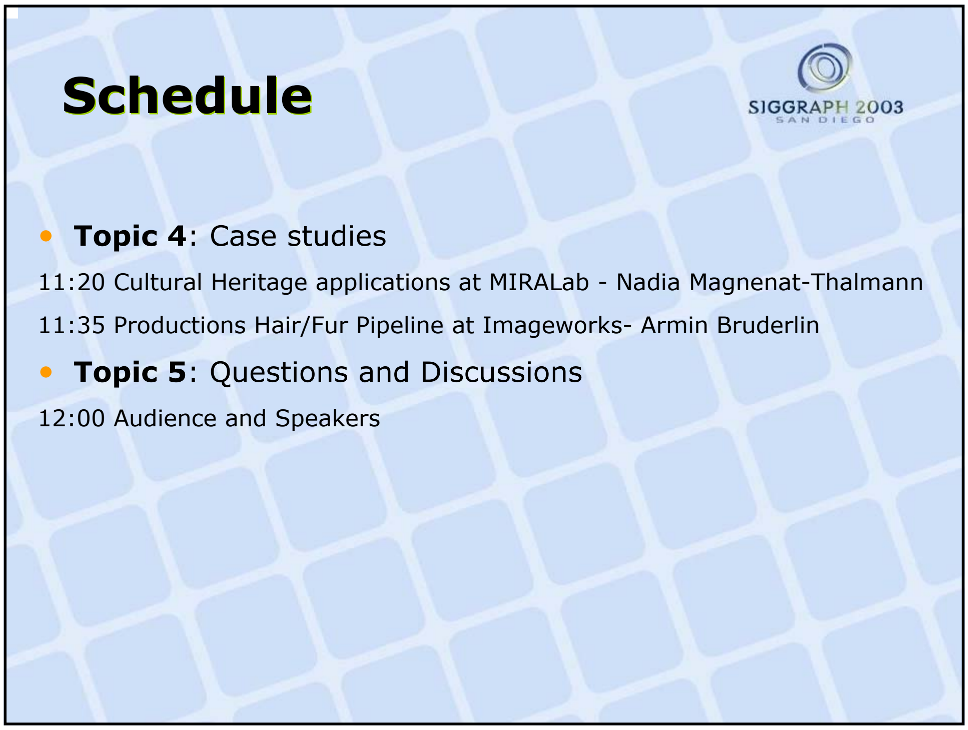
9:35 Hair Dynamics -Sunil Hadap and Yizhou Yu

10:15 Break

- **Topic 3**: Hair Rendering

10:30 Hair microstructure and Illumination- Steve Worley

11:00 Algorithms for hardware accelerated Hair Rendering-Tae-Yong Kim



Schedule

SIGGRAPH 2003 SAN DIEGO

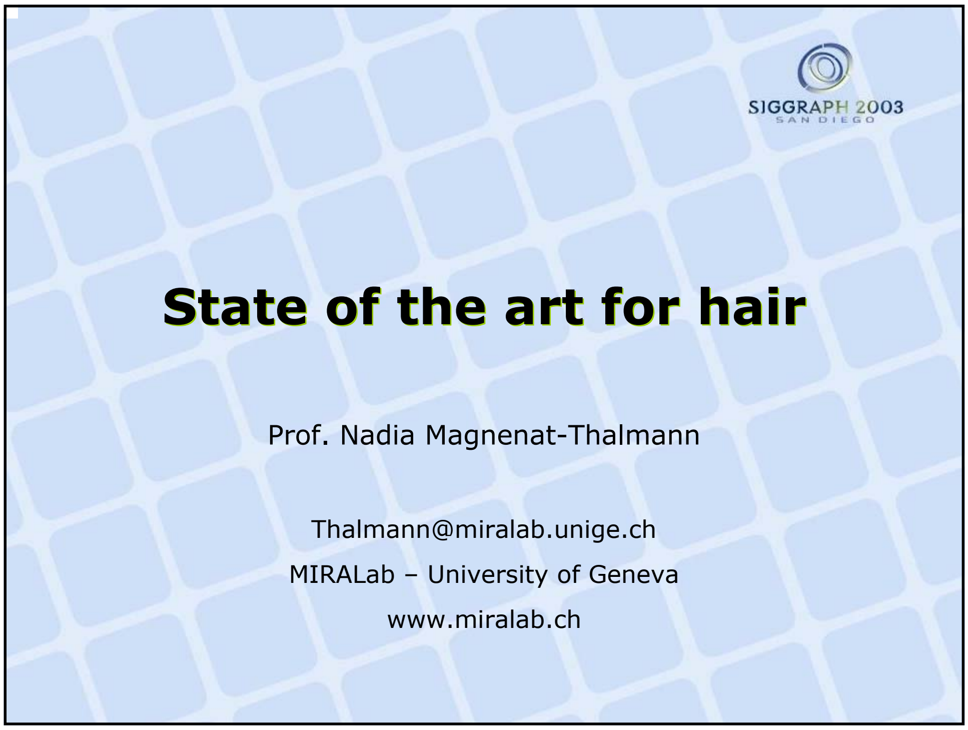
- **Topic 4**: Case studies

11:20 Cultural Heritage applications at MIRALab - Nadia Magnenat-Thalmann

11:35 Productions Hair/Fur Pipeline at Imageworks- Armin Bruderlin

- **Topic 5**: Questions and Discussions

12:00 Audience and Speakers



State of the art for hair

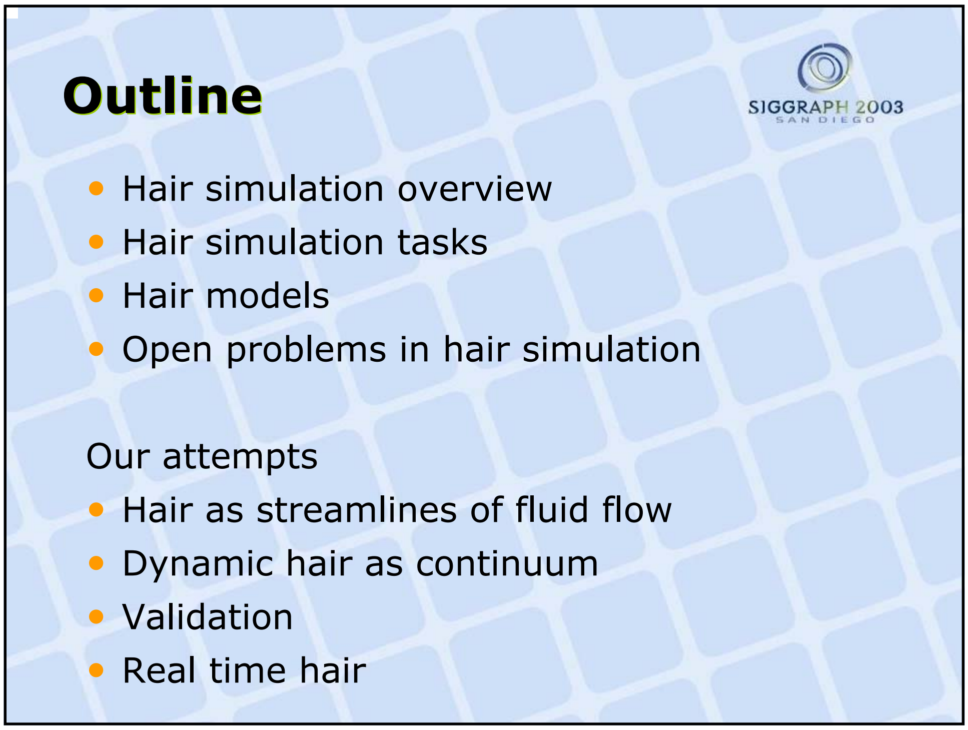
Prof. Nadia Magnenat-Thalmann

Thalmann@miralab.unige.ch

MIRALab – University of Geneva

www.miralab.ch

SIGGRAPH 2003 SAN DIEGO



Outline

- Hair simulation overview
- Hair simulation tasks
- Hair models
- Open problems in hair simulation

Our attempts

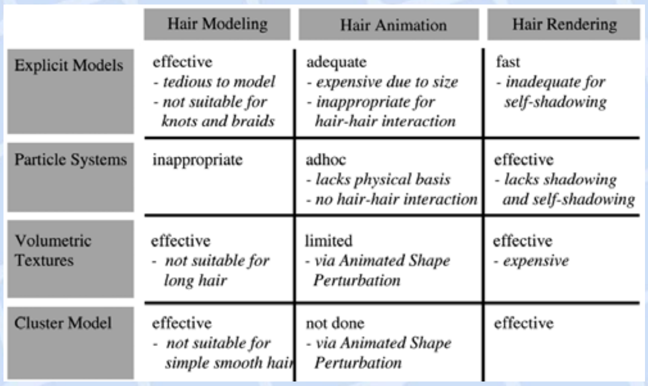
- Hair as streamlines of fluid flow
- Dynamic hair as continuum
- Validation
- Real time hair

SIGGRAPH 2003 SAN DIEGO

Hair Simulation Overview

Tasks

Models



	Hair Modeling	Hair Animation	Hair Rendering
Explicit Models	effective *- tedious to model* *- not suitable for knots and braids*	adequate *- expensive due to size* *- inappropriate for hair-hair interaction*	fast *- inadequate for self-shadowing*
Particle Systems	inappropriate	adhoc *- lacks physical basis* *- no hair-hair interaction*	effective *- lacks shadowing and self-shadowing*
Volumetric Textures	effective *- not suitable for long hair*	limited *- via Animated Shape Perturbation*	effective *- expensive*
Cluster Model	effective *- not suitable for simple smooth hair*	not done *- via Animated Shape Perturbation*	effective

Though seem to be independent tasks, they are highly interrelated.

Hair Simulation Tasks - Styling

- Hair shape is a result of complex physical interaction between hair-hair and hair-body
- Hairstyling – a constant human passion curlers, clips, knots, braids and up-dos
- Hair dynamics at interactive speed is impossible. Heuristic approach is needed for hair shape modeling.
- Thus, hair shape modeling is an exclusive task in computer graphics

Hair Simulation Tasks - Dynamics

Unilever hair shots

- Highly anisotropic physical behavior
  Solid-liquid duality
- Light weight of hair as compared to its acceleration, stiffness, friction and air drag
- Constant collisions / frictional interactions with the body
- 100,000 to 150,000 hair strands on scalp
- Hair-hair interaction, one of the unsolved problems of Computer Graphics

Hair Simulation Tasks - Rendering

- Intricate geometry of individual hair
- Large number of hair strands
- Complex interaction with light and shadows
  multiple scattering, luster, self-shadowing
- Anisotropy in shading
- Small thickness of the hair – anti-aliasing
  How artists paint hair?

Hair Models – Particle Systems

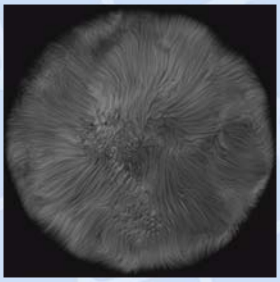
SIGGRAPH 2003 SAN DIEGO

"The End" by Alias|Wavefront 96

- Simple (tricks)
- Imagine the "flock of birds" Hair root leads and rest of the hair follows
- Most of the commercial animation systems support particle system and some form of hair rendering as a particle system
- Inappropriate for intricate hairstyling
- *adhoc* for Hair Dynamics
- Effective for Hair Rendering

Hair Models – Volumetric Textures

SIGGRAPH 2003 SAN DIEGO


Perlin *et al*, 1989

- Spatial density functions such as 3d noise and turbulence
- Stochastic models, nice way of interpreting complexity in nature such as grass, trees, forests and fur

Hair Models – Volumetric Textures

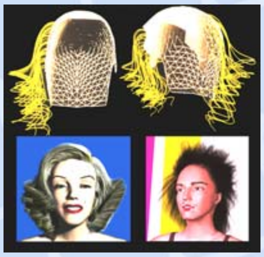
SIGGRAPH 2003 SAN DIEGO

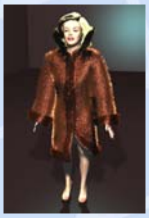
Kajiya *et al,* 1989

- Extension to non-analytic "Texels", tiling of fur texture on geometry
- Unlike hypertextures, can be used on complicated geometry.

Hair Models – Explicit Hair Models (styling)

SIGGRAPH 2003 SAN DIEGO


Daldegan *et al,* 1992



- Few characteristic hair strands are considered for shape
- Interactive definition in 3D, good overall control
- Further, population and shape control through *density, spread, jitter* and *orientation*
- Intuitive and versatile
- Effective but time consuming
- Can not be effectively used for complex hairstyles involving clips, knots and braid

Hair Models – Explicit Hair Models (dynamics)

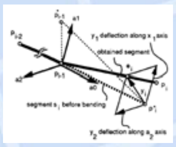
SIGGRAPH 2003 SAN DIEGO

Kurihara *et al,* 1992

- Each and every hair strand is considered for dynamics
- Close to physical reality
- Hair-hair interaction is not modeled although majority of hair volume comes from hair-hair interactions
- Hair volume is wrongly attributed to large hair stiffness

Hair as Cantilever Beam (styling)

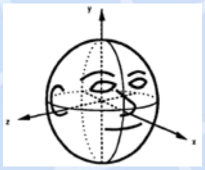
SIGGRAPH 2003 SAN DIEGO

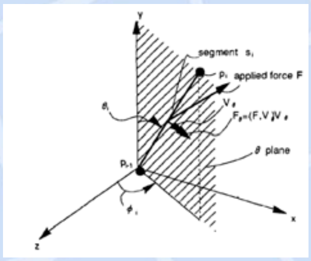

p_{i-1}^*
a1
p_{i-2}
γ_1 deflection along x_1 axis
obtained segment
s_i^*
p_{i-1}
a2
a0
p_i
γ_i
segment s_i before bending
p'_i
γ_2 deflection along a_2 axis

- The 2d cantilever equation is extended to 3d by appropriately selecting a spherical coordinate system
- The cantilevered hair strands form the hairstyle
- One can vary the bending properties to give waviness

Hair as Cantilever Beam (dynamics)

SIGGRAPH 2003
SAN DIEGO



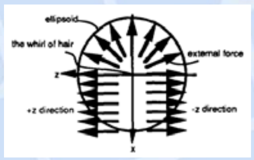


- One dimensional differential equation for angular momentum
  Euler equation

- However, hairstyle is in 3d
  Approximation by evaluating the one dimensional equation in the plane formed by segment and y axis.

Hair as Cantilever Beam (collision)

SIGGRAPH 2003
SAN DIEGO



- Approximation of polygonal head by an ellipsoid

- Pseudo force field to repel hair instead of real collision detection

Hair as Cantilever Beam (critique)

- One of the pioneering work. Use of rigid-body dynamics for hair simulation.
- Complete approach for hairstyling, animation and rendering
- Simplified single hair dynamic equations
- Hair-body interactions were grossly approximated by pseudo force field.
- No hair-hair interaction modeled

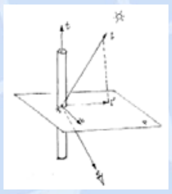
Hair Models – Explicit Hair Models (rendering)

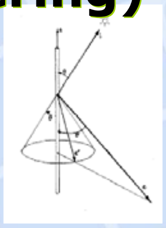
LeBlanc *et al,* 1991

- Each hair is drawn as illuminated polyline
- Use of graphics hardware for speedy line drawing
- Anti-aliasing using pixel blending Single hair thickness only partially covers a pixel width. Thus hair is drawn with a transparency to avoid aliasing.
- Drawing order has to be from back to front

Hair Models – Explicit Hair Models (rendering)

SIGGRAPH 2003 SAN DIEGO


Normal reflection model


Reflection from thin cylinder

Shadows using shadow buffer

LeBlanc *et al*, 1991

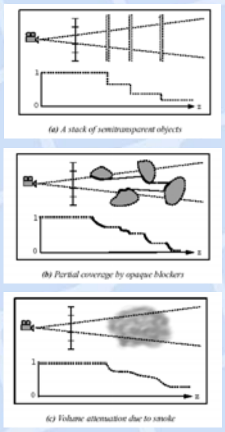
- Illumination of polylines
- Thin cylindrical geometry, source of high anisotropy in shading
- Overall shadowing can be achieved using traditional shadow maps

  Shadow map is depth map from lights point of view

  No self-shadowing

Hair Models – Explicit Hair Models (rendering)

SIGGRAPH 2003 SAN DIEGO


(a) A stack of semitransparent objects
(b) Partial coverage by opaque blockers
(c) Volume attenuation due to smoke
Deep shadow maps

Lokovic *et al*, 2000

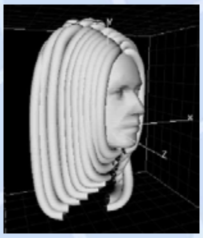
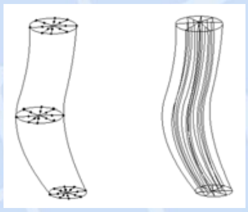
- Self-shadowing can be done using volume ray casting or “deep shadow maps”

Hair Models – Cluster Hair Models (styling)

Zhan Xu,et al, 1999

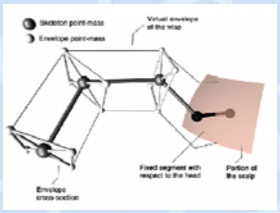
- Or wisp models
- Clump of hair is considered for modeling instead of individual hair strand
- Clumps modeled as *generalized cylinders.*

Hair Models – Cluster Hair Models (styling)

- Generalized cylinder controls overall shape
- Stochastic density variation confined to the generalized cylinder gives details of hair.
- Better control over shape.
- Efficient for rendering. Ray-tracing of generalized cylinders.
- Not suitable for smooth, simple hair

Hair Models – Cluster Hair Models (dynamics)



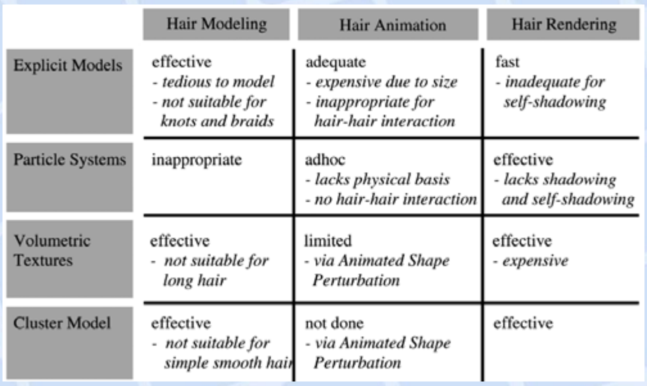
Plante *et al*, 2001

- Layered wisp model
- A skeleton curve that defines large scale motion and deformations. It is modeled as spring-mass system.
- A deformable envelop that coats the skeleton, and defines the deformation of wisp sections
- A certain number of hair strands are distributed inside the envelope and only used for rendering.

Hair Simulation Overview

Tasks

Models



	Hair Modeling	Hair Animation	Hair Rendering
Explicit Models	effective - *tedious to model* - *not suitable for knots and braids*	adequate - *expensive due to size* - *inappropriate for hair-hair interaction*	fast - *inadequate for self-shadowing*
Particle Systems	inappropriate	adhoc - *lacks physical basis* - *no hair-hair interaction*	effective - *lacks shadowing and self-shadowing*
Volumetric Textures	effective - *not suitable for long hair*	limited - *via Animated Shape Perturbation*	effective - *expensive*
Cluster Model	effective - *not suitable for simple smooth hair*	not done - *via Animated Shape Perturbation*	effective

Open Problems in Hair Simulation

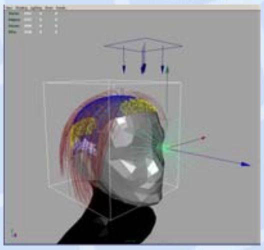
SIGGRAPH 2003 SAN DIEGO

- Hair Shape Modeling
  - Many recent attempts, however not yet matured
- Hair Dynamics
  - Stiffness dynamics of individual hair was grossly approximated considering current computing power (1GHz, 1GB RAM)
  - Hair-hair interaction not attempted (until recently)
- Hair Rendering
  Fairly matured and is available through commercial systems. However, global-illumination for hair with detailed shading models is not yet attempted.
- Real-time hair animation and rendering
  With possible compromise on the realism

Our Approach

SIGGRAPH 2003 SAN DIEGO

- Hair Shape Modeling*
  - Hair shape as streamlines of fluid flow
- Hair Rendering
  - Single iteration hair rendering including volumetric shadows using graphics hardware



*Sunil Hadap and Nadia Magnenat-Thalmann. "Interactive Hair Styler based on Fluid Flow", Eurographics Workshop on Computer Animation and Simulation'2000, Interlaken 2000

Our Approach(2)

SIGGRAPH 2003 SAN DIEGO

- Hair Dynamics**
  - Stiffness dynamics of non-straight hair strand
  - Elaborate inertial dynamics using reduced coordinate formulation
  - Hair-hair, hair-body and hair-air interaction dynamics

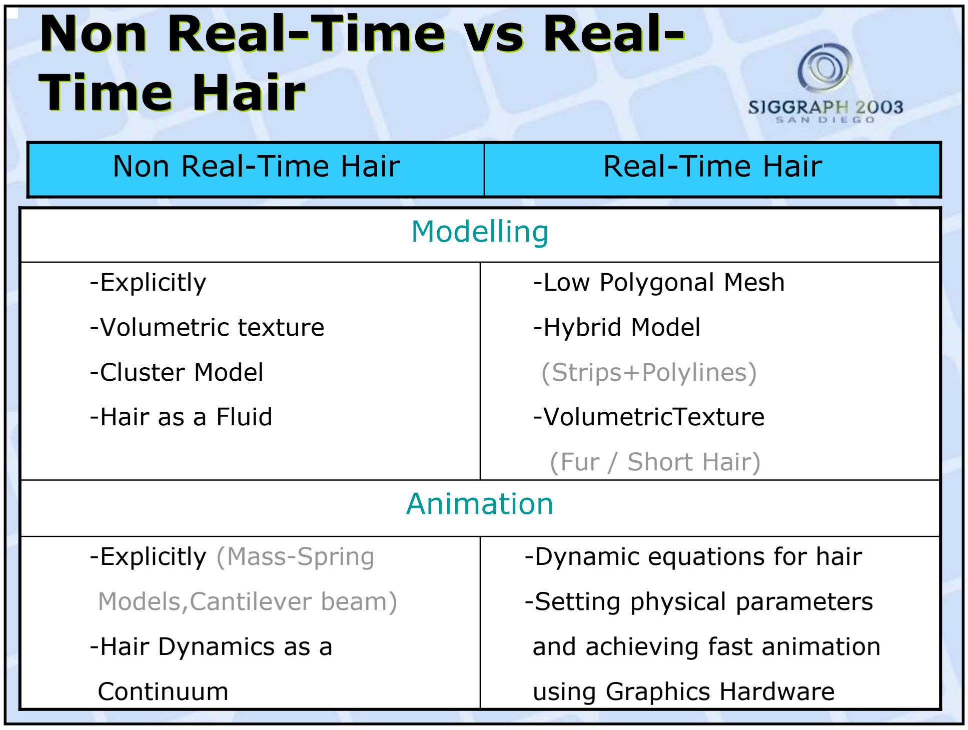
movie

**Sunil Hadap, Nadia Magnenat-Thalmann, "Modeling Dynamic Hair as a Continuum", Computer Graphics Forum, Volume 20, Issue 3, *Eurographics 2001 Proceedings*, Manchester, United Kingdom, September 2001

Real-Time Hair

SIGGRAPH 2003 SAN DIEGO

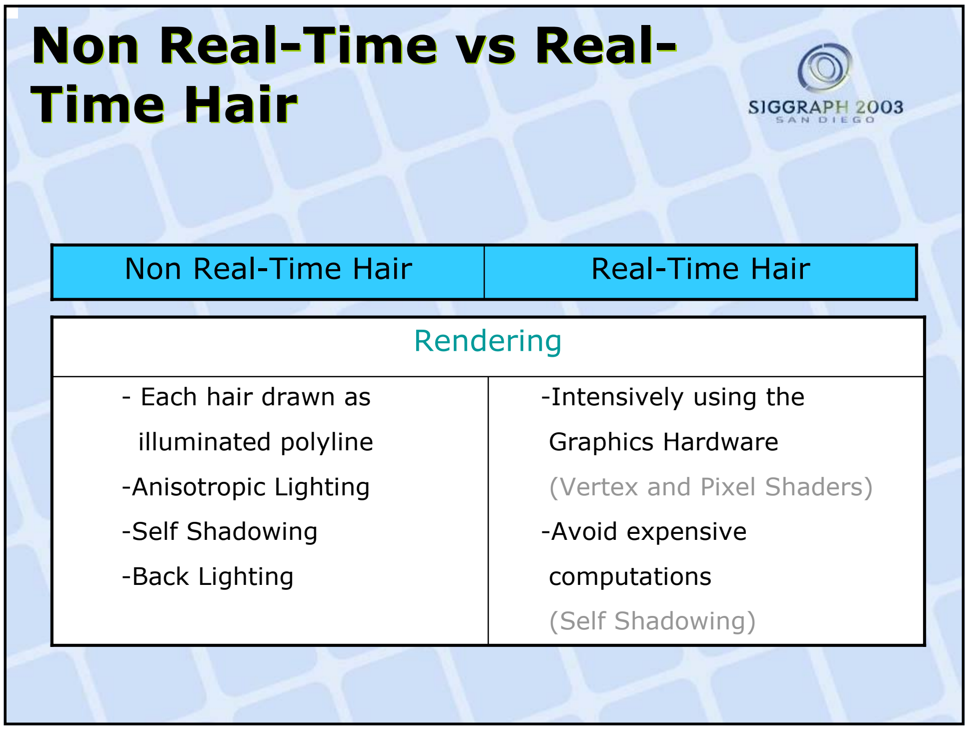
- Rendering of Hair at interactive rates
- Animating Hair according to the various physical forces in real time



Non Real-Time vs Real-Time Hair

SIGGRAPH 2003 SAN DIEGO

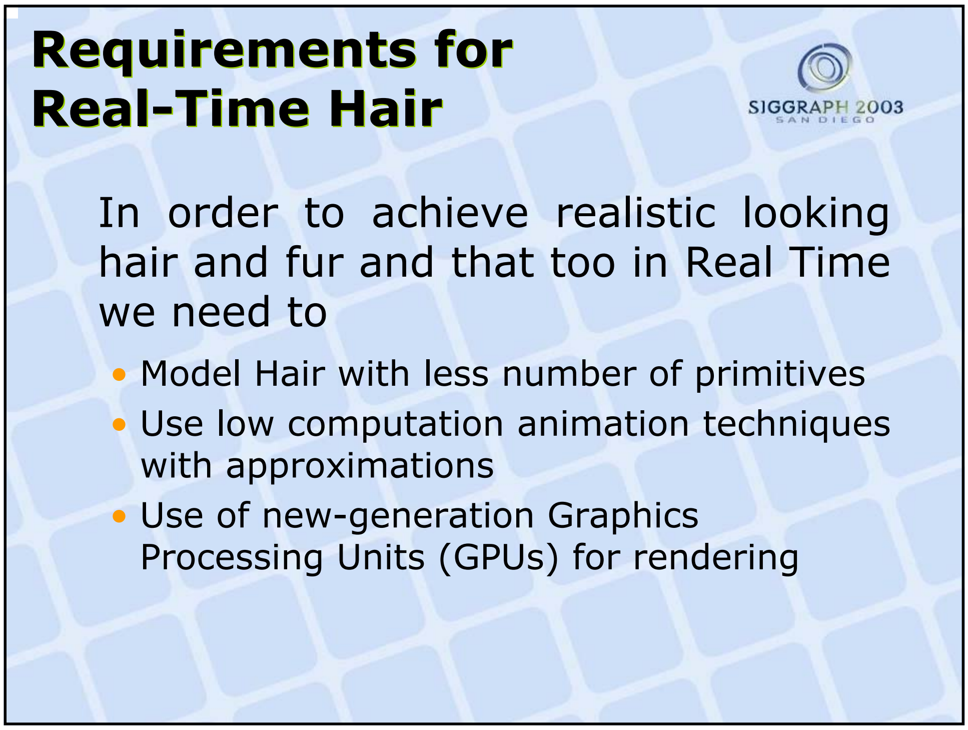
Non Real-Time Hair	Real-Time Hair
Modelling	
-Explicitly -Volumetric texture -Cluster Model -Hair as a Fluid	-Low Polygonal Mesh -Hybrid Model (Strips+Polylines) -VolumetricTexture (Fur / Short Hair)
Animation	
-Explicitly (Mass-Spring Models,Cantilever beam) -Hair Dynamics as a Continuum	-Dynamic equations for hair -Setting physical parameters and achieving fast animation using Graphics Hardware



Non Real-Time vs Real-Time Hair

SIGGRAPH 2003 SAN DIEGO

Non Real-Time Hair	Real-Time Hair
Rendering	
- Each hair drawn as illuminated polyline -Anisotropic Lighting -Self Shadowing -Back Lighting	-Intensively using the Graphics Hardware (Vertex and Pixel Shaders) -Avoid expensive computations (Self Shadowing)

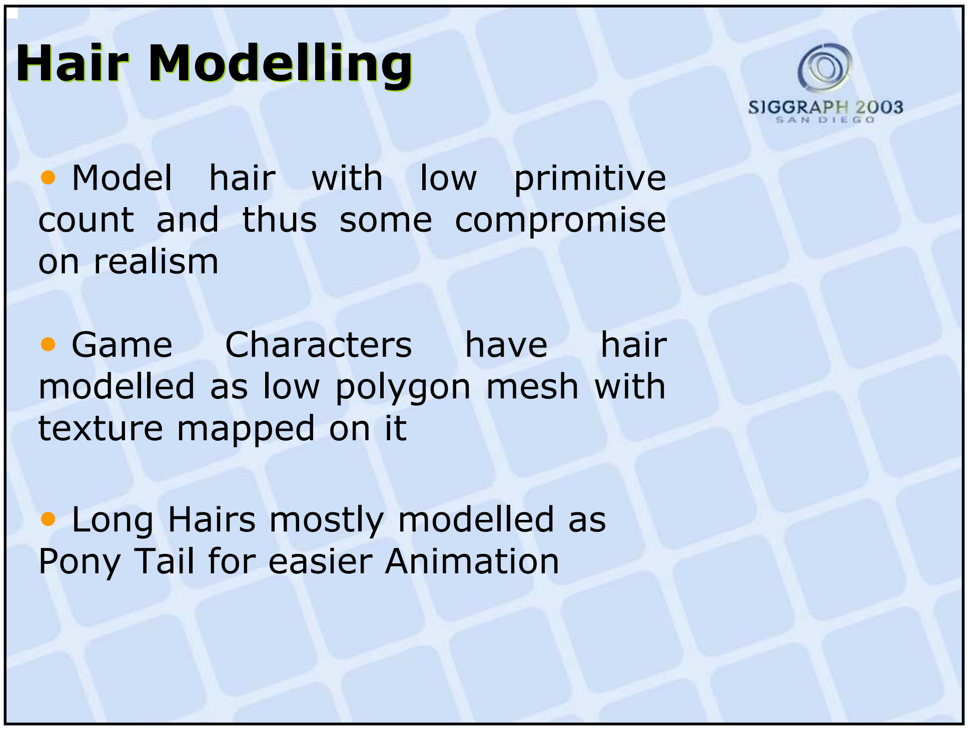


Requirements for Real-Time Hair

SIGGRAPH 2003 SAN DIEGO

In order to achieve realistic looking hair and fur and that too in Real Time we need to

- Model Hair with less number of primitives
- Use low computation animation techniques with approximations
- Use of new-generation Graphics Processing Units (GPUs) for rendering



Hair Modelling

SIGGRAPH 2003 SAN DIEGO

- Model hair with low primitive count and thus some compromise on realism

- Game Characters have hair modelled as low polygon mesh with texture mapped on it

- Long Hairs mostly modelled as Pony Tail for easier Animation

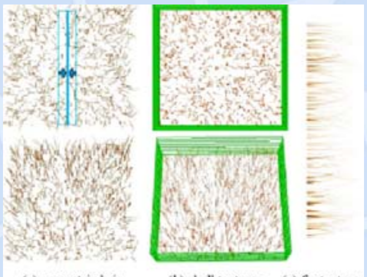
Hair Animation

- Animated Hair adds life into the virtual character
- Need to define a physical model and avoid stiff equations for achieving fast animations

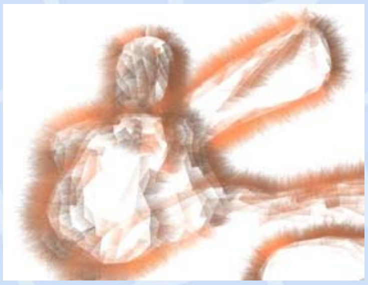
Hair Rendering

- For realism hair should show real-time modification of appearance based on viewing and lighting conditions
- Using modern powerful graphics engine makes it possible to render complex, high quality hair and fur at very interactive frame rates

Real Time Short Hair or Fur


(a) geometric hair (b) shell textures (c) fin texture

Geometrically grow a patch of hair (using a particle system) and sample it into the shell and fin textures



Fin textures used on all edges normal to the surface near silhouettes

Real-Time Fur on Arbitrary Surfaces *[Lengyel '01]*

Real Time Short Hair or Fur (2)

Shell textures are concentric, semi-transparent samples of hair volume

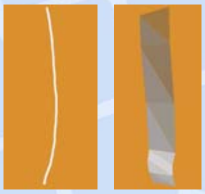
At runtime

- -Render a series of textured, concentric shells
- -Render several fins perpendicular to the surface

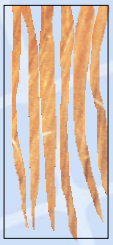
Real Time Hair Framework

SIGGRAPH 2003 SAN DIEGO

Hair Modeled as a Hybrid of Strips and Polylines (adjusting LODs depending upon viewing distance)



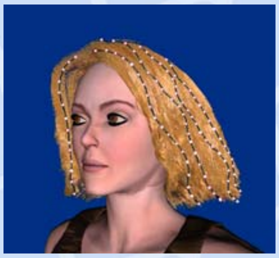
Texture map along with transparency map is applied to the Strips



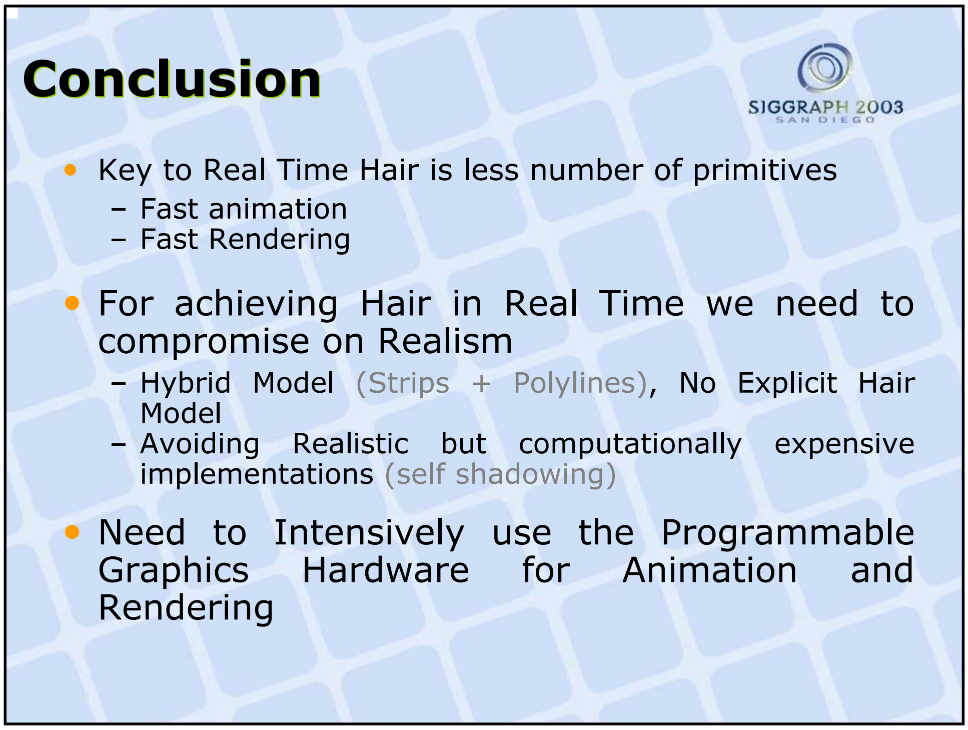
Real Time Hair Framework (2)

SIGGRAPH 2003 SAN DIEGO

Animating the hair model by defining some control strands on the hair model

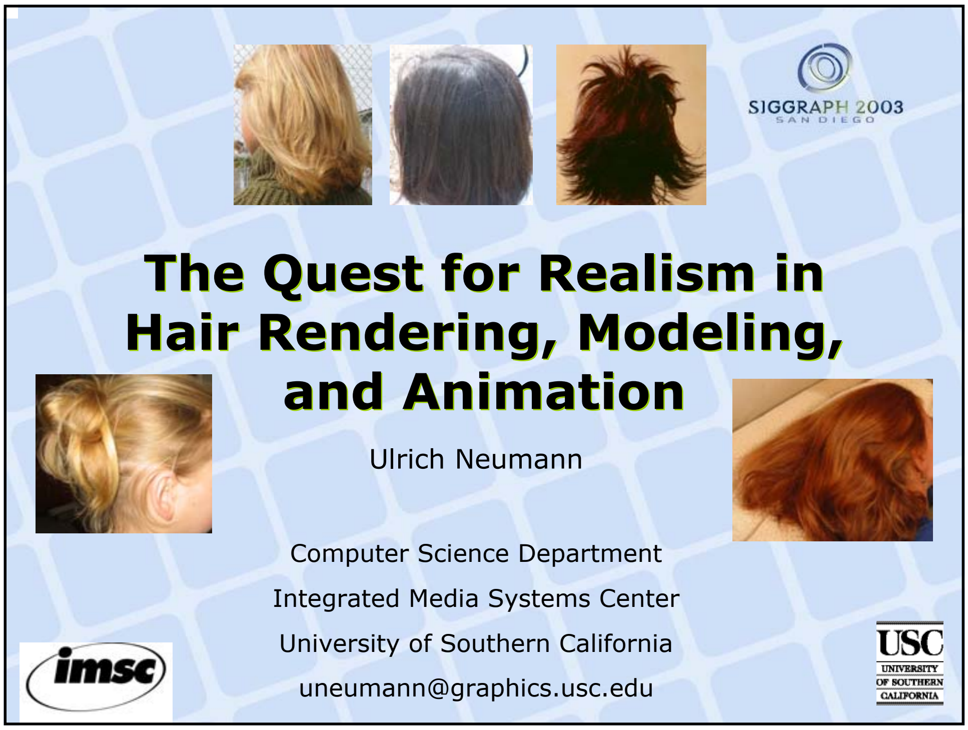


Use Vertex Shaders to Perform Anisotropic lighting of Hair



Conclusion

- Key to Real Time Hair is less number of primitives
  - Fast animation
  - Fast Rendering
- For achieving Hair in Real Time we need to compromise on Realism
  - Hybrid Model (Strips + Polylines), No Explicit Hair Model
  - Avoiding Realistic but computationally expensive implementations (self shadowing)
- Need to Intensively use the Programmable Graphics Hardware for Animation and Rendering



The Quest for Realism in Hair Rendering, Modeling, and Animation

Ulrich Neumann

Computer Science Department

Integrated Media Systems Center

University of Southern California

uneumann@graphics.usc.edu

Background

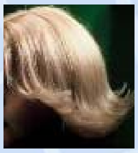
SIGGRAPH 2003 SAN DIEGO

- Graphics methods evolved for surfaces and volumes (sampled continuum)
  - Suited to most of the world and data that people want to model and visualize
- Long human hair is neither surface nor continuum
  - A unique problem for modeling, rendering, and animating
- This talk presents a summary of hair characteristics that lead to unique problems in graphics...

Hair is...

SIGGRAPH 2003 SAN DIEGO

- Strands – lots of them... (100-150K)
  - Long curved strands are represented by multiple line segments (for speed), resulting in >Million segments to represent a hair style
- High complexity - lots of primitives to control, and lots of interactions, collisions, occlusions
  - Algorithm efficiency is important!
- Chemistry analogy...
  - Surfaces are solids
  - Continuum volumes are liquids
  - Hair is a *suspension*
    - Small surfaces (solids) held in a volume (liquid)
    - Characteristics of both, but also unique properties



Lighting and Shading

SIGGRAPH 2003 SAN DIEGO

- Volumetric effects
  - Scattering, shadows, and extinction
- Multi-scale structures
  - Strands - wisps/clusters - layers
  - Anisotropic reflectance model

Sampling Problems

SIGGRAPH 2003 SAN DIEGO

- Hair has extreme anisotropic geometry
  - micro cross-section and macro length
- Pixel value = aggregate shading and occlusion effects of multiple strands
- Supersampling to strand resolution is impractical
  - Strand color accumulates at pixels
  - Occlusion/accumulation is a volumetric density function

Modeling

- Endless variations of hairstyles
  - No constraints between hair strands and head/body model, other than interpenetration
- Multi-scale structures
  - Strands - wisps/clusters - layers

Animation

- Deformation and motion are common... Influenced by
  - Inertia, coulomb forces and friction between hairs, body collisions, gravity, and air (relative motion)
- Bending is easy - stretching is not
  - Soft and stiff system at the same time
- Multi-scale dynamic clustering and breakup under motion
  - Hairstyle does not behave as a single deformable object – clusters form and breakup
  - Elastic limit – style memory vs breakup

(Movie illustrates real hair motions)

Hair Synthesis System

- Modeling and rendering (and animation) are coupled...
  - Can’t see model without a renderer
  - Can’t render (or animate) without a model etc...
- Need to make all the functions of a graphics system “hair-capable” to do any work in this area

A Multiresolution Technique for Hairstyling

Tae-Yong Kim

tae@rhythm.com

Rhythm & Hues Studio

*formerly at the University of Southern California

Modeling Human Hair

SIGGRAPH 2003
SAN DIEGO

- Complex discontinuous volume
- Clusters and curliness
- Virtually any shape is possible
- Number of hair strands (100K

To 200K)

Goal

SIGGRAPH 2003
SAN DIEGO

To build an interactive system for modeling

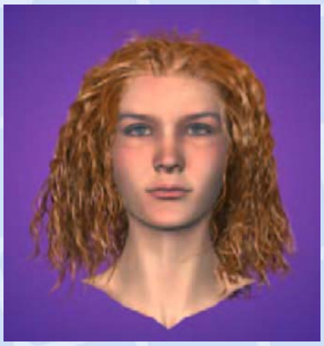
Complex human hair in a reasonable amount of time

(< 1 hour)

- Efficient sculpting of volumetric hair model
- Interactive rendering of arbitrary explicit hair

Models

Goal

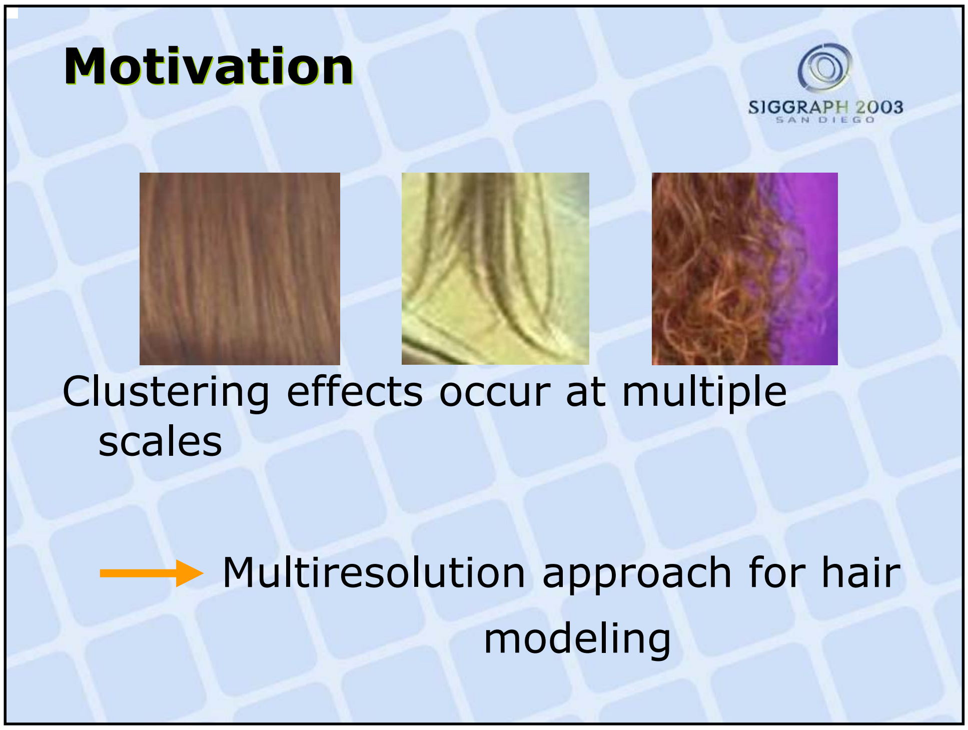

A multi-res hair Model

photograph

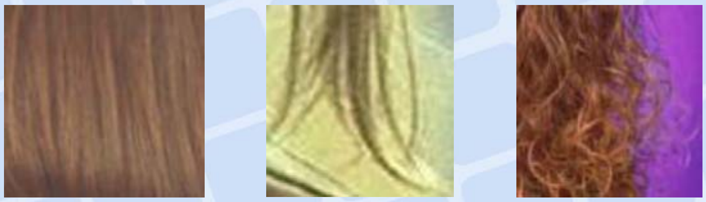
Common Problems

Long hairs cluster, split, and curl away

→ Complex hairstyles with extreme discontinuity are difficult to model

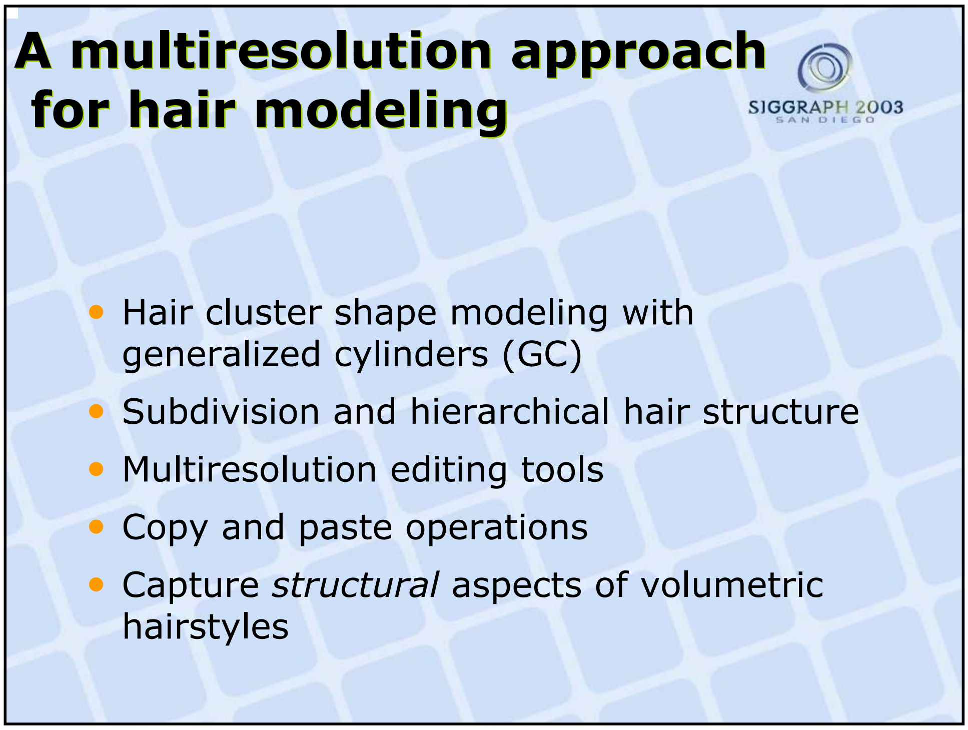


Motivation



Clustering effects occur at multiple scales

→ Multiresolution approach for hair modeling



A multiresolution approach for hair modeling

- Hair cluster shape modeling with generalized cylinders (GC)
- Subdivision and hierarchical hair structure
- Multiresolution editing tools
- Copy and paste operations
- Capture *structural* aspects of volumetric hairstyles

Scalp Surface

SIGGRAPH 2003 SAN DIEGO

- A parametric surface

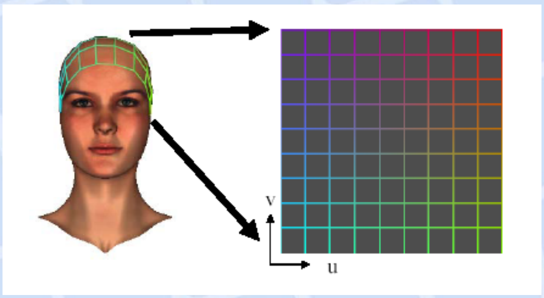
$S = p(u,v)$

- Defines the region of hair

Growth

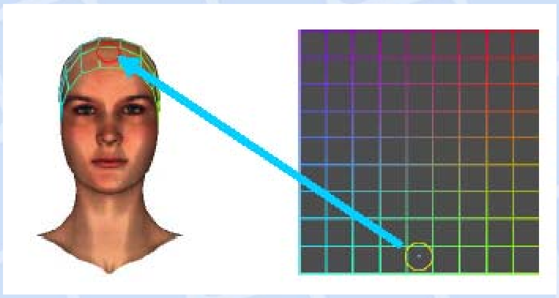
Scalp Surface

SIGGRAPH 2003 SAN DIEGO



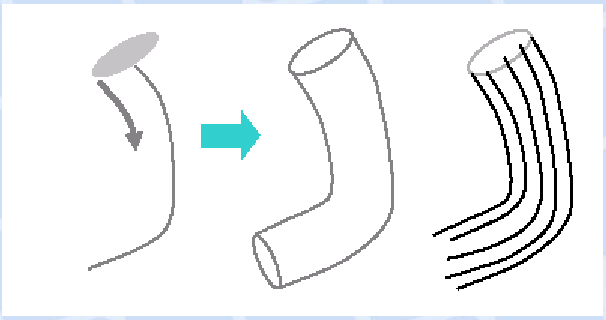
Scalp space editor

Scalp Surface

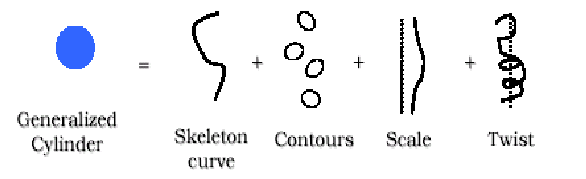


Atlas for hair cluster positioning

Generalized Cylinder (GC)

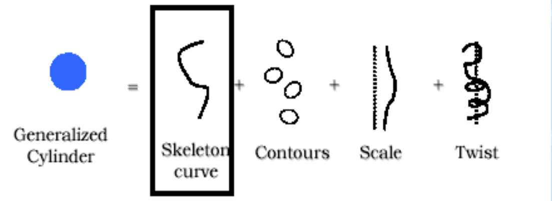


Generalized Cylinder (GC)



$$V(r,\theta,t) = C(t) + rR(\theta,t)\left\{\cos(\hat{\theta})S_N(t)\vec{N}(t) + \sin(\hat{\theta})S_B(t)\vec{B}(t)\right\}$$

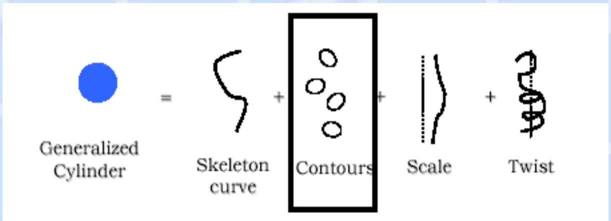
Generalized Cylinder (GC)



$$V(r,\theta,t) = \boxed{C(t)} + rR(\theta,t)\left\{\cos(\hat{\theta})S_N(t)\vec{N}(t) + \sin(\hat{\theta})S_B(t)\vec{B}(t)\right\}$$

Generalized Cylinder (GC)

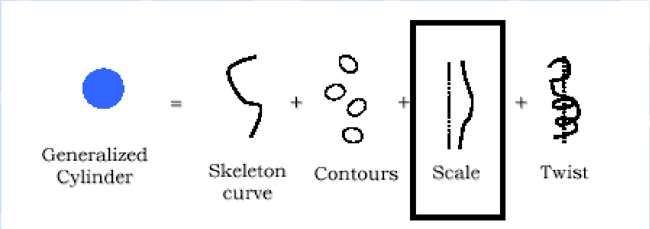
SIGGRAPH 2003 SAN DIEGO



$$V(r,\theta,t)=C(t)+\boxed{rR(\theta,t)}\left\{\cos(\hat{\theta})S_N(t)\vec{N}(t)+\sin(\hat{\theta})S_B(t)\vec{B}(t)\right\}$$

Generalized Cylinder (GC)

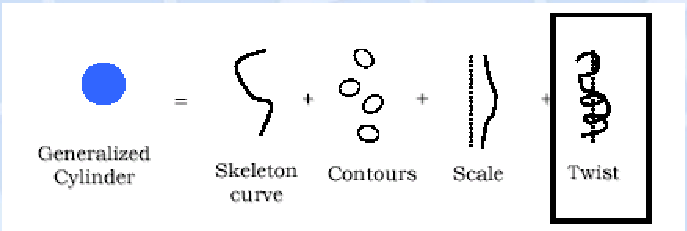
SIGGRAPH 2003 SAN DIEGO



$$V(r,\theta,t)=C(t)+rR(\theta,t)\left\{\cos(\hat{\theta})\boxed{S_N(t)}\vec{N}(t)+\sin(\hat{\theta})\boxed{S_B(t)}\vec{B}(t)\right\}$$

Generalized Cylinder (GC)

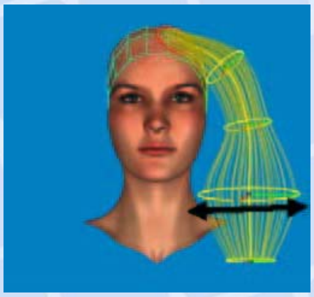
SIGGRAPH 2003 SAN DIEGO



$$V(r,\theta,t)=C(t)+rR(\theta,t)\{\cos\boxed{(\hat{\theta})}S_N(t)\vec{N}(t)+\sin\boxed{(\hat{\theta})}S_B(t)\vec{B}(t)\}$$

Scale Change

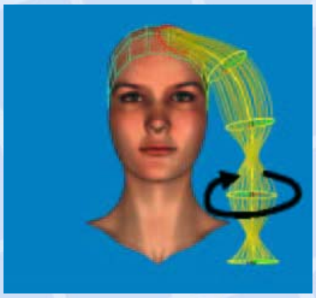
SIGGRAPH 2003 SAN DIEGO



$$V(r,\theta,t)=C(t)+rR(\theta,t)\{\cos(\hat{\theta})\boxed{S_N(t)}\vec{N}(t)+\sin(\hat{\theta})\boxed{S_B(t)}\vec{B}(t)\}$$

Twist Change

SIGGRAPH 2003 SAN DIEGO



$$V(r,\theta,t) = C(t) + rR(\theta,t)\left\{\cos(\boxed{\hat{\theta})S}_N(t)\vec{N}(t) + \sin(\hat{\theta})\boxed{S_B(t)}\vec{B}(t)\right\}$$

Adding Details

SIGGRAPH 2003 SAN DIEGO

- A single generalized cylinder (GC) is limited by a few parameters
- Simply adding more gcs results in too many Controls

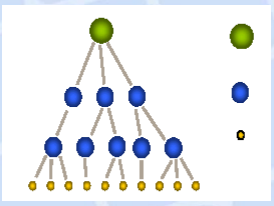
Subdivision

SIGGRAPH 2003 SAN DIEGO

- Subdivide a parent GC into several smaller child gcs
- Hierarchical control over the hair model
- Editing a child GC changes the shape of hair strands
- Editing a parent GC affects child gcs as well as hair strands

Hair Tree

SIGGRAPH 2003 SAN DIEGO

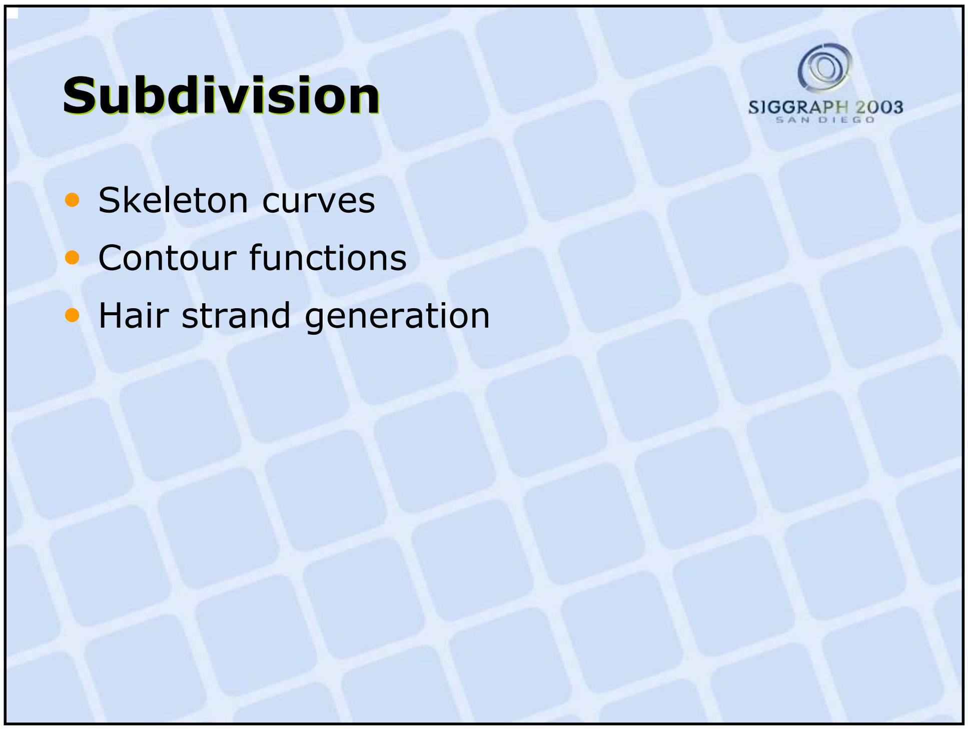


Generalized cylinder

A hair model

Hair strand

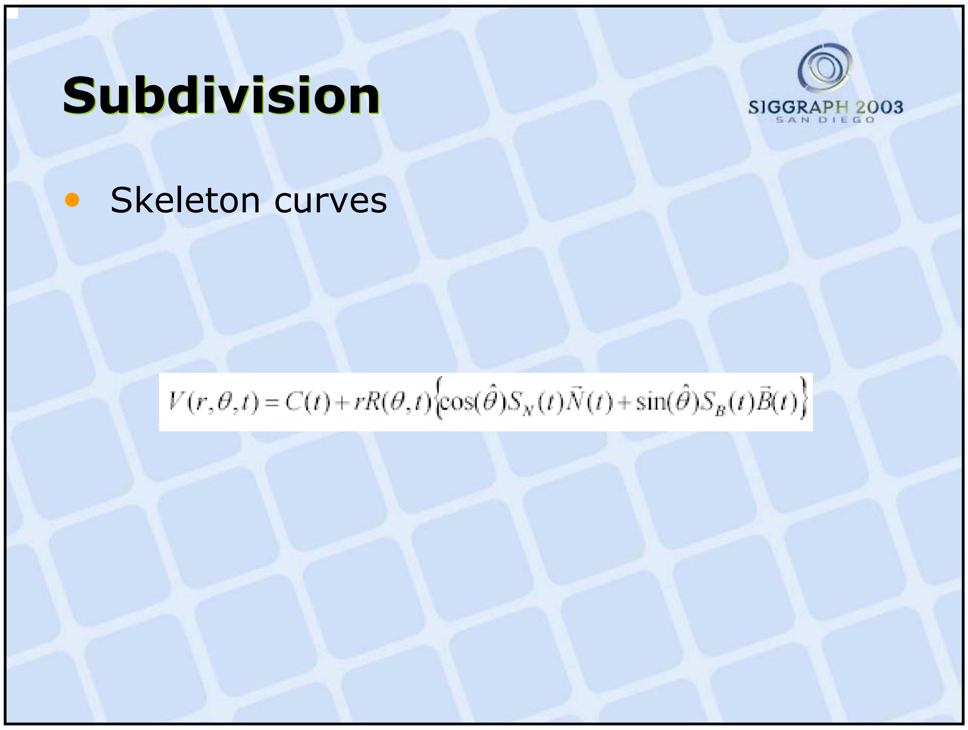
- Subdivision allows more refined controls to the hair model $\longrightarrow$ down to a hair strand



Subdivision

SIGGRAPH 2003 SAN DIEGO

- Skeleton curves
- Contour functions
- Hair strand generation



Subdivision

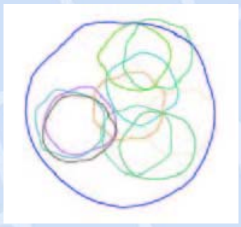
SIGGRAPH 2003 SAN DIEGO

- Skeleton curves

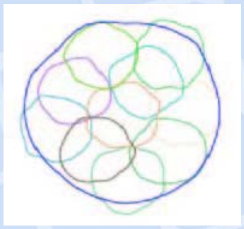
$$V(r,\theta,t) = C(t) + rR(\theta,t)\left\{\cos(\hat{\theta})S_N(t)\vec{N}(t) + \sin(\hat{\theta})S_B(t)\vec{B}(t)\right\}$$

Subdivision

- Contour functions



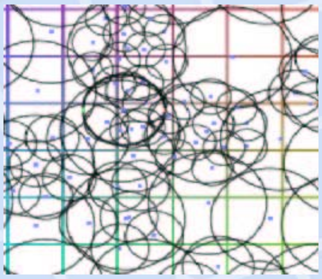
1. Random Positioning



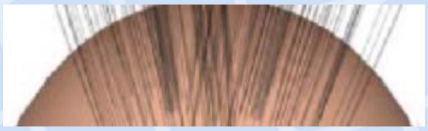
2. Position Relaxation [Turk1991]

Subdivision

- Hair strand generation



Overlaps and holes

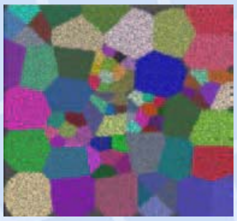


Uneven hair density

Subdivision

SIGGRAPH 2003 SAN DIEGO

- Hair strand generation


Hair assignment

Consistent hair density

Multiresolution Editing

SIGGRAPH 2003 SAN DIEGO

Multiresolution Editing

SIGGRAPH 2003 SAN DIEGO

- Bind

When a GC is selected for edit, its descendants
Are attached to the GC

- Update

During editing, the children gcs are updated
Using GC equation of the parent

Multiresolution Editing

SIGGRAPH 2003 SAN DIEGO

- Bind: $P = V(r,\theta,t)$

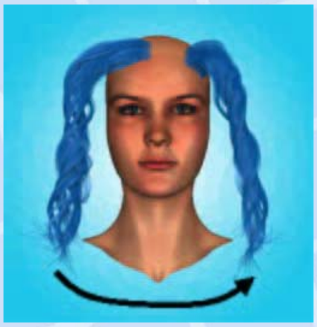
$(r,\theta,t)=V^{-1}(P)$

$V(r,\theta,t) \rightarrow V'(r,\theta,t)$

- Update: $P' = V'(r,\theta,t)$

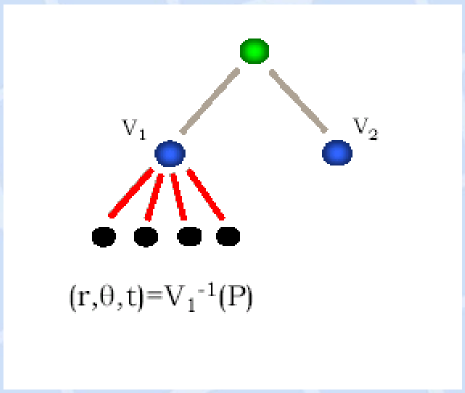
$$V(r,\theta,t) = C(t) + rR(\theta,t)\left\{\cos(\hat{\theta})S_N(t)\vec{N}(t) + \sin(\hat{\theta})S_B(t)\vec{B}(t)\right\}$$

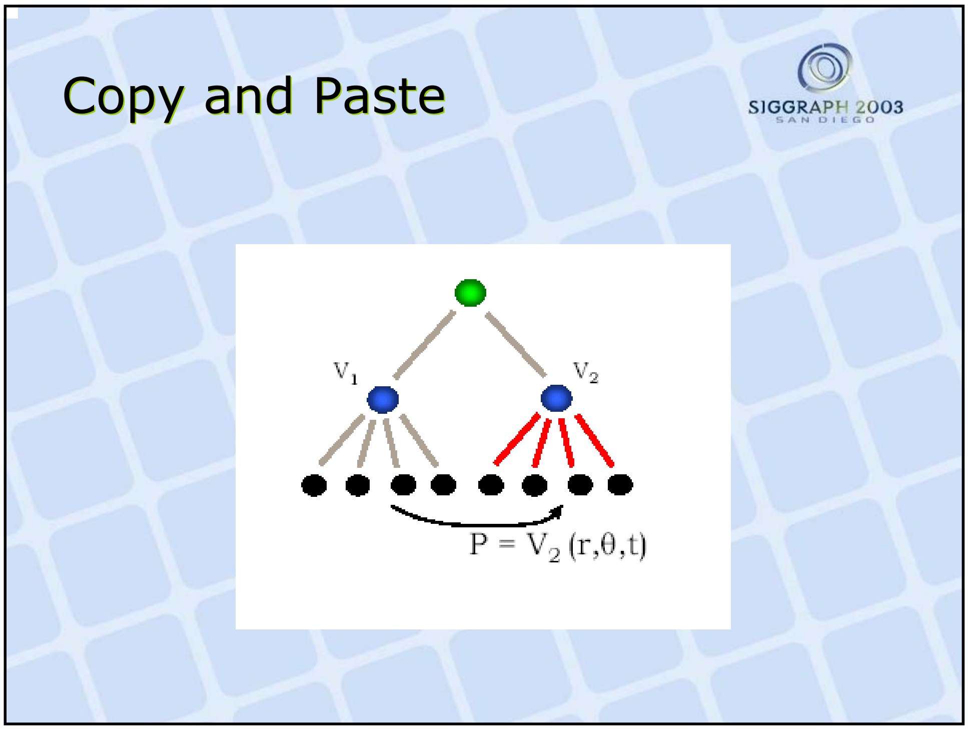
Copy and Paste



Transfer a *style* from one cluster to another

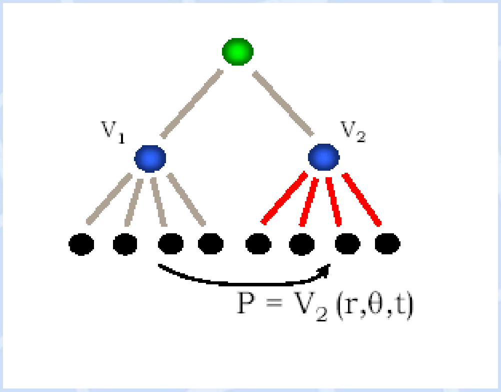
Copy and Paste

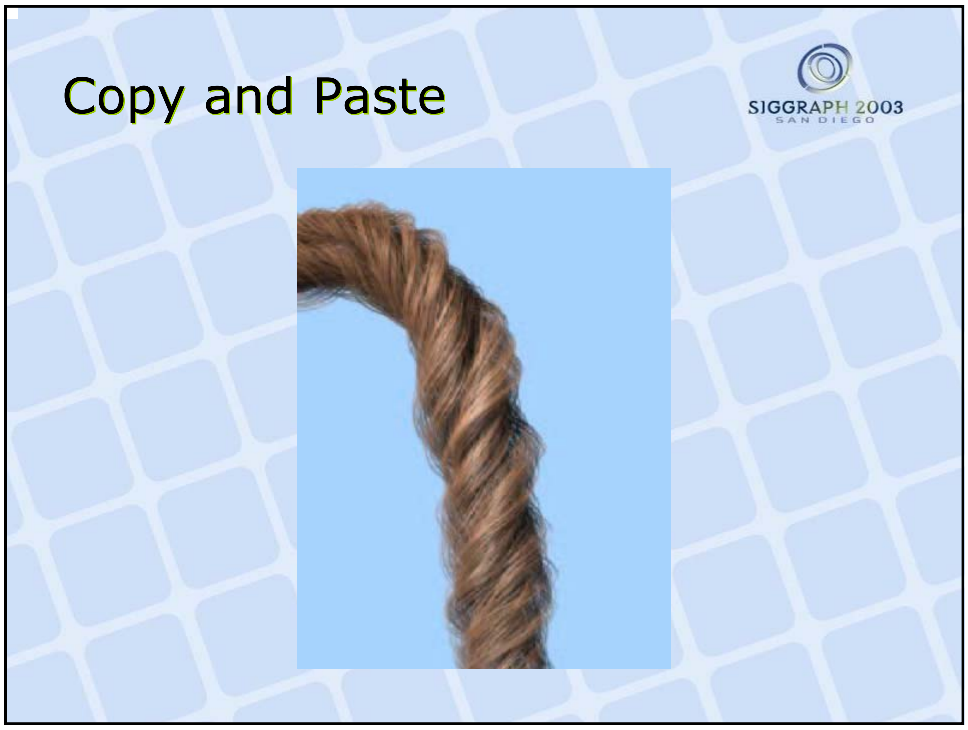




Copy and Paste

SIGGRAPH 2003 SAN DIEGO

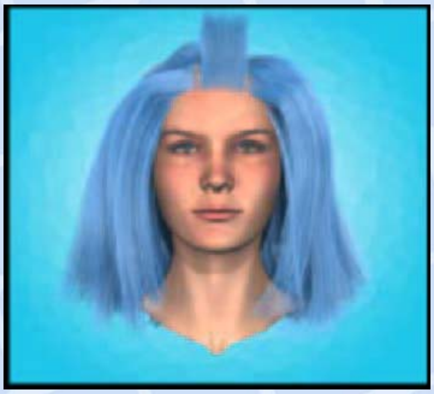




Copy and Paste

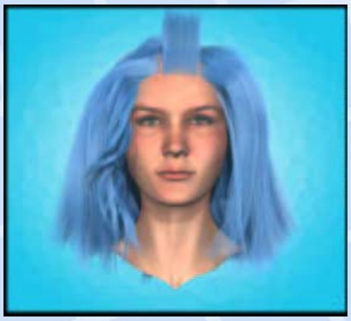
SIGGRAPH 2003 SAN DIEGO

Example hairstyling procedure



Global hair shape modeling with GCs

Example hairstyling procedure



Subdivision and local editing

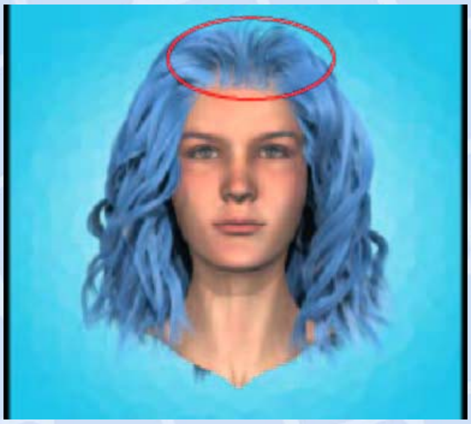
Example hairstyling procedure

SIGGRAPH 2003
SAN DIEGO

Copy and paste

Example hairstyling procedure

SIGGRAPH 2003
SAN DIEGO



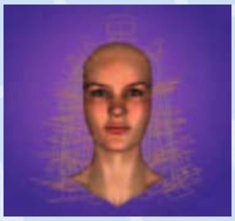
Multiresolution editing and refinement

The example we saw in the beginning

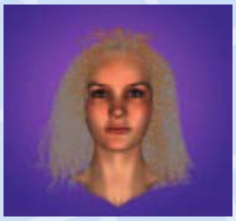
SIGGRAPH 2003 SAN DIEGO

Example

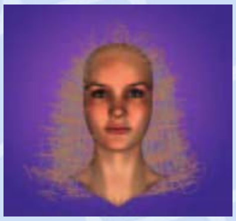
SIGGRAPH 2003 SAN DIEGO


Top level

(30 GCs)


2nd level

(177 GCs)


3rd level

(840 GCs)

Interactive Rendering

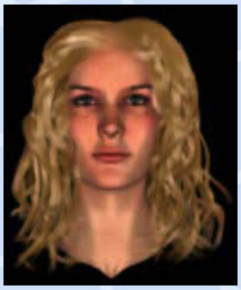
SIGGRAPH 2003 SAN DIEGO

- Each hair strand is drawn as opengl lines.
- Integrated rendering system with openGL hardware
- Instant feedback for WYSIWYG hair modeling

→ Interactive visualization of complex hair geometry during modeling

Level of Detail

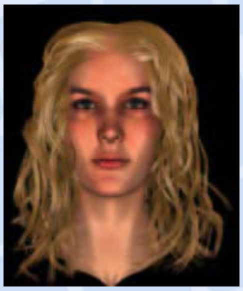
SIGGRAPH 2003 SAN DIEGO



High complexity model

1.2M lines

20000 strands, 60 lines / strand, a = 0.3

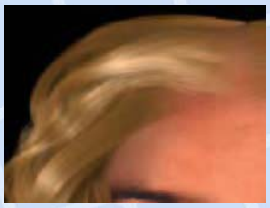


Low complexity model

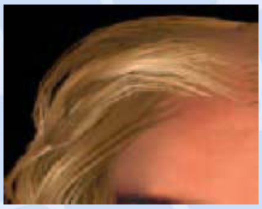
(100K lines)

5000 strands, 20 lines / strand, a = 1.0

Level of Detail

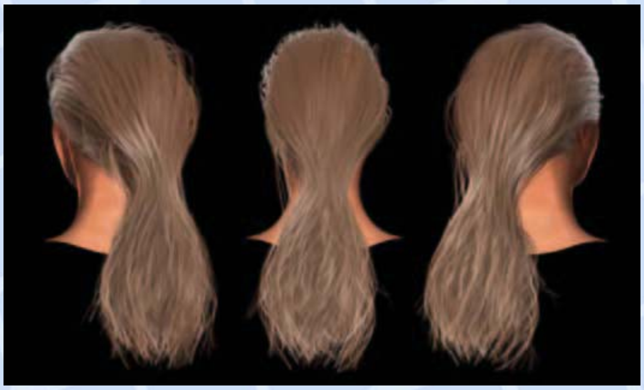


High complexity model
1.2M lines



Low complexity model
(100K lines)

Results - Curly ponytail



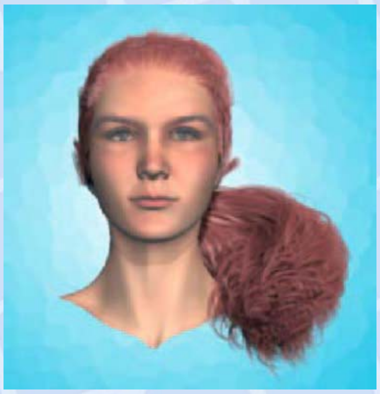
Results

SIGGRAPH 2003
SAN DIEGO

Results

SIGGRAPH 2003
SAN DIEGO

Results



Results

See how a hairstyle changes the look on the same face model!

Modeling Hair Shape as Streamlines of Fluid Flow

Sunil Hadap

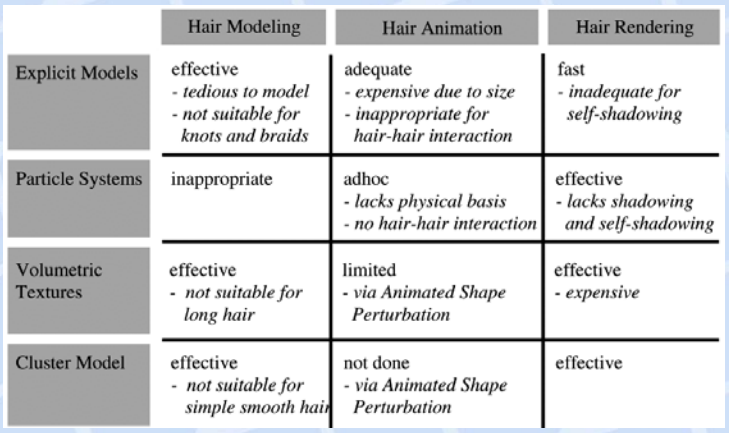
R&D staff, PDI/DreamWorks
hadap@acm.org

Doctoral research at MIRALab, university of Geneva

Hair Simulation Overview

Task

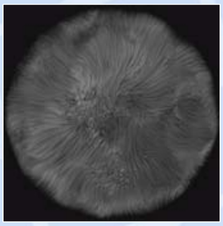
Models



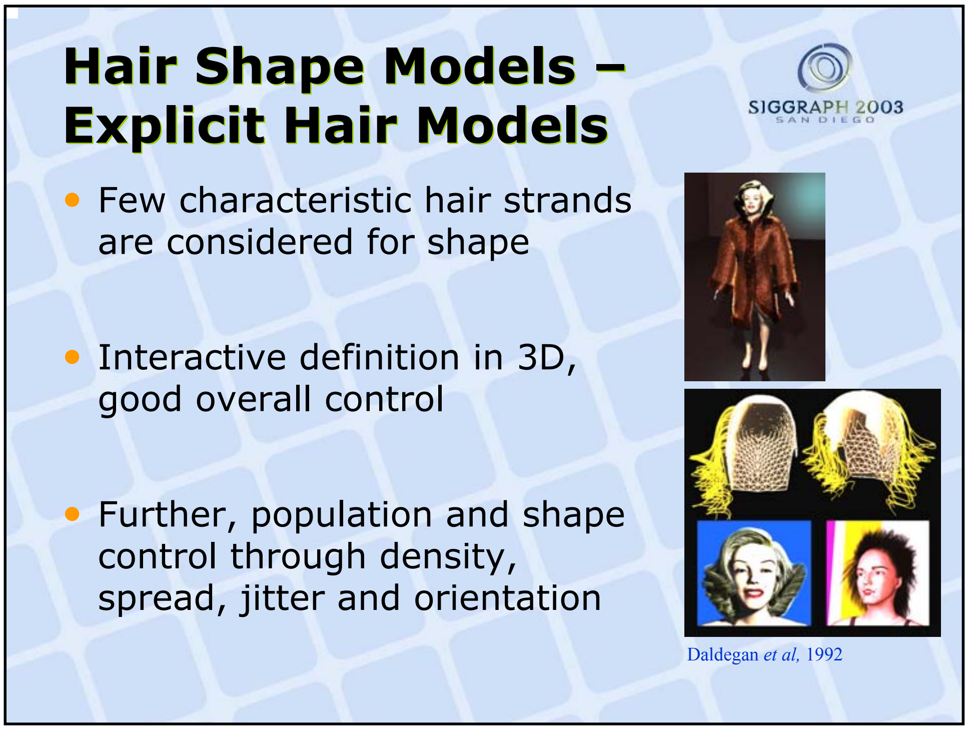
	Hair Modeling	Hair Animation	Hair Rendering
Explicit Models	effective - *tedious to model* - *not suitable for knots and braids*	adequate - *expensive due to size* - *inappropriate for hair-hair interaction*	fast - *inadequate for self-shadowing*
Particle Systems	inappropriate	adhoc - *lacks physical basis* - *no hair-hair interaction*	effective - *lacks shadowing and self-shadowing*
Volumetric Textures	effective - *not suitable for long hair*	limited - *via Animated Shape Perturbation*	effective - *expensive*
Cluster Model	effective - *not suitable for simple smooth hair*	not done - *via Animated Shape Perturbation*	effective

Hair Shape Models – Volumetric Textures

- Spatial density functions such as 3d noise and turbulence
- Stochastic models, nice way of interpreting complexity in nature such as grass, trees, forests and fur
- Analytically defined - good for anti-aliasing in rendering

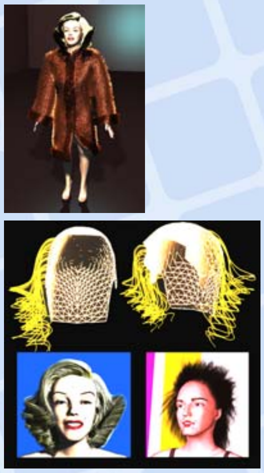

Perlin *et al*, 1989

Kajiya *et al*, 1989

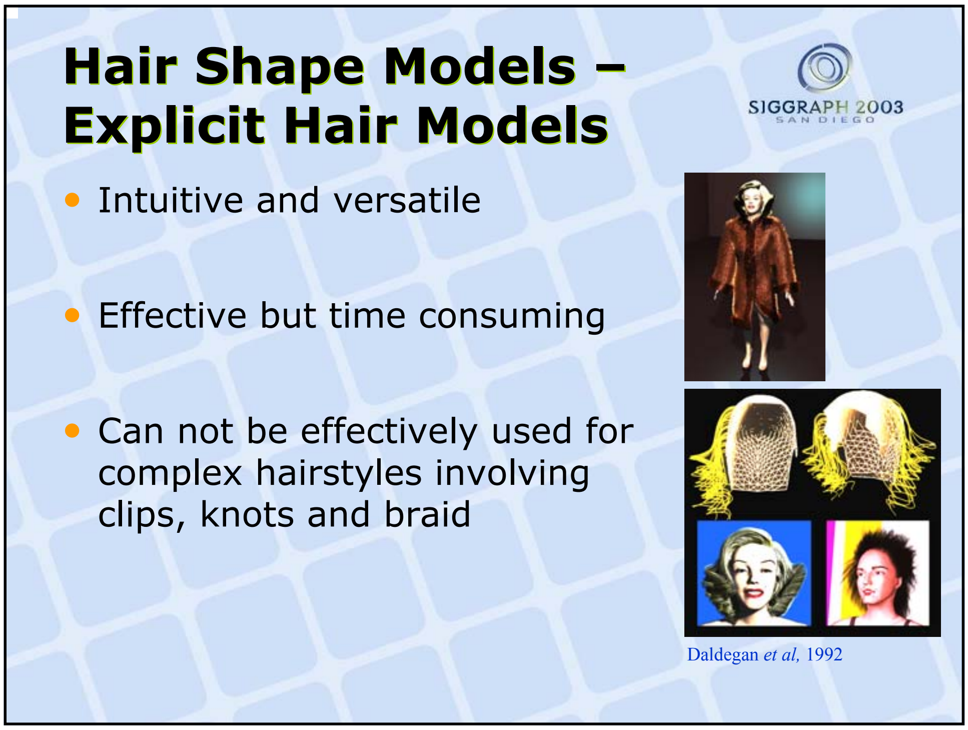


Hair Shape Models – Explicit Hair Models

- Few characteristic hair strands are considered for shape
- Interactive definition in 3D, good overall control
- Further, population and shape control through density, spread, jitter and orientation

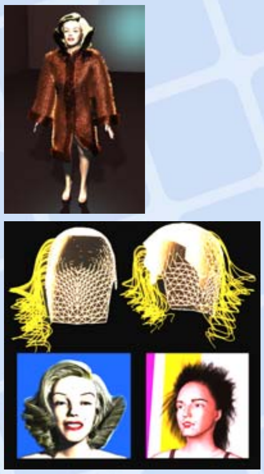


Daldegan *et al,* 1992



Hair Shape Models – Explicit Hair Models

- Intuitive and versatile
- Effective but time consuming
- Can not be effectively used for complex hairstyles involving clips, knots and braid

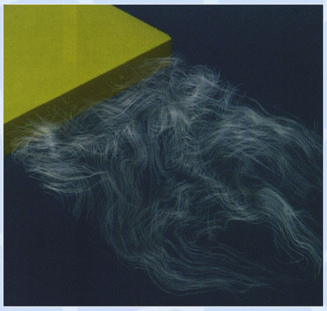


Daldegan *et al,* 1992

Best of Both – Explicit Hair Models and Volumetric Texture

SIGGRAPH 2003 SAN DIEGO

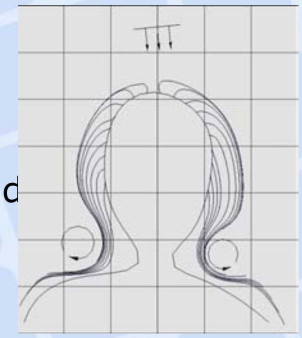
- Fluid flow has both
  - Explicit global shape control (long hair)
  - Rich local details (fur)
- Flow Visualization [Cabral93] [Zckler96]



Hair as Streamlines of Fluid Flow

SIGGRAPH 2003 SAN DIEGO

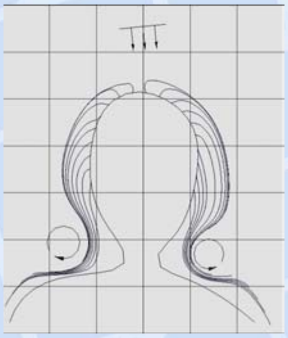
- Only a snapshot of fluid flow
- Gravity
  free stream flow condition
- Hair Growth
  secondary flow from within head
- Hair-hair Interactions
  continuum property



Hair as Streamlines of Fluid Flow

SIGGRAPH 2003 SAN DIEGO

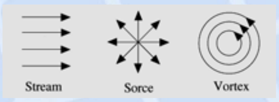
- Hair-body Interactions
  flow tangency condition
- Hair Root Lift
  flow normal velocity
- Parting Line
  stagnation point

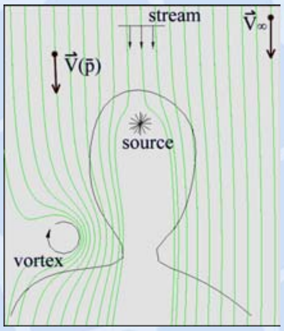


Fluid Hair Model

SIGGRAPH 2003 SAN DIEGO

- Ideal Flow
  stream, source and vortex
- And their linear combinations





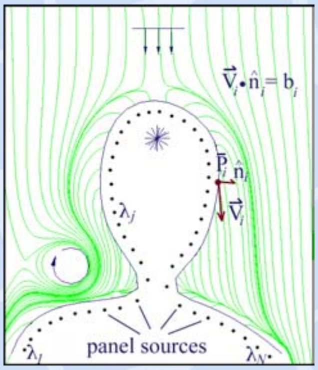
- Stream as gravity
- Source as an obstacle
- Vortex as a curler

… shape the complex flow

Panel Method – handling obstacle avoidance

SIGGRAPH 2003 SAN DIEGO

- Place a number of "unknown" sources bellow the body
- Flow Boundary Condition
- Solving for "unknown" source strengths
- Flow Form Factor, LU decomposition

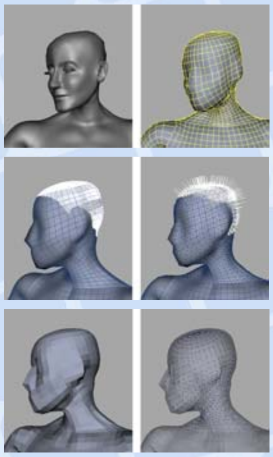


… now, one does not worry about hair-hair and hair-body collisions

Interactive Hair Styler based on Fluid Flow

SIGGRAPH 2003 SAN DIEGO

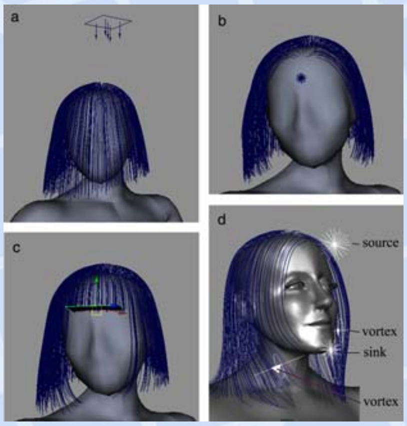
- Polygon reduced geometry to define panels
- Hair growth map and velocity map
- Placing panel sources



Hairstyling

SIGGRAPH 2003 SAN DIEGO

- Place a stream to give hair a downward direction
- Place a source to repel away unwanted hair on face
- Trim hair
- Place vortices to make curls



… global shape control

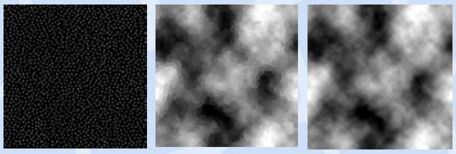
Results – simple and complex fluid flow

SIGGRAPH 2003 SAN DIEGO

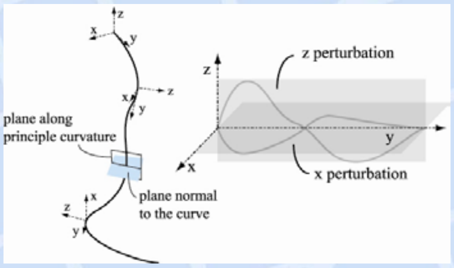
Structured Volumetric Textures

SIGGRAPH 2003 SAN DIEGO

- Volumetric perturbations waves and noise



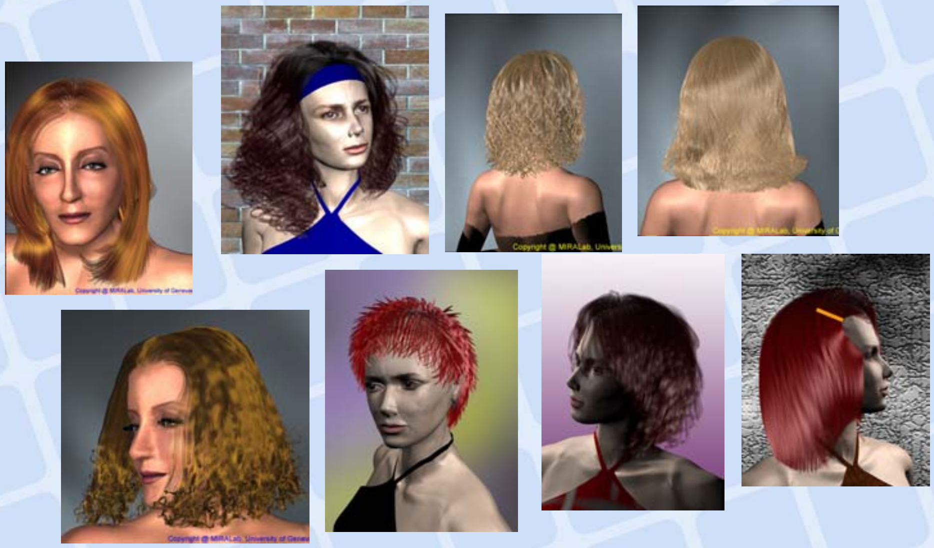
- Curvilinear coordinate system

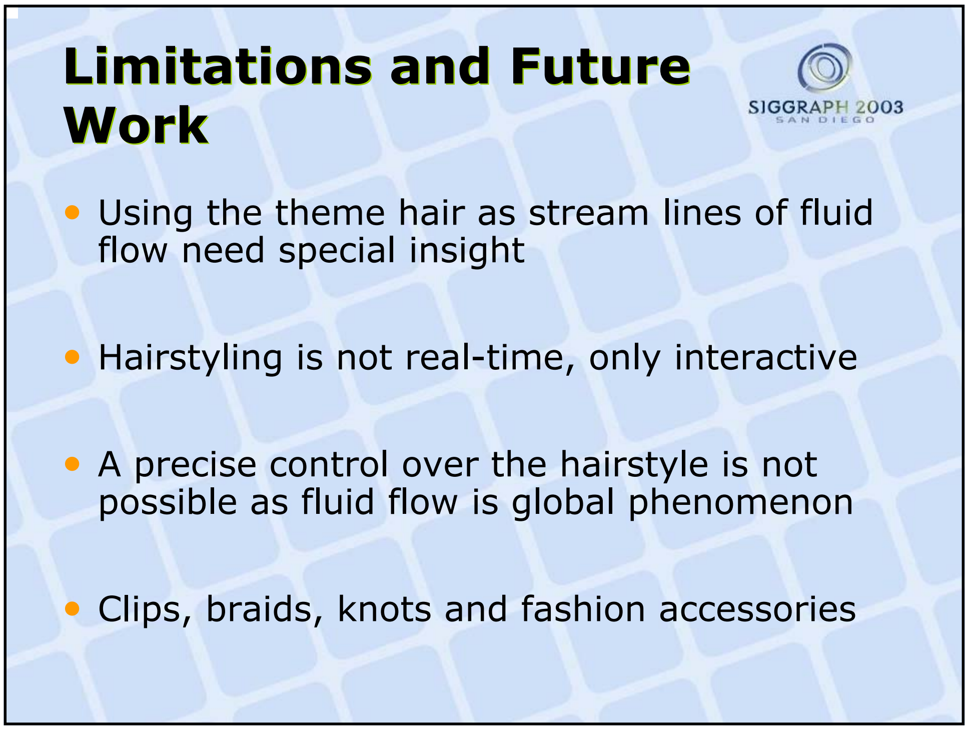


- Perturbations applied to the individual hair strand or to the clump
- Why "structured"?

Results – volumetric textures added to flow

SIGGRAPH 2003 SAN DIEGO

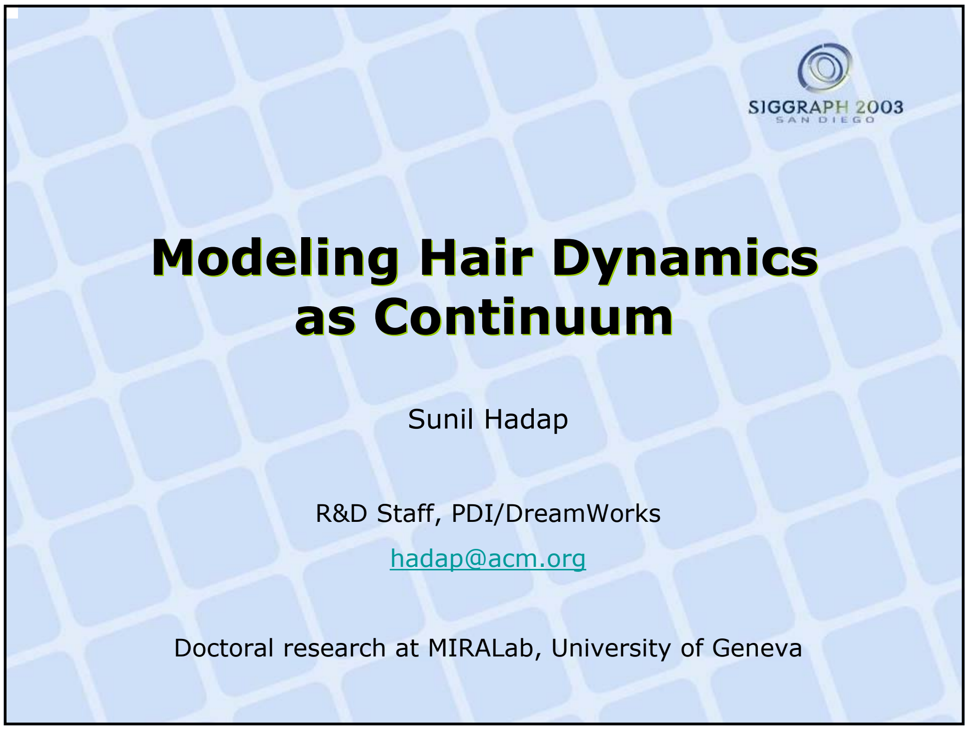




Limitations and Future Work

SIGGRAPH 2003 SAN DIEGO

- Using the theme hair as stream lines of fluid flow need special insight
- Hairstyling is not real-time, only interactive
- A precise control over the hairstyle is not possible as fluid flow is global phenomenon
- Clips, braids, knots and fashion accessories



SIGGRAPH 2003 SAN DIEGO

Modeling Hair Dynamics as Continuum

Sunil Hadap

R&D Staff, PDI/DreamWorks

hadap@acm.org

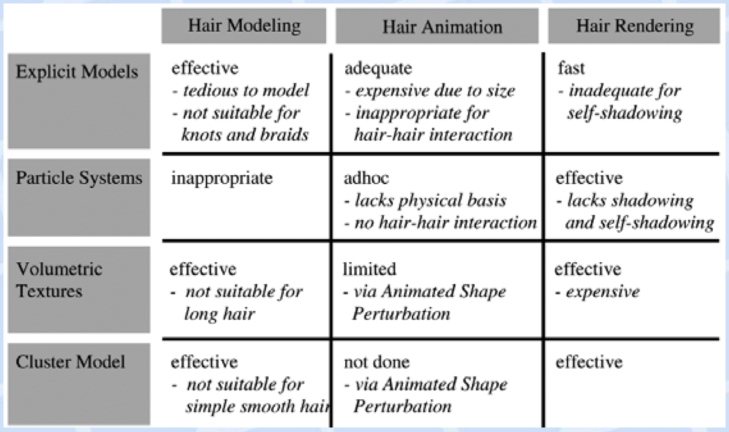
Doctoral research at MIRALab, University of Geneva

Hair Simulation Overview

SIGGRAPH 2003
SAN DIEGO

Task

Models



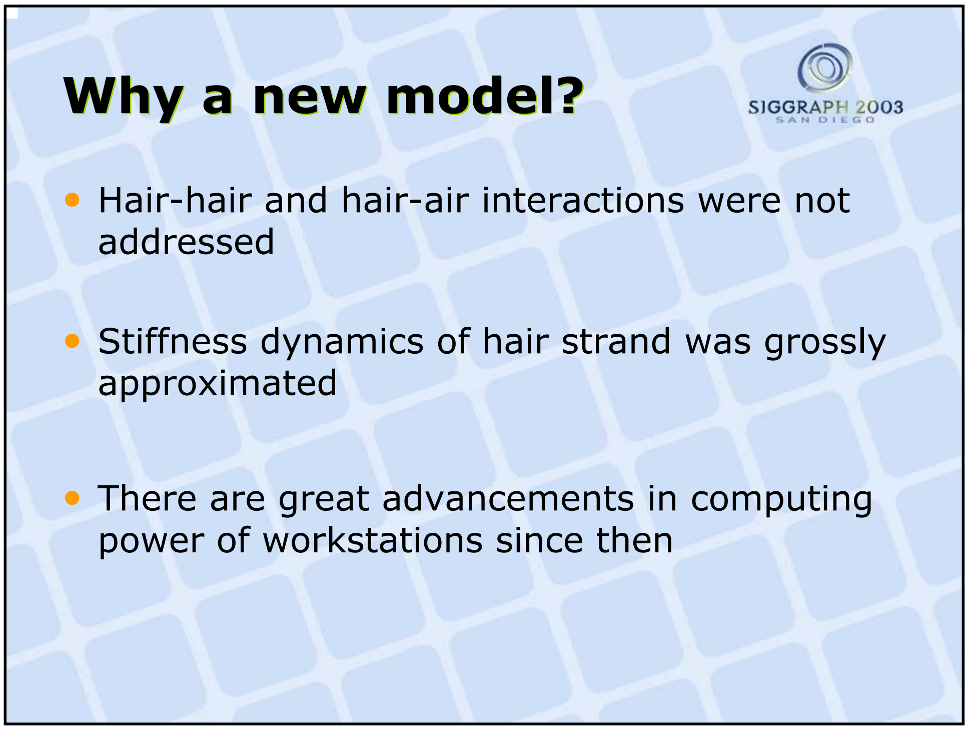
	Hair Modeling	Hair Animation	Hair Rendering
Explicit Models	effective *- tedious to model* *- not suitable for knots and braids*	adequate *- expensive due to size* *- inappropriate for hair-hair interaction*	fast *- inadequate for self-shadowing*
Particle Systems	inappropriate	adhoc *- lacks physical basis* *- no hair-hair interaction*	effective *- lacks shadowing and self-shadowing*
Volumetric Textures	effective *- not suitable for long hair*	limited *- via Animated Shape Perturbation*	effective *- expensive*
Cluster Model	effective *- not suitable for simple smooth hair*	not done *- via Animated Shape Perturbation*	effective

Previous Attempts

SIGGRAPH 2003
SAN DIEGO

Mainly explicit hair models or wisp models as they are intuitive and close to reality

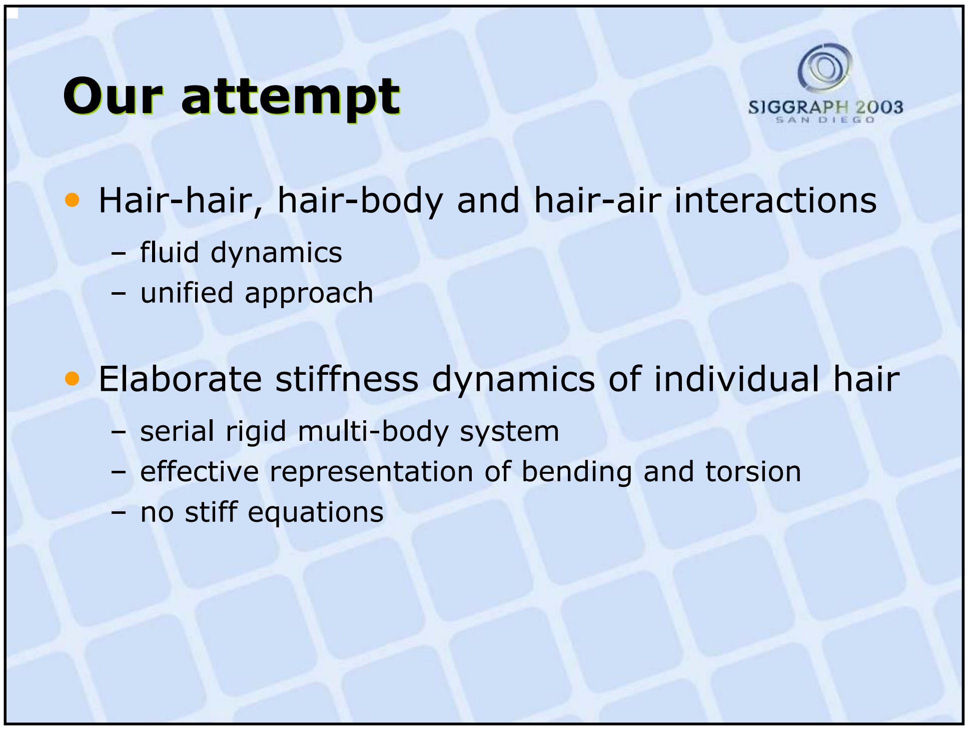
- Mass-spring-hinge models
  [Daldegan *et al*, Rosenblum *et al*]
- Simplified cantilever beam, one dimensional projective equation
  [Anjyo *et al*, Lee *et al*]
- Wisp models
  [Daldegan *et al,* Plante *et al*]



Why a new model?

SIGGRAPH 2003 SAN DIEGO

- Hair-hair and hair-air interactions were not addressed
- Stiffness dynamics of hair strand was grossly approximated
- There are great advancements in computing power of workstations since then



Our attempt

SIGGRAPH 2003 SAN DIEGO

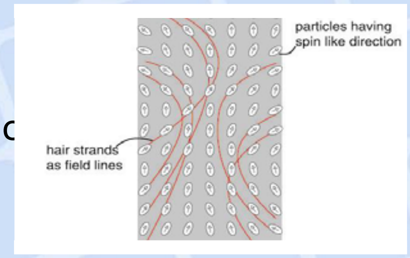
- Hair-hair, hair-body and hair-air interactions
  - fluid dynamics
  - unified approach
- Elaborate stiffness dynamics of individual hair
  - serial rigid multi-body system
  - effective representation of bending and torsion
  - no stiff equations

Hair Medium - Solid Liquid Duality

SIGGRAPH 2003 SAN DIEGO

- Solids have shape memory
- Liquids continue to deform under shearing stresses
- Hair exhibit both the properties

- Initial thinking…
- Long polymer fluid dynamic
- Cellular automata


particles having spin like direction
hair strands as field lines

Hair Medium - Solid Liquid Duality

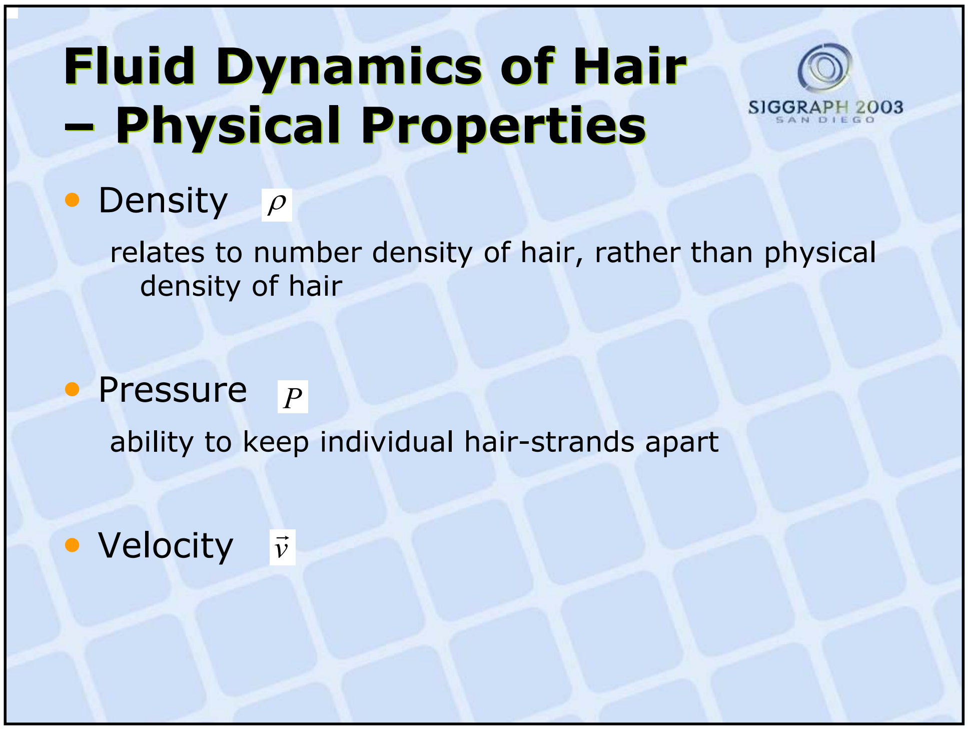
SIGGRAPH 2003 SAN DIEGO

We propose to …

Model hair-hair, hair-body, hair-air interactions as a continuum, more precisely fluid dynamics

Retain geometry of individual hair strand and its stiffness, inertial dynamics

Notion of "hair medium"



Fluid Dynamics of Hair – Physical Properties

SIGGRAPH 2003 SAN DIEGO

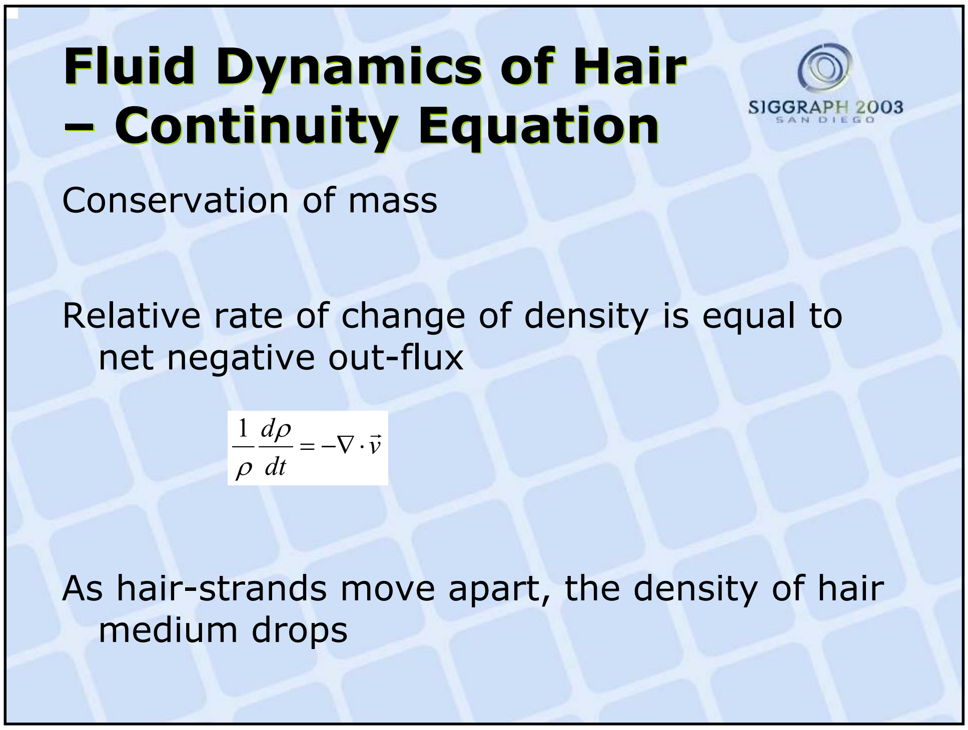
- Density ρ

  relates to number density of hair, rather than physical density of hair

- Pressure P

  ability to keep individual hair-strands apart

- Velocity $\vec{v}$



Fluid Dynamics of Hair – Continuity Equation

SIGGRAPH 2003 SAN DIEGO

Conservation of mass

Relative rate of change of density is equal to net negative out-flux

$$\frac{1}{\rho}\frac{d\rho}{dt} = -\nabla \cdot \vec{v}$$

As hair-strands move apart, the density of hair medium drops

Fluid Dynamics of Hair – Momentum Equation

$$\rho \frac{d\vec{v}}{dt} = \upsilon \nabla \cdot (\nabla \vec{v}) - \nabla p + F_{bd}$$

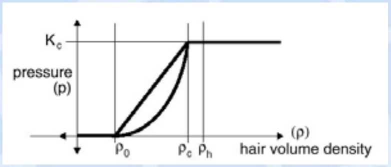
Acceleration $\frac{d\vec{v}}{dt}$ will be such that
it will even out the pressure variation $-\nabla p$

and there will always be a resistance to
the motion $\upsilon \nabla \cdot (\nabla \vec{v})$
in the form of viscosity

Fluid Dynamics of Hair – Equation of State

- "Design" EOS to suit our needs

$$p = \begin{cases} 0 & \text{if } \rho < \rho_0, \\ K_c(\frac{\rho-\rho_0}{\rho_c-\rho_0})^n & \text{if } \rho_0 \le \rho < \rho_c, \\ K_c & \text{if } \rho_c < \rho \end{cases}$$



- If hair volume is squeezed, it gives rise to pressure to keep individual hair apart
- Choice of K_c
- Alignment of hair defines power n

Fluid Dynamics of Hair – Hair-body Interactions

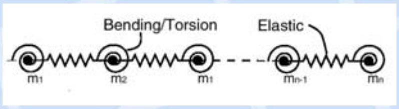
SIGGRAPH 2003 SAN DIEGO

- No need to model collisions of individual hair strands with the body
- Model it in a unified way as flow boundary condition
  - Flow tangency condition – fluid flow normal to the boundary is zero
  - Flow slip condition – boundary exerts a viscous pressure

Single Hair – Stiffness and Inertial Dynamics

SIGGRAPH 2003 SAN DIEGO

- We need to retain individual character of hair



- Set of point masses connected by springs
- n particles, 6n DOFs
- Stiff equations, redundant DOFs
- Inaccurate inertial dynamics

Single Hair Dynamics – avoid stiff equations

SIGGRAPH 2003 SAN DIEGO

- Consider thin cylinders instead of point masses
- Remove redundant 3n DOFs
- Constrained Dynamics

  Express the system 6n DOFs and constrain the 3n DOFs
- Reduced Coordinate Formulation

  Express the system in exact number of 3n DOFs, i.e. 2n bending DOFs and n torsional DOFs

Single Hair Dynamics – Rigid Multibody Serial Chain

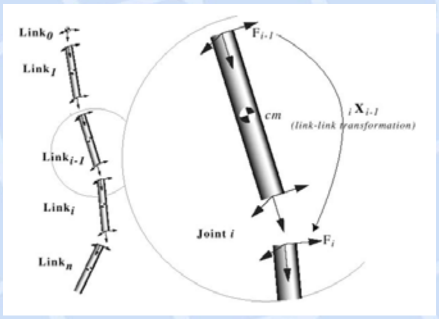
SIGGRAPH 2003 SAN DIEGO

Reduced Coordinate Formulation is preferred

- Finally, system has fewer (3n) DOFs
- Simple topology
- Reduced coordinates directly facilitates parametric definition of bending and torsion

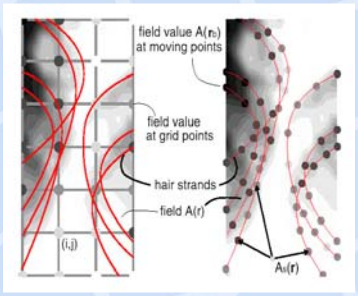
Hair as Serial Rigid Multi-body Chain

- n segments of equal lengths, single un-branched open-loop chain
- Each link is connected to other by 3 DOF Spherical Joint
- Stiffness dynamics is introduced by joint actuation torques
- Inertial dynamics is computed using articulated rigid-body algorithm



Kinematic link between the two models

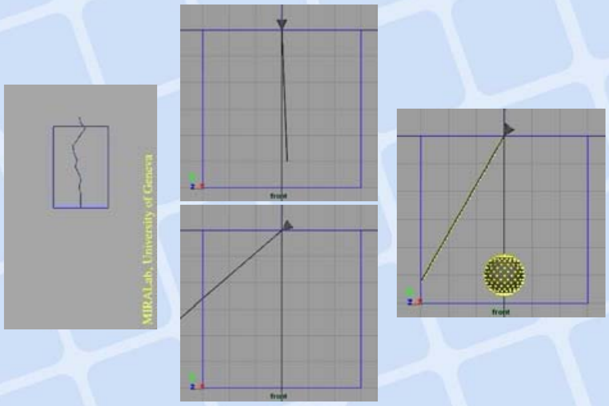
- Imagine hair strands immerged in a thick fluid. The hair-hair interaction is captured by fluid dynamics.The fluid is allowed to move freely, it is rather glued to hair strands by a kinematic link.
- Eulerian vs Langrangian viewpoints
- We use Smoothed Particle Hydrodynamics
- We did not succeed using Lattice-Boltzmann equation



field value $A(r_b)$ at moving points
field value at grid points
hair strands
field $A(r)$
(i,j)
$A_b(r)$

Hair Strand Inertial and Stiffness Dynamics

SIGGRAPH 2003 SAN DIEGO



Results

SIGGRAPH 2003 SAN DIEGO

MIRALab, University of Geneva

Results

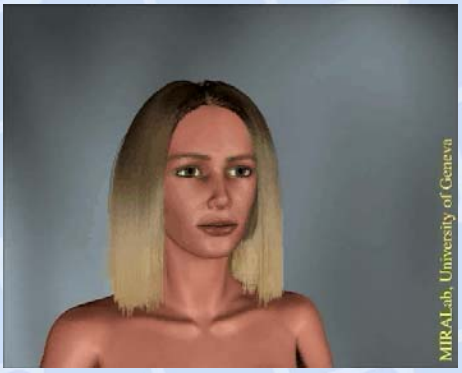
SIGGRAPH 2003
SAN DIEGO

Results

SIGGRAPH 2003
SAN DIEGO

Results

SIGGRAPH 2003
SAN DIEGO

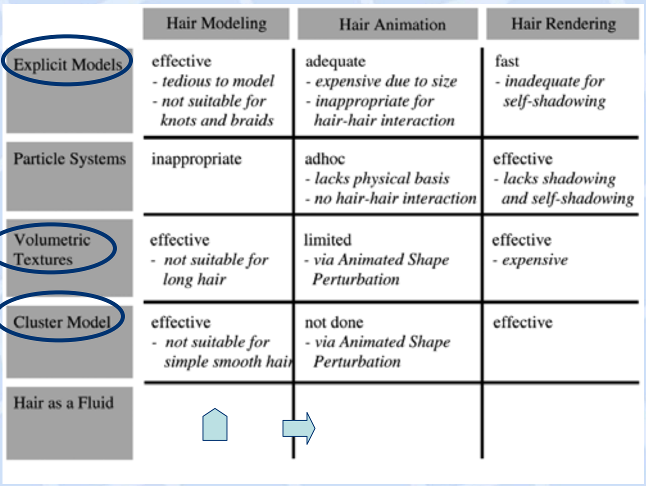


Specific Contributions and Strengths

SIGGRAPH 2003
SAN DIEGO

Tasks

Models

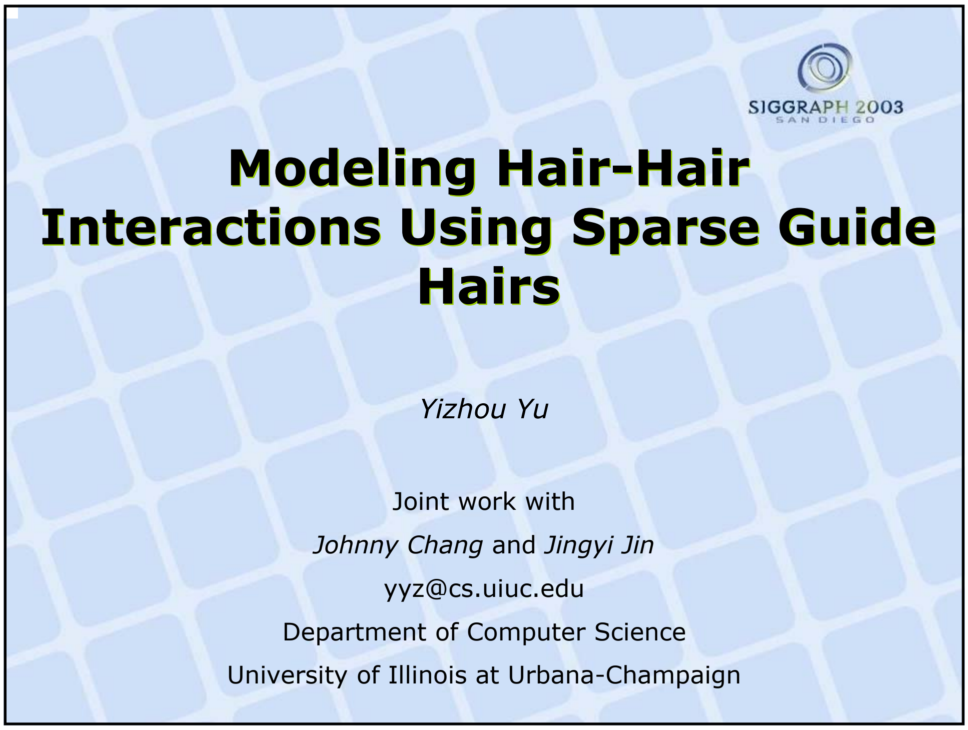


	Hair Modeling	Hair Animation	Hair Rendering
Explicit Models	effective - *tedious to model* - *not suitable for knots and braids*	adequate - *expensive due to size* - *inappropriate for hair-hair interaction*	fast - *inadequate for self-shadowing*
Particle Systems	inappropriate	adhoc - *lacks physical basis* - *no hair-hair interaction*	effective - *lacks shadowing and self-shadowing*
Volumetric Textures	effective - *not suitable for long hair*	limited - *via Animated Shape Perturbation*	effective - *expensive*
Cluster Model	effective - *not suitable for simple smooth hair*	not done - *via Animated Shape Perturbation*	effective
Hair as a Fluid			

Limitations and Future Work

- Impulse dynamics
- Would want to explore mass-spring-hinge system using implicit numerical integration
- Formation hair clumps using condensation of hair medium

hairdance



Modeling Hair-Hair Interactions Using Sparse Guide Hairs

Yizhou Yu

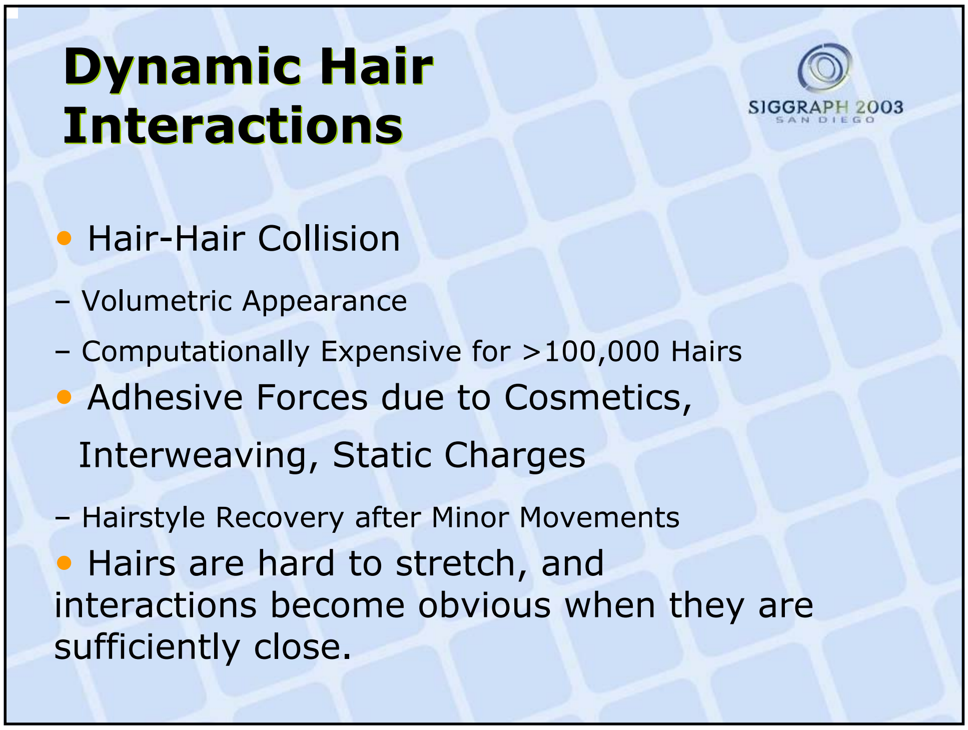
Joint work with

Johnny Chang and *Jingyi Jin*

yyz@cs.uiuc.edu

Department of Computer Science

University of Illinois at Urbana-Champaign



Dynamic Hair Interactions

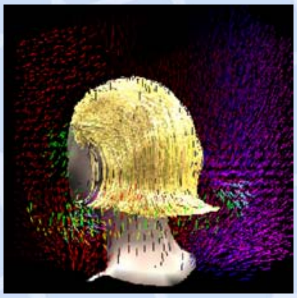
- Hair-Hair Collision
  - Volumetric Appearance
  - Computationally Expensive for >100,000 Hairs
- Adhesive Forces due to Cosmetics, Interweaving, Static Charges
  - Hairstyle Recovery after Minor Movements
- Hairs are hard to stretch, and interactions become obvious when they are sufficiently close.

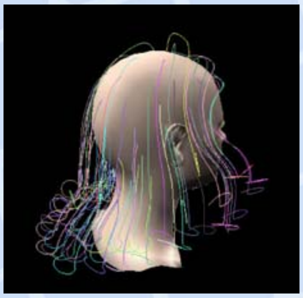
Hair Simulation Using Sparse Guide Hairs

- Simulating Sparse Guide Hairs
  - Single strand dynamics for each guide hair
  - Simulating adhesive forces using static links
  - Simulating hair-hair collisions using density modulate triangle strips
- Dense Hair Simulation by Interpolation
  - Hair interpolation happens at each frame.
  - Fixed correspondences between dense hairs and guide hairs to achieve temporal coherence
  - Hair-object collisions are handled after interpolation for each individual strand.Simulating Sparse Guide H

Guide Hair Modeling

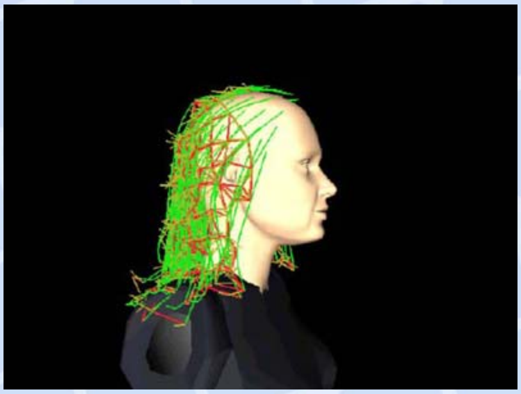
- Modeling hair flows with vector fields





Video: Guide Hairs

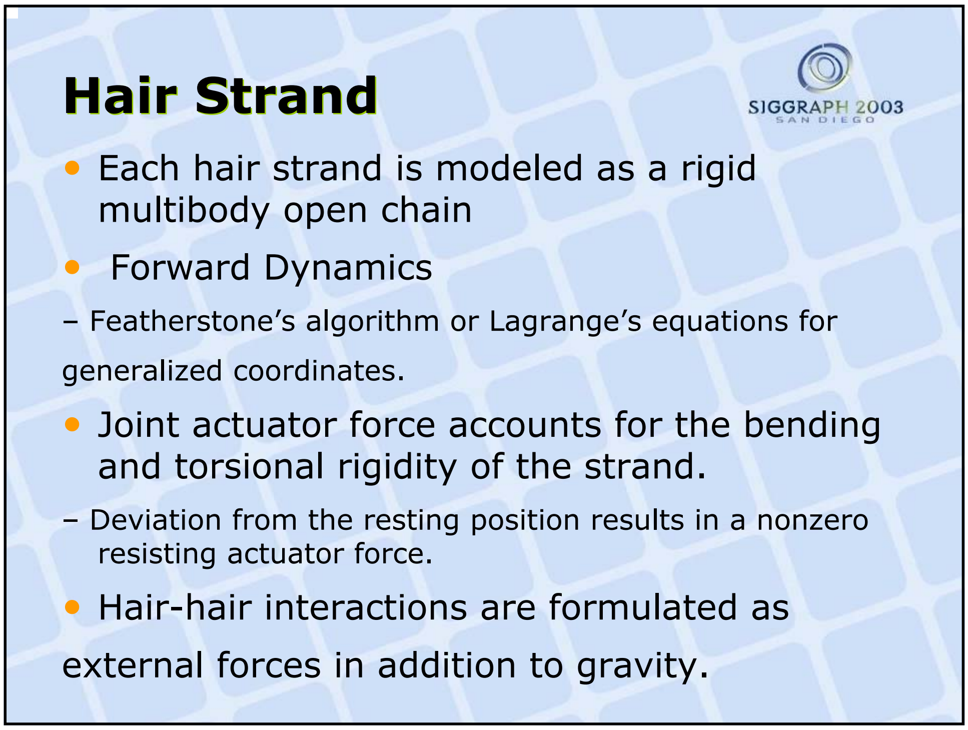
SIGGRAPH 2003 SAN DIEGO



Related Work on Hair Animation

SIGGRAPH 2003 SAN DIEGO

- Single Strand Dynamics
  - Mass-Spring-Hinge Model
    [Rosenblum et al. 91], [Daldegan et. al. 93]
  - Cantilever Beam
    [Anjyo et al. 92]
  - Multi-body Open Chain
    [Hadap & Thalman 01]
- Hair-Hair Interactions
  - Fluid-based Model
    [Hadap & Thalman 01]
  - Wisp-based Model
    [Plante et al. 01]



Hair Strand

SIGGRAPH 2003 SAN DIEGO

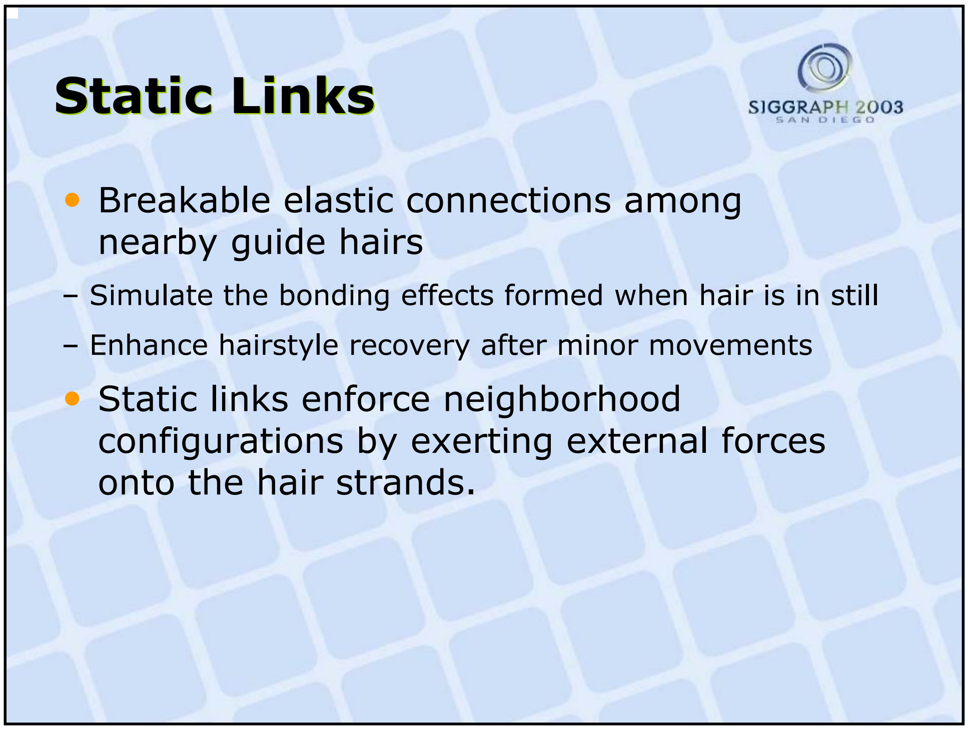
- Each hair strand is modeled as a rigid multibody open chain
- Forward Dynamics

– Featherstone's algorithm or Lagrange's equations for generalized coordinates.

- Joint actuator force accounts for the bending and torsional rigidity of the strand.

– Deviation from the resting position results in a nonzero resisting actuator force.

- Hair-hair interactions are formulated as external forces in addition to gravity.



Static Links

SIGGRAPH 2003 SAN DIEGO

- Breakable elastic connections among nearby guide hairs

– Simulate the bonding effects formed when hair is in still

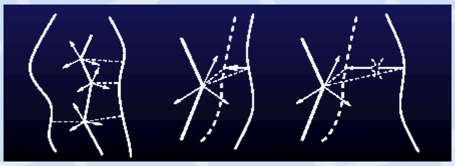
– Enhance hairstyle recovery after minor movements

- Static links enforce neighborhood configurations by exerting external forces onto the hair strands.

Static Links as Positional Springs

SIGGRAPH 2003 SAN DIEGO

- Introduce a local coordinate system to each segment of the hair strands.
- Transform points on the nearby strands to the segment's local system and keep them as the reference points.
- Forces are generated to recover the original relative positions of these reference points.



Force from Static Links

SIGGRAPH 2003 SAN DIEGO

- The accumulated force a segment receives due to static links can be formulated as

$$\mathbf{f}_h = \sum_i \left[k^s_{h,i} |\mathbf{l}_i| - k^d \frac{\mathbf{v}_i \cdot \mathbf{l}_i}{|\mathbf{l}_i|} \right] \frac{\mathbf{l}_i}{|\mathbf{l}_i|}$$

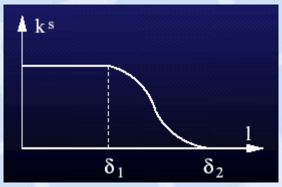
– *ks* is the spring constant *kd* is the damping constant, *v* is the time derivative of l, and

$$|\mathbf{l}_i = p^n_{h,i} - p^o_{h,i}|$$

- A static link can be broken

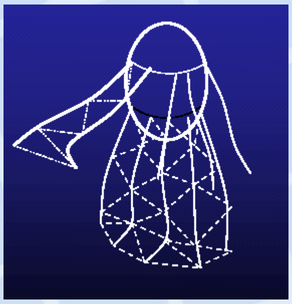
when its length change

exceeds a threshold.



Dynamic Interactions

- Use of auxiliary triangle strips to imagine the space in between the set of sparse guide hair
- Collisions between the hair segments and the triangle strips are explicitly considered



Modeling Hair Density

- Every face on a triangle strip is associated with a density value which can be zero.
  - The length of the triangle edges serves as the indicator for the hair density on a strip.
  - If a triangle becomes too elongated, its density is labeled as zero.
- Hair strands are allowed to go through sparse or broken pieces of a triangle strip more easily.

Modeling Collision Forces

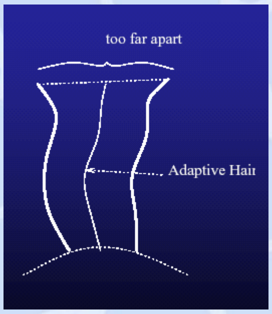
- Depending on the orientation of the penetrating hair vertex and the triangular face, the repelling spring force might vary

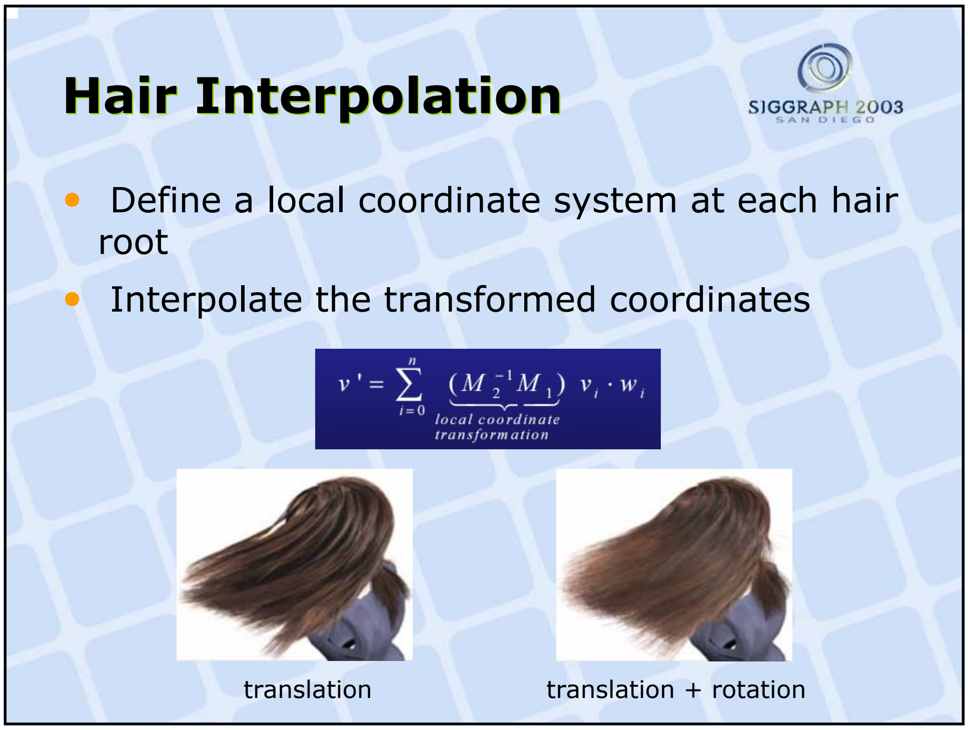
$$\mathbf{f}_r = \lambda(1 - |\mathbf{a} \cdot \mathbf{b}|)\mathbf{f}_s$$

- Where a is the normalized tangential vector of the hair at the penetrating vertex, b is the interpolated hair orienation on the triangular face, λ is the scale factor
- The scale factor λ is adjusted according to the hair density.

Adaptive Hair Generation

- Generate additional guide strands adaptively on the fly to cover the over interpolated regions
  – Compare the distance between two guide strands. If the distance is too far, an
- Inserted guide hairs can also be removed during the simulation

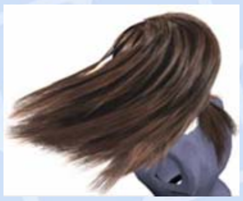


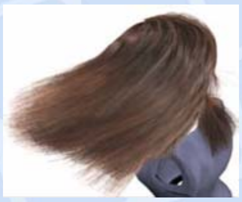


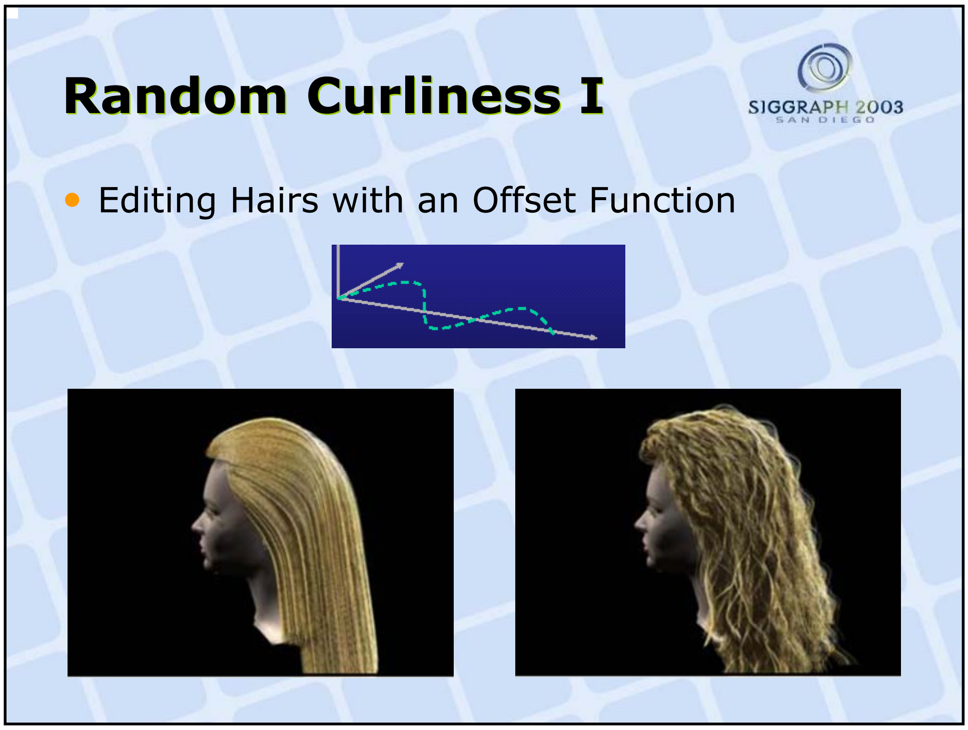
Hair Interpolation

- Define a local coordinate system at each hair root
- Interpolate the transformed coordinates

$$v' = \sum_{i=0}^{n} \underbrace{(M_2^{-1} M_1)}_{\substack{\text{local coordinate} \\ \text{transformation}}} v_i \cdot w_i$$

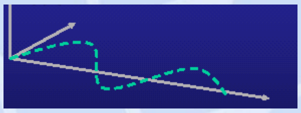

translation

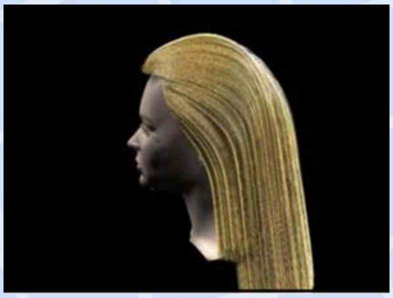

translation + rotation

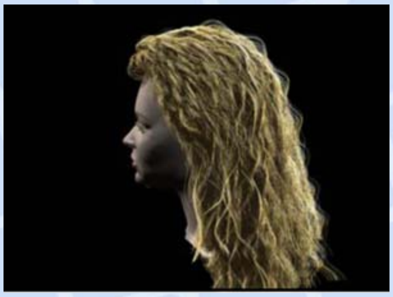


Random Curliness I

- Editing Hairs with an Offset Function

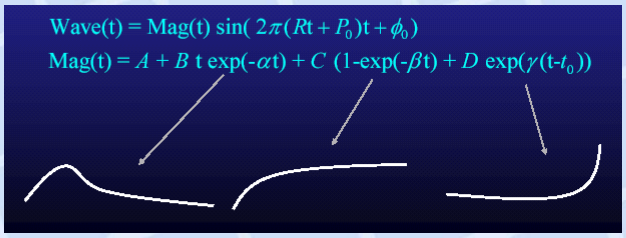






Random Curliness II

- Parametric Offset Function
  - –Variable magnitude + variable period



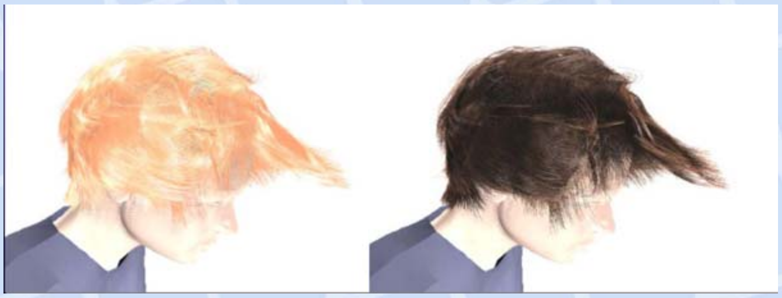
$$\mathrm{Wave}(t) = \mathrm{Mag}(t)\sin(2\pi(Rt + P_0)t + \phi_0)$$

$$\mathrm{Mag}(t) = A + B\,t\exp(-\alpha t) + C\,(1-\exp(-\beta t) + D\exp(\gamma(t-t_0))$$

Examples of Curly /Wavy Hair Models

Hair Rendering

Kajiya-Kay Illumination Model + Adjustable Translucency



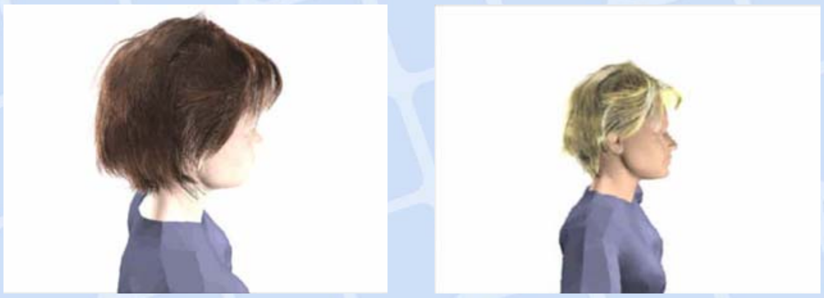
Video: Braided Hair

Video: Long Hair

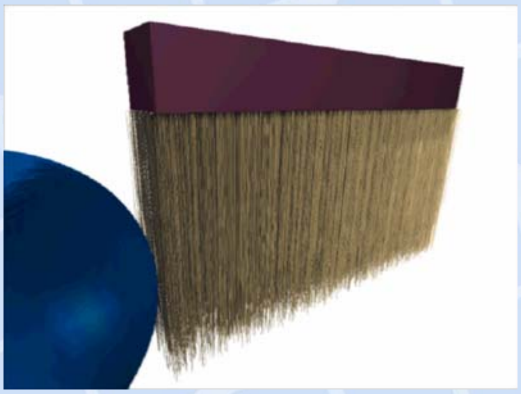
SIGGRAPH 2003
SAN DIEGO

Video: Long Hair

SIGGRAPH 2003
SAN DIEGO



Video: Brush



Conclusions

- Hair mutual interactions are indispensable for

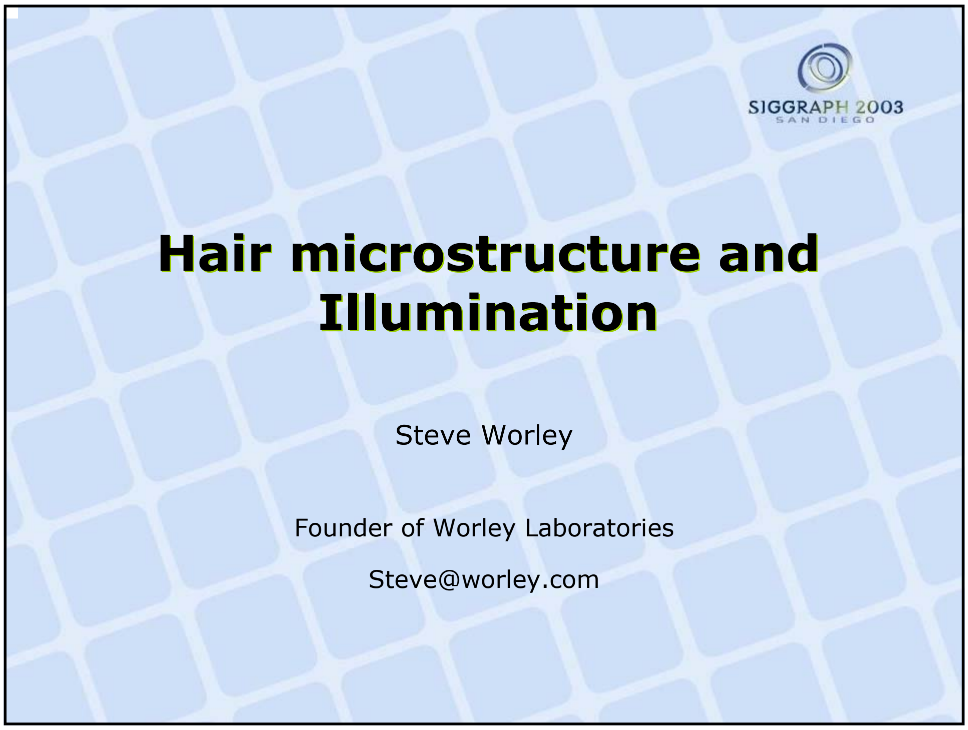
realistic hair simulations.

- We use sparse guide hairs to produce hair

motion, and densely interpolated hairs for the final appearance.

- We propose to use static links to simulate

adhesive forces and enhance hairstyle recovery, and density modulated triangle strips for hair-hair collisions.

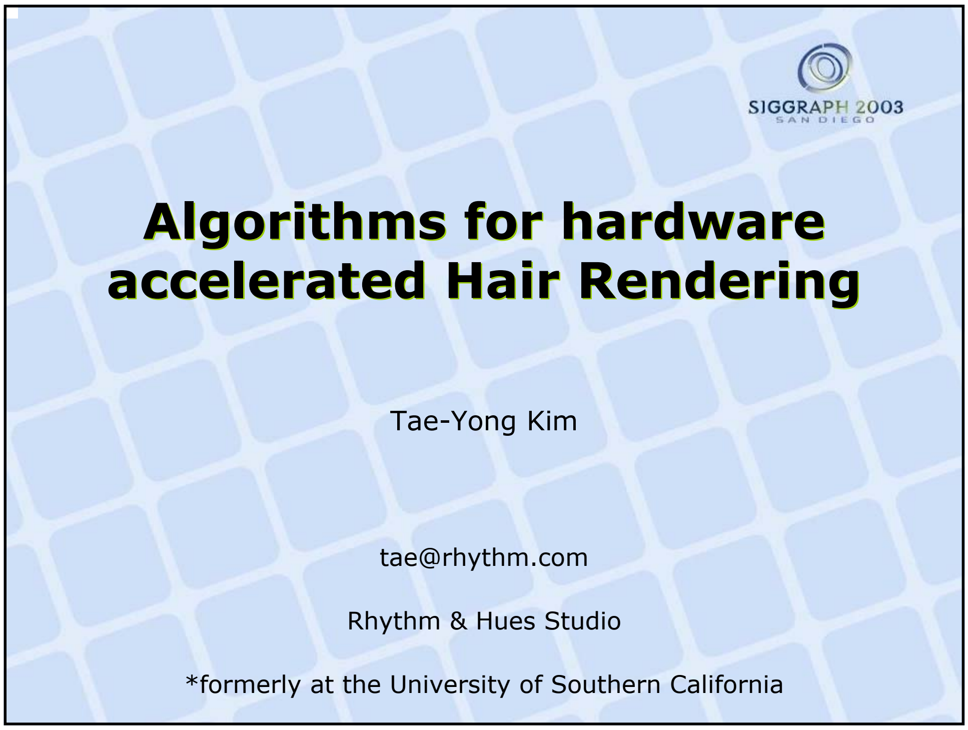


Hair microstructure and Illumination

Steve Worley

Founder of Worley Laboratories

Steve@worley.com



Algorithms for hardware accelerated Hair Rendering

Tae-Yong Kim

tae@rhythm.com

Rhythm & Hues Studio

*formerly at the University of Southern California

Cultural Heritage applications at MIRALab

Prof. Nadia Magnenat-Thalmann

Thalmann@miralab.unige.ch

MIRALab – University of Geneva

Augmented life in Pompei

Lifeplus project(2002-2004)

- LIFEPLUS proposes new Augmented Reality narrative spaces for the innovative revival of life in ancient frescos-paintings in Pompeii

Objective

SIGGRAPH 2003 SAN DIEGO

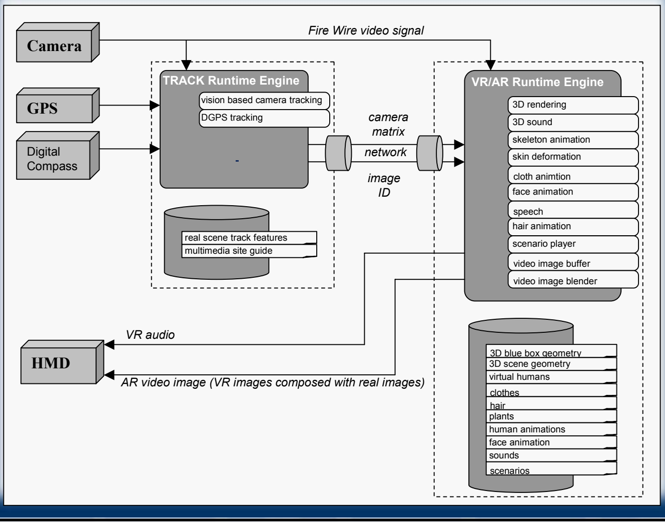
- The revival is based on real scenes captured on live video sequences augmented with real-time autonomous groups of 3D virtual fauna and flora

- a mobile AR on-site guide based on immersive wearable computing

Xybernaut™ Poma Mobile Assistant

System Architecture and Content Database

SIGGRAPH 2003 SAN DIEGO



Research in Clothes and Facial Animation

SIGGRAPH 2003 SAN DIEGO

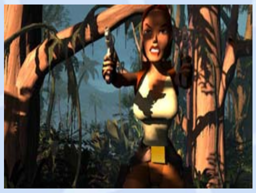
- Real-time Cloth simulation
  - Hybrid deformation
- Real-time Facial emotion expression and Speech animation

Research in Virtual Hair and Flora Simulation

SIGGRAPH 2003 SAN DIEGO

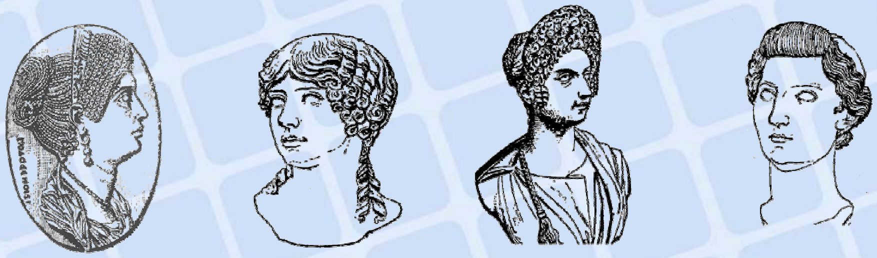
- Real-time Hair simulation
- Real-time plant simulation

Copyright @ MIRALab, University of Geneva



The Making of Ancient Hair Styles

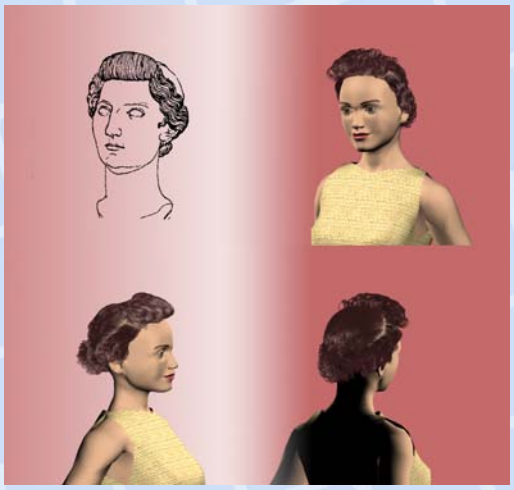
- References



From Antonio d'Ambrosio, WOMEN AND BEAUTY IN POMPEII, « L'Erma »

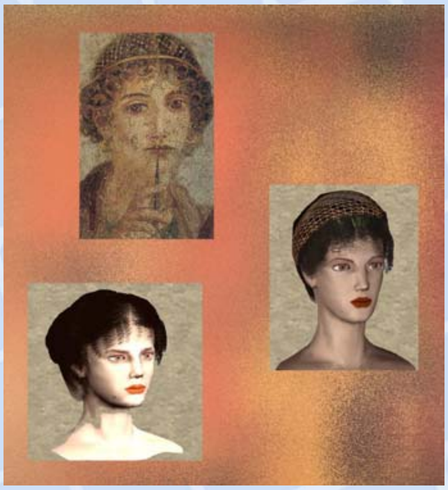
The Making of Ancient Hair Styles

- From Pictures (drawing) to 3D model



The Making of Ancient Hair Styles

- From Mural to 3D models

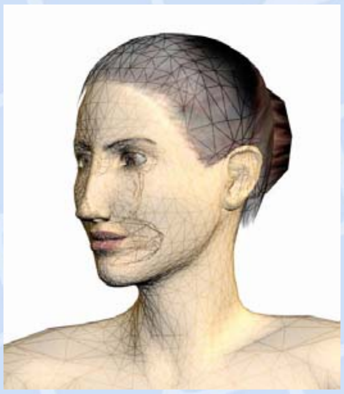


Salvatore Nappo, POMPÉI, Gründ

The Making of Ancient Hair Styles

- From Concept to 3D model

Ascla
J. V. Seeg



Character from LifePlus project

On-site studies

Reconstruction

Work in progress

Research in Humans Simulation

SIGGRAPH 2003 SAN DIEGO

- Real-time realistic skin rendering and interactive programmable shading module

- Artificial life methods for behavioural animation of virtual characters

movie

Humans in reconstructed environment

SIGGRAPH 2003 SAN DIEGO

Augmented Reality in ancient Pompeii

SIGGRAPH 2003 SAN DIEGO

movie

Early Mixed Realities simulations

SIGGRAPH 2003 SAN DIEGO

Production Hair/Fur Pipeline at Imageworks

Armin Bruderlin

Sony Pictures Imageworks

armin@imageworks.com

State of the Art in Hair Simulation

Nadia Magnenat-Thalmann Sunil Hadap Prem Kalra [*]

MIRALab, CUI, University of Geneva
24, rue du General Dufour, CH-1211 Geneva, Switzerland
email: {thalmann,sunil}@cui.unige.ch

Abstract

In this paper we summarize the technological advances in hair simulation for computer graphics. There are mainly three tasks in hair simulation - Hair Shape Modeling, Hair Dynamics and Hair Rendering. Various models developed for these tasks, fall mainly in the categories of *particle systems*, *explicit hair models*, *cluster hair models* and models based on *volumetric textures*. We discuss advantages and disadvantages of each of these approaches. We also introduce a new hair shape modeling paradigm based on fluid flow. The proposed method provides a sound basis for modeling hair-body and hair-hair interaction.

Keywords: hair shape modeling, hair animation, hair rendering, hypertexture

1 Introduction

One of the many challenges in simulating believable virtual humans has been to produce realistic looking hair. The virtual humans, two decades ago, were given polygonal hair structure. Today, this is not acceptable. Realistic visual depiction of virtual humans has improved over the years. Attention has been given to all the details necessary for producing visually convincing virtual humans and many improvements have been done to this effect.

On a scalp, human hair are typically 100,000 to 150,000 in number. Geometrically they are long thin curved cylinders having varying thickness. The strands of hair can have any degree of waviness from straight to curly. The hair color can change from white to grey, red to brown, due to the pigmentation, and have shininess. Thus, difficulties of simulating hair stem from the huge number and geometric intricacies of individual hair, complex interaction of light and shadow among the hairs, the small scale of thickness of one hair compared to the rendered image and intriguing hair to hair interaction while in motion. One can conceive three main aspects in hair simulation - hair shape modeling, hair dynamics or animation, and hair rendering. Often these aspects are interconnected while processing hairs. Hair shape modeling deals with exact or fake creation of thousands of individual hair - their geometry, density, distribution, and orientation. Dynamics of hair addresses hair movement, their collision with other objects particularly relevant for long hair, and self-collision of hair. The rendering of hair involves dealing with hair color, shadow, specular highlights, varying degree of transparency and anti-aliasing. Each of the aspects is a topic of research.

Many research efforts have been done in hair simulation research, some dealing only with one of the aspects of simulation -shape modeling, dynamics or rendering. Several research efforts were inspired by the general problem of simulation of natural phenomena such as grass, and trees. These addressed a more limited problem of simulating of fur or short hair. We divide hair simulation models into four categories depending upon the underlying technique involved: *particle systems, volumetric textures, explicit hair models and cluster hair model*. We discuss models presented by researchers in each of these model categories and state their contribution to the three aspects of hair simulation, i.e. hair shape modeling, hair dynamics and hair rendering. We also introduce a new hair shape modeling paradigm based on fluid flow.

The paper is organized as follows. First we give the state of the art in hair shape modeling. The hair shape modeling research in each category of the simulation models is presented. Models for hair dynamics are briefly described in Section 3. Section 4 presents the problem of hair rendering and the various solutions proposed by different people. Finally, we summarize the effectiveness and limitations of models in the four categories related to each aspect of hair simulation in the form of a table. Some future avenues for research in hair simulation are also outlined.

2 Hair Shape Modeling

Intricate hairstyle is indeed a consequence of physical properties of an individual hair and complex hair-hair and hair-body interactions. As we will see in the next section, modeling complex hair dynamics, that too at interactive speeds, is currently impractical. For the reasons, it would be worthwhile to treat *hair shape modeling* as a separate problem and use some heuristic approach.

Early attempts of styling long hair were based on *explicit hair models*. In the explicit hair model, each hair strand is considered for the shape and the dynamics. Daldegan

[*]Visiting from Department of Computer Science and Engineering, Indian Institute of Technology, Delhi, India. pkalra@cse.iitd.ernet.in

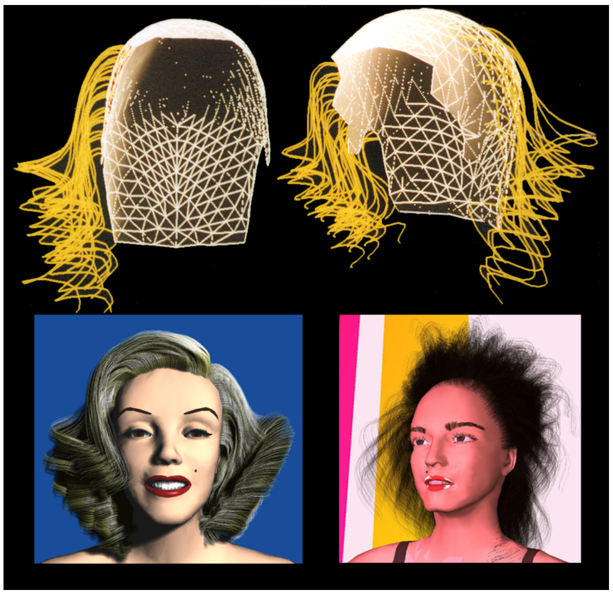


Figure 1: Hairstyling by defining a few curves in 3D

et al [5] proposed that the user could interactively define a few characteristic hair strands in 3D and then populate the hair style based on them. The user is provided with a flexible graphical user interface to sketch a curve in 3D around the scalp. A few parameters such as density, spread, jitter and orientation control the process that duplicates the characteristic hairs to form a hair style. Figure 1 illustrates the method of defining few characteristic curves and resulting hairstyles from the method. Similarly, even for the fur modeling, Daldegan *et al* [4], Gelder *et al* [8] and Bruderlin *et al* [1] took similar explicit hair modeling approach. Figure 12 illustrates a furry coat modeled by the explicit hair model.

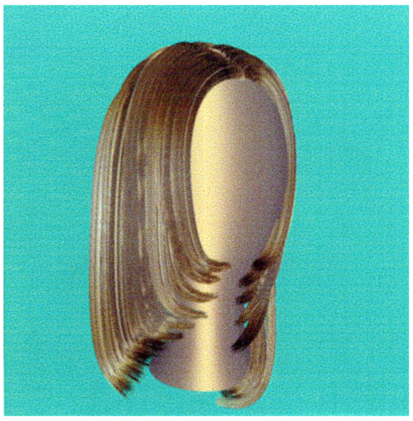


Figure 2: Cluster Hair Model, by Yan *et al*

The explicit hair models are very intuitive and close to reality. Unfortunately, they are tedious for hairstyling. Typically, it takes 5-10 hours to model a complex hair style, as in figure 1, using the method in [5]. They are also numerically expensive for hair dynamics. These difficulties are partially overcome by considering a bunch of hair instead of individual hair in the case of the wisp/cluster models. This assumption is quite valid as in reality. Due to effects of adhesive/cohesive forces, hairs tend to form clumps. Watanabe introduced the wisp modeled in [24, 25]. Yan *et al* [26] modeled the wisps as *generalized cylinders*, see figure 2. One of the contributions of the work was also in rendering of hair using the blend of ray-tracing generalised cylinders and the *volumetric textures*. The wisp model is also evident in [2]. Surprisingly, till now, the wisp models are only limited to static hair shape modeling and we feel that it offers an interesting research possibility of modeling hair dynamics, efficiently. It would be interesting to model, how hair leave one wisp and join the other under dynamics.

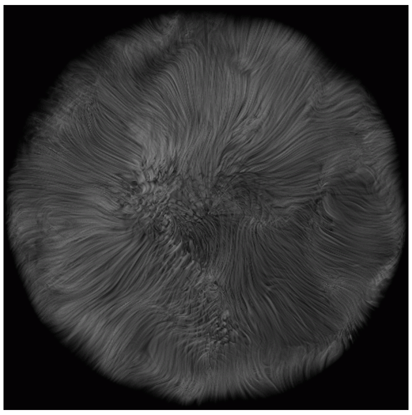


Figure 3: Fur as a Volumetric Texture, by Perlin *et al*

Nature exhibits some interesting fuzzy objects such as clouds, fire, eroded rocks and fur for which it is hard to have explicit geometric definition. Using the volumetric texture approach, fur can be modeled as a volumetric density function. Perlin *et al* [18] introduced *hypertextures*, which can model fur, see figure 3. Here, fur is modeled as intricate density variations in a 3D space, which gives an illusion of the fur like medium without defining geometry of each and every fiber. The model is essentially an extension to procedural solid texture synthesis evaluated through out the region, instead of only in the surface. They demonstrated that, combinations of simple analytical functions could define furry ball or furry donut. They further used 3D vector valued noise and turbulence to perturb the 3D texture space. This gave the natural looks to the otherwise even fur defined by the hypertexture. A good discussion on the procedural approach to modeling volumetric texture and fur in particular is in [7]. Hypertexture method by Perlin *et al* is only limited to geometries that can be analytically defined. Kajiya *et al* [12] extended this approach to have hypertextures tiled on to complex geometry. They demonstrated this by modeling a furry bear, see figure 10. They used a single solid texture tile namely texel and mapped it repeatedly on the bear's geometry. The texels automatically orient in the direction away from the surface and thus one has fuzzy volumetric density variation around the bear, which is the fur.

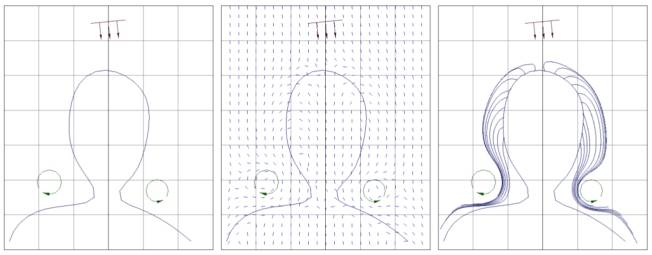


Figure 4: Hair as streamlines of a fluid flow

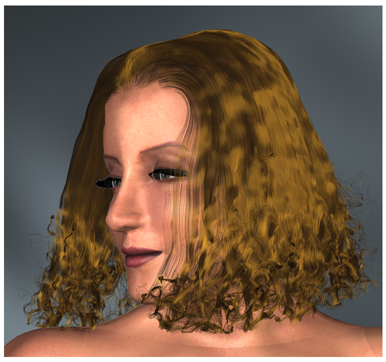


Figure 6: Adding overall volumetric perturbations to the fluid flow

As evident from previous discussions, one of the strengths of the explicit hair models is their intuitiveness and ability to control the global shape of the hair. On the contrary, volumetric textures give a nice way of interpreting complexity in nature and they are rich in details. We notice that the fluid flow has both the characteristics, which we would like to exploit for hair shape modeling. We model hair shape as streamlines of a fluid flow. For complete details of the method, we refer to [10]. We choose the flow to be an ideal flow. User can setup few ideal flow elements around the body geometry to design a hairstyle, as shown in figure 2. The hair-body interaction is modeled using *source panel method* and hair-hair interaction is handled by the continuum property of fluid. Thus user can design complex hairstyles without worrying about hair-body and hair-hair interaction. Hairstyles in figure 5 and 6 are the examples of modeling hair as a fluid flow.

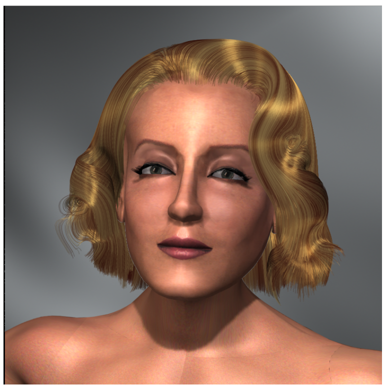


Figure 5: Hair as a fluid flow

3 Hair Dynamics

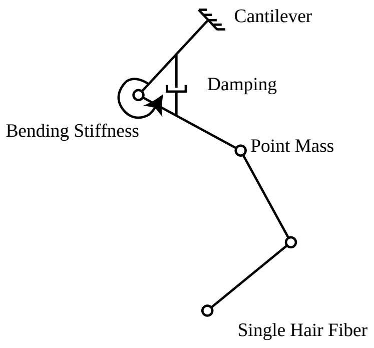


Figure 7: Simple mass-spring system for an individual hair dynamics

Anjyo *et al* [11], Rosenblum *et al* [22] and Kurihara *et al* [23] developed dynamic models that are issentialy based on individual hair. An individual hair is modeled as connected rigid segments having bending stiffness at each joint. Then the individual hair is solved for the movement due to the inertial forces and the collision with the body. Though the cantilever dynamics and collision avoidance with the body of each hair is within the scope of current computing power, modeling complex hair-to-hair interaction is still a challenge. Figure 8 illustrates the effectiveness of the dynamic model even though no hair-hair interaction is considered.

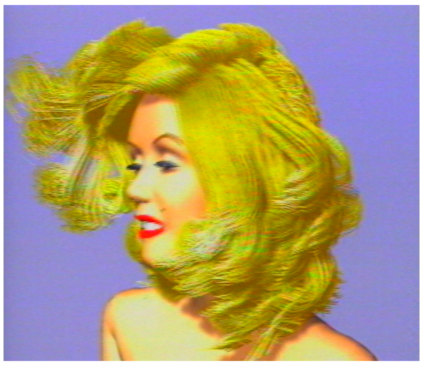


Figure 8: Hair animation using the explicit model, by Kurihara *et al*

In the case of fur, which is mostly modeled as volumetric texture, one cannot take the explicit model approach for the animation. In this case, a time varying volume density function can facilitate animation of fur. One can simulate effects of turbulent air on the fur using stochastic space perturbation such as turbulence, noise, Brownian motion etc. Apart from Lewis [15] and Perlin [17, 18], work by Dischler [6] gave a generalized method for these animated shape perturbations.

4 Hair Rendering

In the field of virtual humans, hair presents one of the most challenging rendering problems. The difficulties arise from various reasons: large number of hair, detailed geometry of individual hair and complex interaction of light and shadow among the hairs and their small thickness. The rendering of hair often suffers from the aliasing problem due to many individual hairs reflecting light and casting shadows on each other contribute to the shading of each pixel. Further, concerning display of hairs, we see not only individual hairs but also a continuous image consisting of regions of hair color, shadow, specular highlights, varying degree of transparency and haloing under backlight conditions. The image, in spite of the structural complexity, shows a definite pattern and texture in its aggregate form.

In the last decade, the hair-rendering problem has been addressed by a number of researchers, in some cases with considerable success. However, most cases work well in particular conditions and offer limited (or none) capabilities in terms of dynamics or animation of hair. Much of the work refers to a more limited problem of rendering fur, which also has a lot in common with rendering natural phenomena such as grass and trees. As follows we give the related work in hair rendering focusing their salient features and limitations.

Particle systems introduced by Reeves *et al* [19], primarily meant to model class of fuzzy objects such as fire. Despite particles small size -smaller than even a pixel- the particle manifests itself by the way it reflects light, casts shadows, and occludes objects. Thus, the subpixel structure of the particle needs to be represented only by a model that can

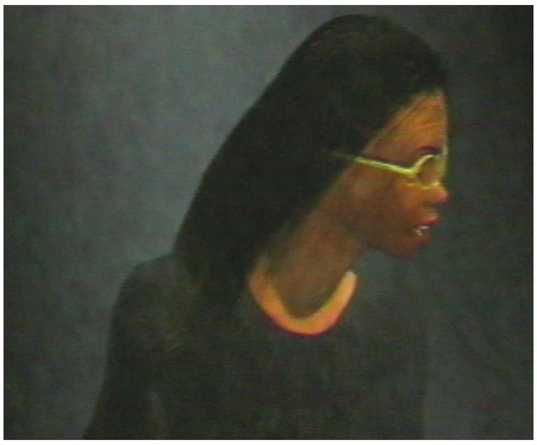


Figure 9: Hair as Connected Particle System, “The End” by Alias—Wavefront

represent these properties. A particle system is rendered by painting each particle in succession onto the frame buffer, computing its contribution to the pixel and compositing it to get the final color at the pixel. The technique has been successfully used for rendering these fuzzy objects and integrated in many commercial animation systems. Figure 9 is an example of how one can use connected particle systems for the modeling of hair. However, the technique has some limitations for shadowing and self-shadowing. Much of it is due to the inherent modeling using particle systems: simple stochastic models are not adequate to represent the type of order and orientation of hair. Also, it requires appropriate lighting model to capture and control the hair length and orientation. The specular highlights in particular owing to the geometry of the individual strands are highly anisotropic.

Impressive results have been obtained for the more limited problem of rendering fur, which can be considered as very short hair. As we have already discussed in the case of hair shape modeling, Perlin *et al* [18] introduced hypertextures that can model fur like objects. Hypertexture approach remains limited to geometries that can be defined analytically. Kajiya and Kay extended this approach to use it on the complex geometries. They used a single solid texture tile namely texel. The idea of texels was inspired by the notion of volume density used in [18]. A texel is a 3D texture map where both the surface frame and lighting model parameters are embedded over a volume. Texels are a type of model intermediate between a texture and a geometry. A texel is however, not tied to the geometry of any particular surface and thus makes the rendering time independent of the geometric complexity of the surface that it extracts. The results are demonstrated by rendering a teddy bear (figure 10). Texels are rendered using ray casting, in a manner similar to that for volume densities using a suitable illumination model. Kajiya *et al* discusses more about the particular fur illumination model and a general rendering method for rendering volume densities. The rendering of volume densities are also covered in great detail in the book by Eber *et al* [7].

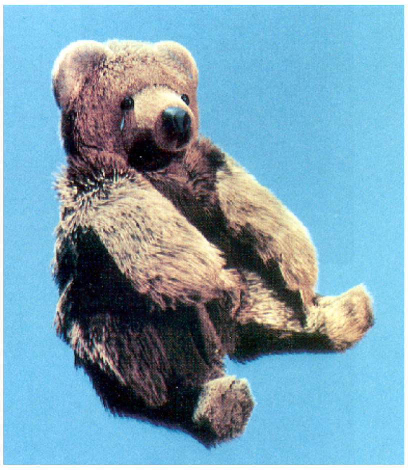


Figure 10: Volumetric Texture rendering by Kajiya *et al*

In another approach by Goldman [9], emphasis is given on rendering visual characteristics of fur in cases where the hair geometry is not visible at the final image resolution -object being far away from the camera. A probabilistic rendering algorithm, also referred to as fakefur algorithm is proposed. In this model, the reflected light from individual hairs and from the skin below is blended using the expectations of a ray striking a hair in that area as the opacity factor.

Though the volumetric textures are quite suitable for rendering furry objects or hair patches, rendering of long hair using this approach does not seem obvious.

A brute force method to render hair is to model each individual hair as curved cylinder and render each cylinder primitive. The shear number of primitives modeling hair poses serious problem to this approach. However, the explicit modeling of hair has been used for different reasons employing different types of primitives.

An early effort by Csuri *et al* [3] generated fur-like volumes using polygons. Each hair was modeled as a single triangle laid out on a surface and rendered using a Z-buffer algorithm for hidden surface removal. Miller [16] produced better results by modeling hair as pyramids consisting of triangles. Oversampling was employed for anti-aliasing. These techniques however, impose serious problems considering reasonable number and size of hairs.

In an another approach, a hardware Z-buffer renderer was used with Gouraud shading for rendering hair modeled as connected segments of triangular prisms on a full human head. However, the illumination model used was quite simplistic and no effort was done to deal with the problem of aliasing. LeBlanc *et al* [14] proposed an approach of rendering hair using pixel blending and shadow buffers. This technique has been one of the most effective and practical hair rendering approach. Though it could be applied for the variety of hairy and furry objects, one of the primary intention of the approach was to be able to render realistic different styles of human hairs. Hair rendering is done by mix of ray tracing and drawing polyline premitives, with added module for

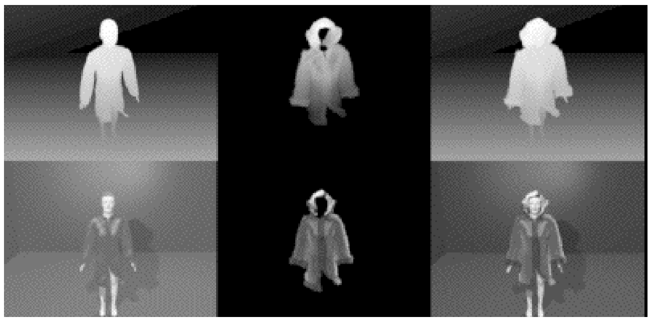


Figure 11: Rendering pipeline of the method-”Pixel Blending and Shadow Buffer”

the shadow buffer [20]. The rendering pipeline has the following steps: first the shadow of the scene is calculated for each light source. Then, hair shadow buffer is computed for each light source for the given hair style model; this is done by drawing each hair segment into a Z-buffer and extracting the depth map. The depth maps for the shadow buffers for the scene and hair are composed giving a single composite shadow buffer for each light source. The scene image with its Z-buffer is generated using scene model and composite shadow buffers. The hair segments are then drawn as illuminated polylines [27] into the scene using Z-buffer of scene for determining the visibility and the composite shadow buffers for finding the shadows. Figure 11 shows the process and Figure 12 gives final rendered image of a hairstyle of a synthetic actor with a fur coat.

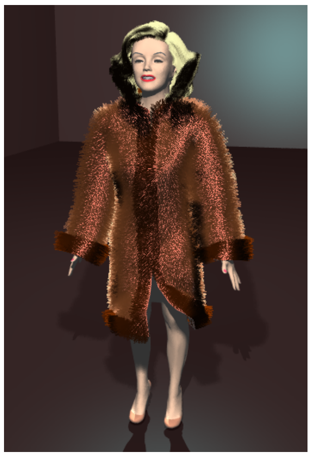


Figure 12: Fur using Explicit Hair Model

Special effects like rendering wet hair require change in the shading model. Bruderlin [1] presented some simple ways to account for the wetness of hair -changing the specularity. That is, hairs on the side of a clump facing the light are brighter than hairs on a clump away from the light.

Kong and Nakajima *et al* [13] presented an approach of using visible volume buffer to reduce the rendering time. The volume buffer is a 3D cubical space defined by the user depending upon the available memory and the resolution required. They consider hair model as combination coarse background hair and detailed surface hair determined by the distance from the viewpoint or the opacity value. The technique reduces considerably the rendering time, however, the quality of results is not so impressive.

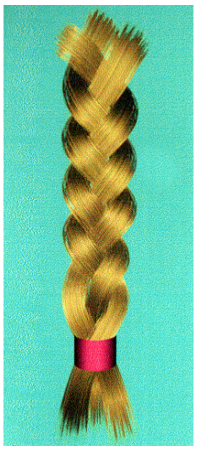


Figure 13: Braid rendered using generalized cylinders and volumetric texture, by Yan *et al*

Yan *et al* [26] combine volumetric texture inside the explicit geometry of hair cluster defined as a generalized cylinder. Ray tracing is employed to get the boundaries of the generalized cylinder and then the standard volume rendering is applied along the ray to capture the characteristics of the density function defined. This may be considered as a hybrid approach for hair rendering.

5 Conclusion

	Hair Modeling	Hair Animation	Hair Rendering
Explicit Models	effective - *tedious to model* - *not suitable for knots and braids*	adequate - *expensive due to size* - *inappropriate for hair-hair interaction*	fast - *inadequate for self-shadowing*
Particle Systems	inappropriate	adhoc - *lacks physical basis* - *no hair-hair interaction*	effective - *lacks shadowing and self-shadowing*
Volumetric Textures	effective - *not suitable for long hair*	limited - *via Animated Shape Perturbation*	effective - *expensive*
Cluster Model	effective - *not suitable for simple smooth hair*	not done - *via Animated Shape Perturbation*	effective
Hair as a Fluid	effective - *not suitable for knots and braids*	not done	not done

Figure 14: Comparison of the various hair models

In this paper we present the state of the art in hair simulation, one of the most challenging problem of virtual humans. We consider three aspects in hair simulation: hair shape modeling, hair rendering and hair dynamics. Different approaches have been proposed in the literature dealing with one or more aspects of hair simulation. We divide them into four categories based on the underlying technique: particle systems, volumetric textures, explicit hair models and cluster hair model. Some of these techniques are appropriate and effective only for one of the aspects in hair simulation. In figure 14, we summarize their role with their effectiveness and limitations for each aspect of the hair simulation. Notice that we have introduced a new hair modeling paradigm - “Hair as a Fluid”. We believe, this approach has good potential in terms of hair shape modeling and hair dynamics, as the methodology gives a basis for modeling complex hair-hair interactions.

No, doubt research in hair simulation despite the inherent difficulty of its size has been encouraging and shown remarkable improvements over the years. People in general are not ready to accept a bald digital actor or an animal without fur. Such realism to computer graphics characters is also becoming more widely available to the animators. Many of the commercial software provide suitable solutions and plug ins for creating hairy and furry characters. An article by Robertson [21] gives an overview of various techniques available for animators.

However, the quest of realism increases after noticing what one can already achieve. This asks to continue our research for better solutions. Hair dynamics for instance remains an area, where existing computing resources impose constraints. It is still very far to imagine real time hair blowing with full rendering and collisions. Hair dressing and styling also require flexible and convenient modeling paradigms. Fast and effective rendering methods for all hair styles -short or long, in all conditions -dry or wet, modeling all the optical properties of hair are still to be explored. So there is still long way to go.

6 Acknowledgements

This work is supported by the Swiss National Research Foundation (FNRS).

References

[1] Bruderlin, A. A method to generate wet and broken-up animal fur. *Pacific Graphics '99* (October 1999). Held in Seoul, Korea.

[2] Chen, L.-H., Saeyor, S., Dohi, H., and Ishizuka, M. A system of 3d hair style synthesis based on the wisp model. *The Visual Computer 15*, 4 (1999), 159–170. Springer-Verlag, ISSN 0178-2789.

[3] CSURI, C., HAKATHORN, R., AND PARENT, R. Towards an interactive high visual complexity animation system. In *Computer Graphics* (1979).

[4] DALDEGAN, A., AND MAGNENAT-THALMANN, N. Creating virtual fur and hair styles for synthetic actors. In *Communicating with Virtual Worlds* (1993), N. Magnenat-Thalmann and D. Thalmann, Eds., Springer-Verlag.

[5] DALDEGAN, A., THALMANN, N. M., KURIHARA, T., AND THALMANN, D. An integrated system for modeling, animating and rendering hair. *Computer Graphics Forum (Eurographics '93) 12*, 3 (1993), 211–221. Held in Oxford, UK.

[6] DISCHLER, J.-M. A general model of animated shape perturbation. *Graphics Interface '99* (June 1999), 140–147. ISBN 1-55860-632-7.

[7] EBERT, D. S., MUSGRAVE, F. K., PEACHEY, D., PERLIN, K., AND WORLEY, S. *Texturing and Modeling*. Academic Press, 1998.

[8] GELDER, A. V., AND WILHELMS, J. An interactive fur modeling technique. *Graphics Interface '97* (May 1997), 181–188. ISBN 0-9695338-6-1 ISSN 0713-5424.

[9] GOLDMAN, D. B. Fake fur rendering. *Proceedings of SIGGRAPH 97* (August 1997), 127–134. ISBN 0-89791-896-7. Held in Los Angeles, California.

[10] HADAP, S., AND MAGNENAT-THALMANN, N. Interactive hair styler based on fluid flow. In *Proceedings of Eurographics Workshop on Computer Animation and Simulation '2000* (2000). to appear.

[11] ICHI ANJYO, K., USAMI, Y., AND KURIHARA, T. A simple method for extracting the natural beauty of hair. *Computer Graphics (Proceedings of SIGGRAPH 92) 26*, 2 (July 1992), 111–120. ISBN 0-201-51585-7. Held in Chicago, Illinois.

[12] KAJIYA, J. T., AND KAY, T. L. Rendering fur with three dimensional textures. *Computer Graphics (Proceedings of SIGGRAPH 89) 23*, 3 (July 1989), 271–280. Held in Boston, Massachusetts.

[13] KONG, W., AND NAKAJIMA, M. Visible volume buffer for efficient hair expression and shadow generation. *Computer Animation '99, IEEE Computer Society* (May 1999). IEEE Press, Held in Geneva, Switzerland.

[14] LEBLANC, A., TURNER, R., AND THALMANN, D. Rendering hair using pixel blending and shadow buffer. *Journal of Visualization and Computer Animation 2* (1991), 92–97. John Wiley.

[15] LEWIS, J.-P. Algorithms for solid noise synthesis. *Computer Graphics (Proceedings of SIGGRAPH 89) 23*, 3 (July 1989), 263–270. Held in Boston, Massachusetts.

[16] MILLER, G. S. P. From wire-frames to furry animals. *Graphics Interface '88* (June 1988), 138–145.

[17] PERLIN, K. An image synthesizer. *Computer Graphics (Proceedings of SIGGRAPH 85) 19*, 3 (July 1985), 287–296. Held in San Francisco, California.

[18] PERLIN, K., AND HOFFERT, E. M. Hypertexture. *Computer Graphics (Proceedings of SIGGRAPH 89) 23*, 3 (July 1989), 253–262. Held in Boston, Massachusetts.

[19] REEVES, W. T. Particle systems - a technique for modeling a class of fuzzy objects. *ACM Transactions on Graphics 2*, 2 (April 1983), 91–108. Held in USA.

[20] REEVES, W. T., SALESIN, D. H., AND COOK, R. L. Rendering antialiased shadows with depth maps. *Computer Graphics (Proceedings of SIGGRAPH 87) 21*, 4 (July 1987), 283–291. Held in Anaheim, California.

[21] ROBERTSON, B. Hair-raising effects. *Computer Graphics World, Magazine* (October 1995).

[22] ROSENBLUM, R., CARLSON, W., AND TRIPP, E. Simulating the structure and dynamics of human hair: Modeling, rendering and animation. *Journal of Visualzation and Computer Animation 2* (June 1991), 141–148. John Wiley.

[23] TSUNEYA KURIHARA, KEN-ICHI ANJYO, D. T. *Models and Techniques in Computer Animation*. Springer-Verlag, ch. Hair Animation with Collision Detection.

[24] WATANABE, Y., AND SUENAGA, Y. Drawing human hair using wisp model. In *Proceedings of Computer Graphics International'89* (1989), Springer Verlag, pp. 691–700.

[25] WATANABE, Y., AND SUENAGA, Y. A trigonal prism-based method for hair image generation. *IEEE Computer Graphics & Applications 12*, 1 (January 1992), 47–53.

[26] YAN, X. D., XU, Z., YANG, J., AND WANG, T. The cluster hair model. *Journal of Graphics Models and Image Processing* (1999). Academic Press.

[27] ZCKLER, M., STALLING, D., AND HEGE, H.-C. Interactive visualization of 3d-vector fields using illuminated streamlines. *IEEE Visualization '96* (October 1996), 107–114. ISBN 0-89791-864-9.

A Multiresolution Technique for Hairstyling

Tae-Yong Kim*
tae@rhythm.com

Rhythm & Hues Studio

*formerly at the University of Southern California

In this talk, I will explain the details of the multiresolution hair modeling technique that was originally presented in SIGGRAPH 2002. The paper introduced a constructive hair modeling system with which users can sculpt a wide variety of hairstyles. In the multiresolution hair modeling (MHM) 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.

Additional Materials

1. Tae-Yong Kim and Ulrich Neumann, Interactive Multiresolution Hair Modeling and Editing, ACM SIGGRAPH Proceedings 2002 (reprinted in the course note).
2. Tae-Yong Kim, Modeling, Rendering, and Animating Human Hair, Ph. D. Dissertation, University of Southern California, 2002 (available at http://graphics.usc.edu/~taeyong)
3. Links to more papers on hair modeling can be found at http://graphics.usc.edu/~taeyong/Links.htm

Modeling human hair

- Complex discontinuous volume
- Clusters and curliness
- Virtually any shape is possible
- Number of hair strands (100K to 200K)

Modeling human hair such as the one shown in the photograph is known to be difficult for many reasons. To mention a just few, human hair is inherently a complex discontinuous volume formed by clustering effects due to hair/hair interaction and styling artifacts such as curling. Through hairstyling, a hair volume can be transformed to virtually any shape. All these complexities become even worse if we consider the number of hair strands to deal with which typically ranges from 100,000 to 200,000

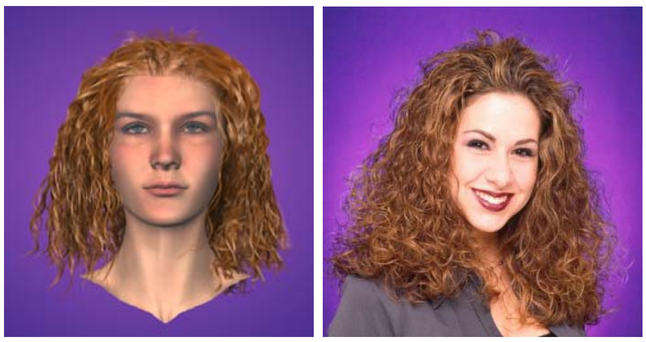
Goal

To build an interactive system for modeling complex human hair in a reasonable amount of time (< 1 hour)

- Efficient sculpting of volumetric hair model
- Interactive rendering of arbitrary explicit hair models

Our goal is to develop an interactive hair modeling system that allows the user to model complex human hair in a reasonable amount of time (say, in less than an hour). We aim at developing tools with which the user can efficiently sculpt a wide variety of volumetric hair model. We also want the user to be able to interactively edit hair models in a WYSIWYG fashion, by providing an interactive rendering capability for arbitrary hair models.

Goal

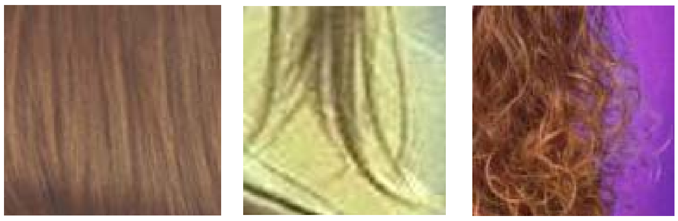


A multi-res hair model

photograph

In this example, we tried to create a hairstyle that looks similar to the hairstyle shown on the photograph on the right side. The model construction time was about an hour.

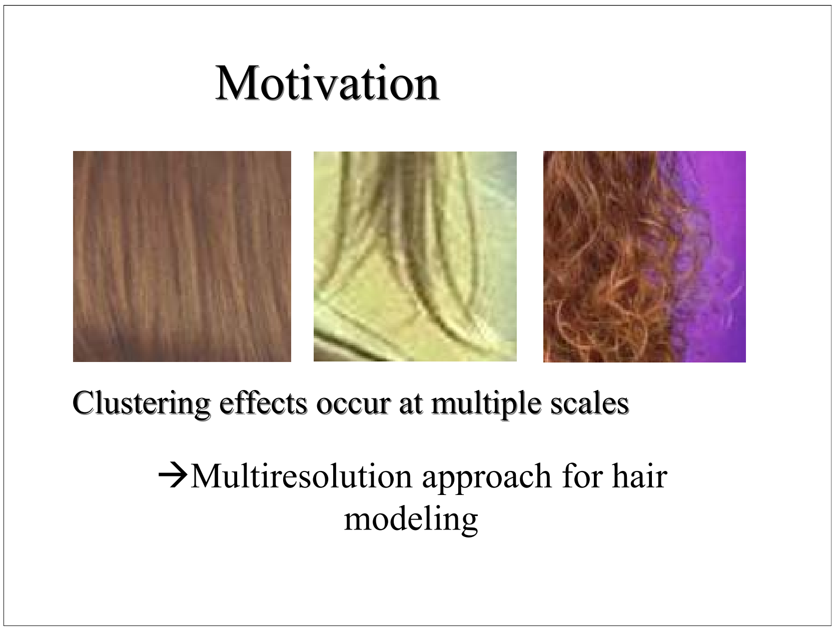
Common Problems



Long hairs cluster, split, and curl away

→Complex hairstyles with extreme discontinuity are difficult to model

What is missing in previous efforts in hair modeling is the ability to model complex discontinuous hairstyles as shown in these photographs. Due to the hair/hair interaction and styling procedure, long hairs tend to form a cluster that can split and curl away from each other. These types of complex hairstyles are difficult to model with existing techniques since most existing techniques rely on the assumption that the geometry of hair varies smoothly and coherently.



Motivation

Clustering effects occur at multiple scales

→Multiresolution approach for hair modeling

The multiresolution hair modeling technique was motivated by the observation that these discontinous clustering effects often occur at multiple scales. When viewed from distance, a hairstyle appears to consist of a few big clusters. But if we examine carefully, we can see that each big cluster consists of several smaller clusters. The multiresolution concept has been used in many areas of computer graphics. The idea is to use multiple levels of controls when we deal with complex objects, starting from global shapes down to small scale details. We apply this idea to hair modeling.

A multiresolution appraoch for hair modeling

- Hair cluster shape modeling with generalized cylinders (GC)
- Subdivision and hierarchical hair structure
- Multiresolution editing tools
- Copy and paste operations
- Capture *structural* aspects of volumetric hairstyles

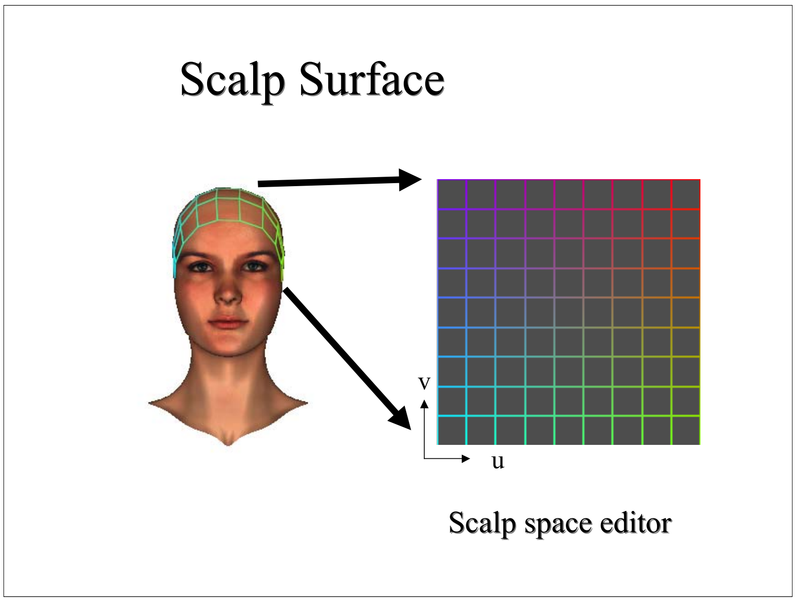
We formulate a multiresolution framework for hair modeling as follows. The user first designs the global shape of a hair model with a small number of generalized cylinder(GC)s. To add more details, the generalized cylinders are subdivided into smaller ones, constructing a hierarchical hair structure. Multiresolution editing tools are applied for editing an evolving hair model at any level of detail. Copy and paste operations transfer details from any portion of the hair model to another. With these tools, our hope is to provide the user a necessary engine to model and capture the structural aspects of complex volumetric hairstyles. Our system is a constructive hair modeling system where the user starts from scratch with a specific head model

Scalp Surface

- A parametric surface

  $S = P(u,v)$
- Defines the region of hair growth

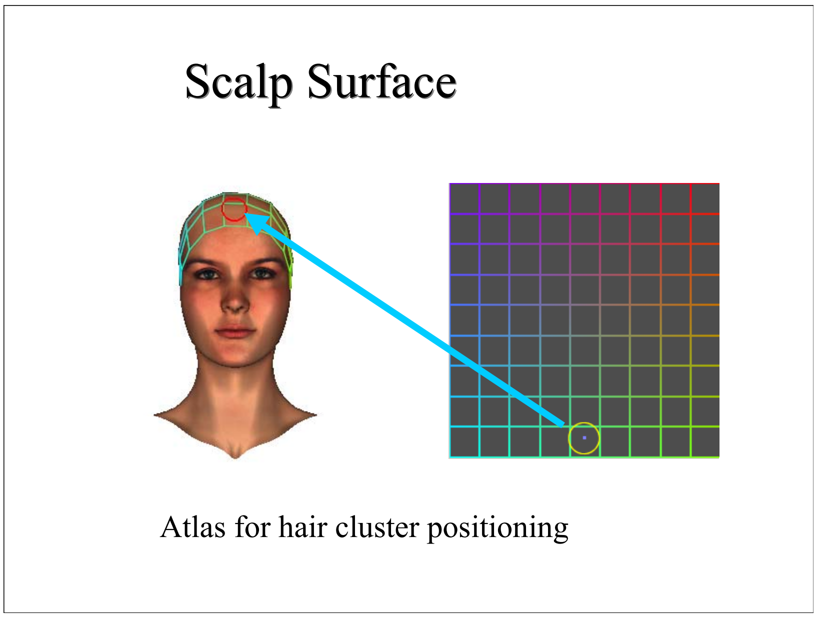
For each head mesh, we define a parametric surface that we call a scalp surface. The scalp surface defines the region on the head where the user can place hairs.



Scalp Surface

Scalp space editor

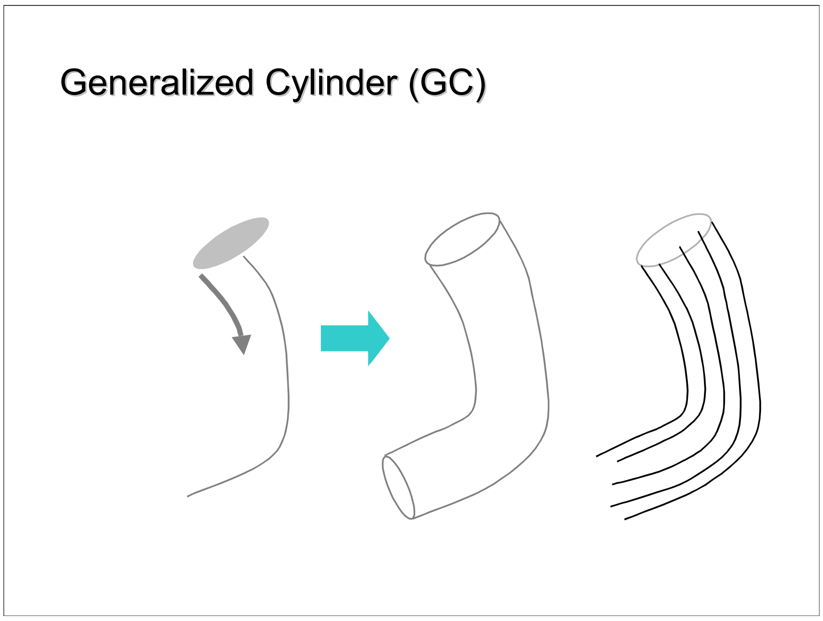
The parametric space of the scalp surface is mapped to a 2D editor that we call a scalp space editor.



Scalp Surface

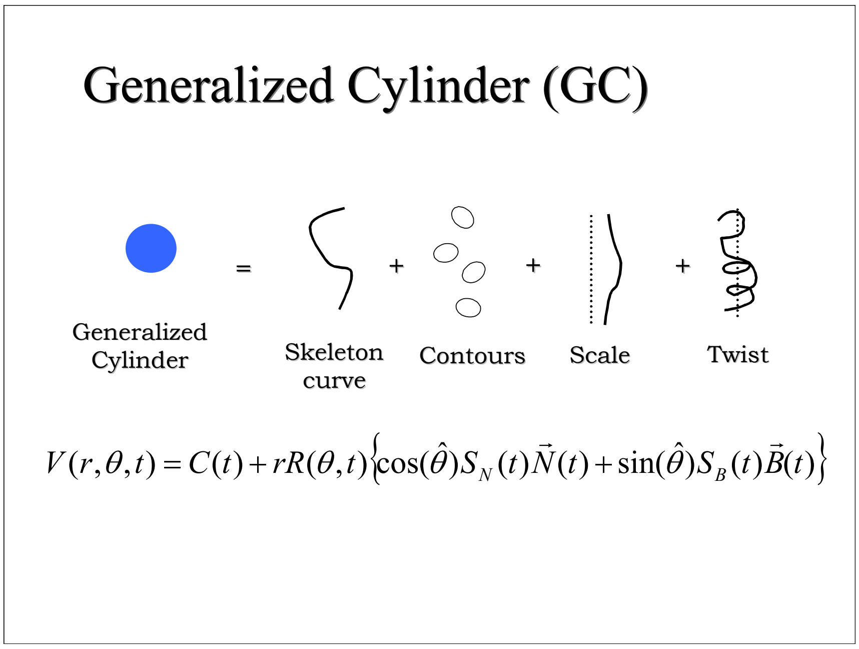
Atlas for hair cluster positioning

This 2D scalp space forms an atlas where the user can specify the root position of each hair cluster. Once the user places a contour on the scalp, a generalized cylinder is created and the user can control a group of hairs by manipulating this generalized cylinder.



Generalized Cylinder (GC)

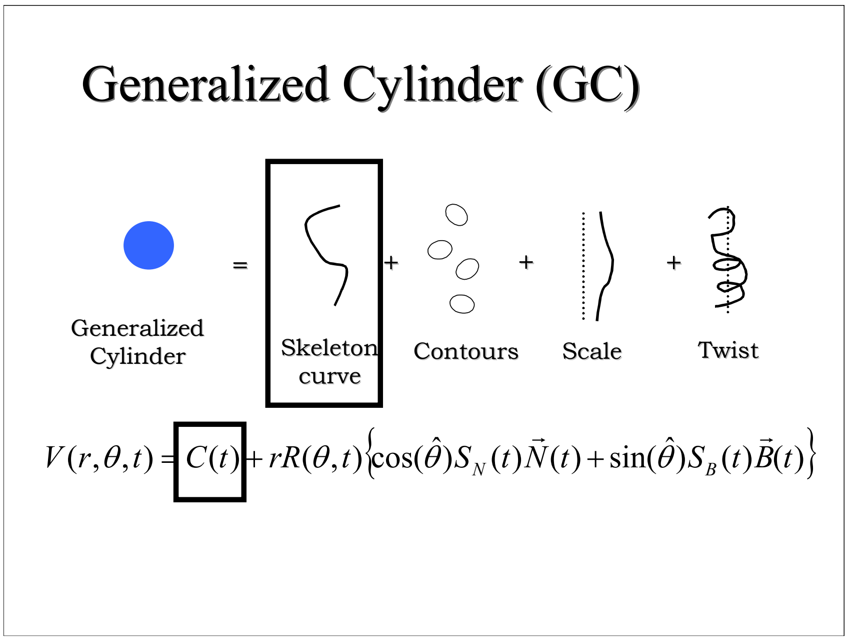
Once a generalized cylinder is defined, we can place hairs within its boundary.



Generalized Cylinder (GC)

$$V(r,\theta,t) = C(t) + rR(\theta,t)\left\{\cos(\hat{\theta})S_N(t)\vec{N}(t) + \sin(\hat{\theta})S_B(t)\vec{B}(t)\right\}$$

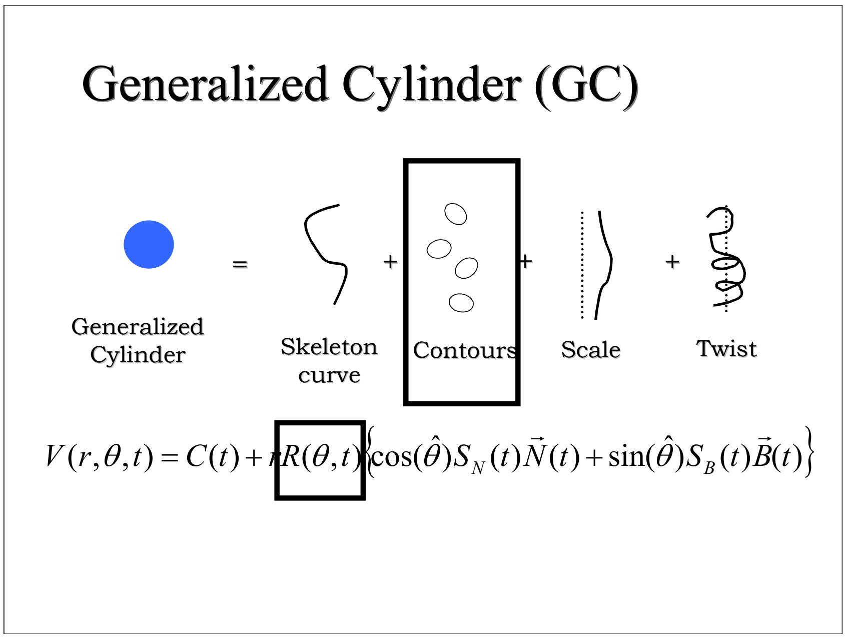
We represent a generalized cylinder (GC) as combination of four components – skeleton curves, contours, scale and twist terms. The equation of the GC is shown in the slide.



Generalized Cylinder (GC)

$$V(r,\theta,t) = \boxed{C(t)} + rR(\theta,t)\left\{\cos(\hat{\theta})S_N(t)\vec{N}(t) + \sin(\hat{\theta})S_B(t)\vec{B}(t)\right\}$$

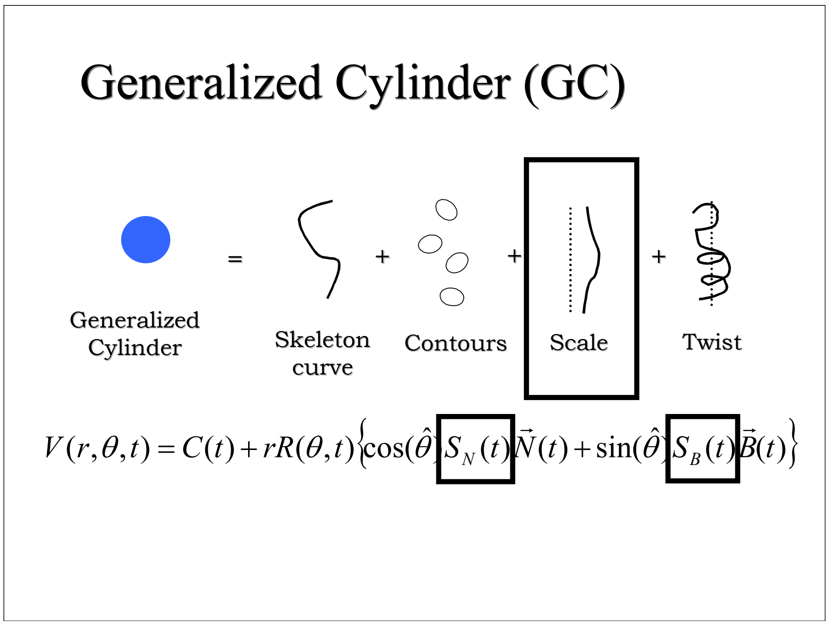
A parametric skeleton curve defines the overall shape of the GC. We use a Catmull-Rom Spline curve to represent the skeleton curve *C(t)*.



Generalized Cylinder (GC)

$$V(r,\theta,t) = C(t) + \boxed{rR(\theta,t)}\left\{\cos(\hat{\theta})S_N(t)\vec{N}(t) + \sin(\hat{\theta})S_B(t)\vec{B}(t)\right\}$$

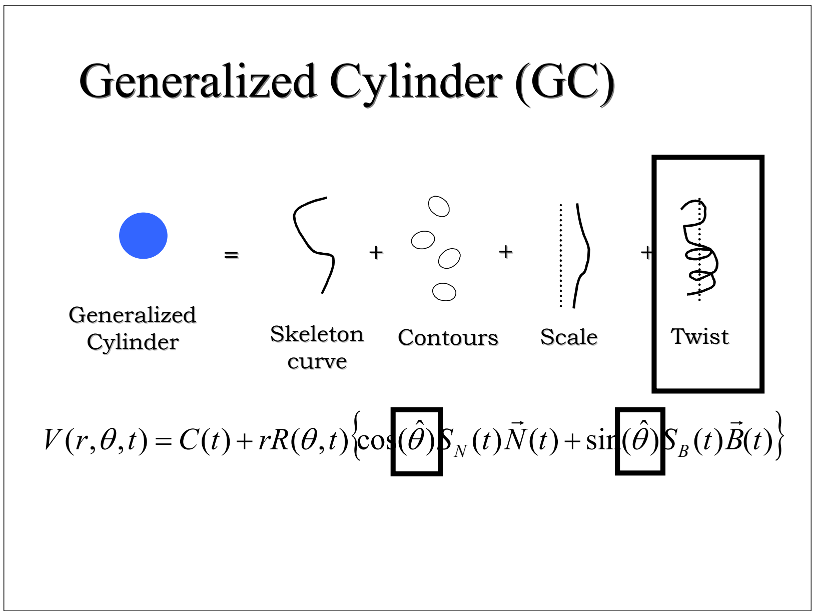
Contour functions R(θ,t) are placed along the skeleton curve and define the boundary of the generalized cylinder



Generalized Cylinder (GC)

$$V(r,\theta,t) = C(t) + rR(\theta,t)\left\{\cos(\hat{\theta})\boxed{S_N(t)}\vec{N}(t) + \sin(\hat{\theta})\boxed{S_B(t)}\vec{B}(t)\right\}$$

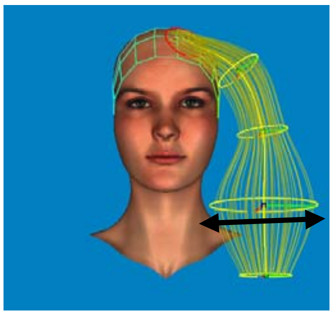
And the two auxiliary functions (scale, and twist) are used.



Generalized Cylinder (GC)

$$V(r,\theta,t) = C(t) + rR(\theta,t)\left\{\cos\boxed{(\hat{\theta})}S_N(t)\vec{N}(t) + \sin\boxed{(\hat{\theta})}S_B(t)\vec{B}(t)\right\}$$

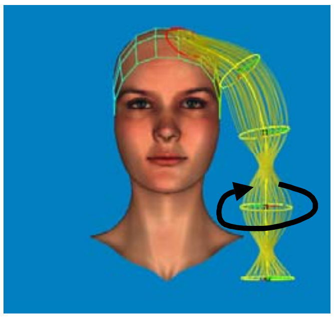
Scale Change



$$V(r,\theta,t) = C(t) + rR(\theta,t)\left\{\cos(\hat{\theta})\boxed{S_N(t)}\vec{N}(t) + \sin(\hat{\theta})\boxed{S_B(t)}\vec{B}(t)\right\}$$

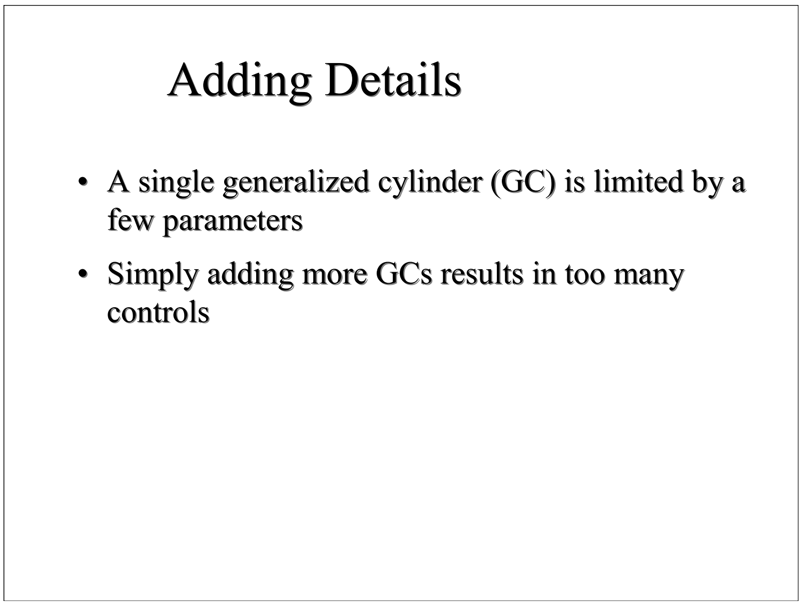
Here is an example of editing those components. In this example, the user changes the scaling term to fatten the hair cluster.

Twist Change



$$V(r,\theta,t) = C(t) + rR(\theta,t)\left\{\cos\boxed{(\hat{\theta})}S_N(t)\vec{N}(t) + \sin(\hat{\theta})\boxed{S_B(t)}\vec{B}(t)\right\}$$

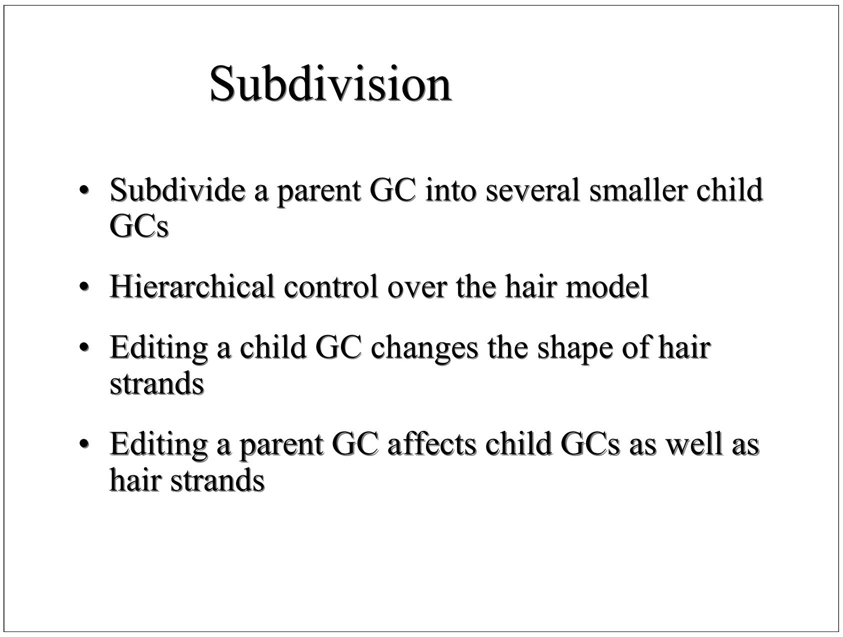
And in this example, the user curls a hair cluster by changing the twist term.



Adding Details

- A single generalized cylinder (GC) is limited by a few parameters
- Simply adding more GCs results in too many controls

The details that can be modeled through a single generalized cylinder are limited by a few parameters that we saw (contours, twist, and scale functions). To add more details to the model, one can simply increase the number of GCs. But, doing so will also increase the number of controls that the user has to deal with.

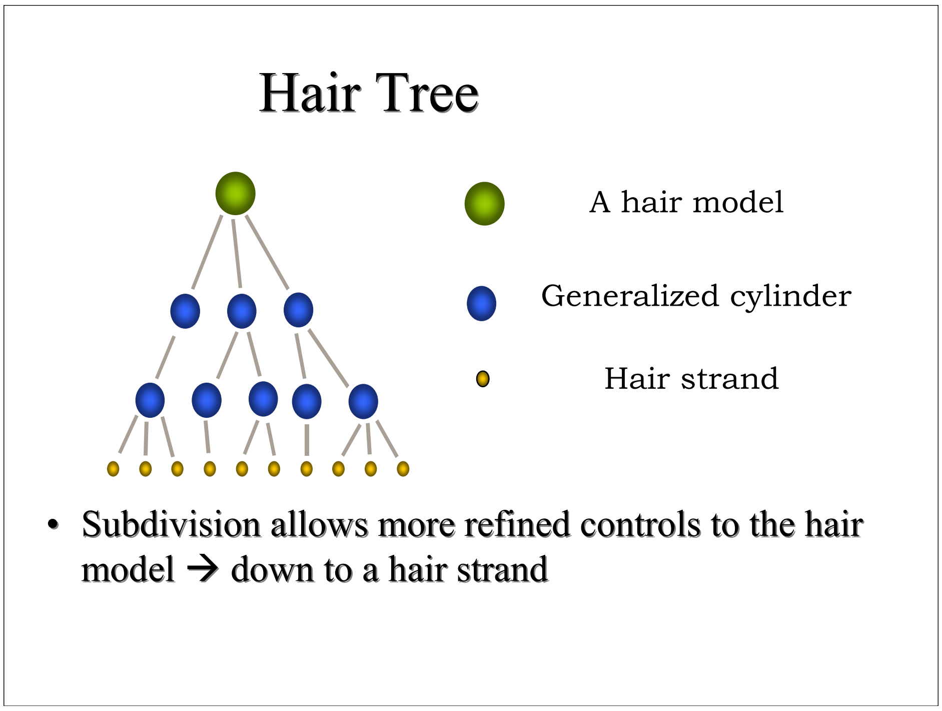


Subdivision

- Subdivide a parent GC into several smaller child GCs
- Hierarchical control over the hair model
- Editing a child GC changes the shape of hair strands
- Editing a parent GC affects child GCs as well as hair strands

For more efficient control, we subdivide a GC, the parent, into several smaller ones. Using the parent-children relationship, we provide a hierarchical control over the hair model.

Editing a child GC changes the shape of hair strands that are controlled by this GC. On the other hand, editing a parent GC changes the shape of its children, consequently changing the shape of hair strands, in a hierarchical manner.



Hair Tree

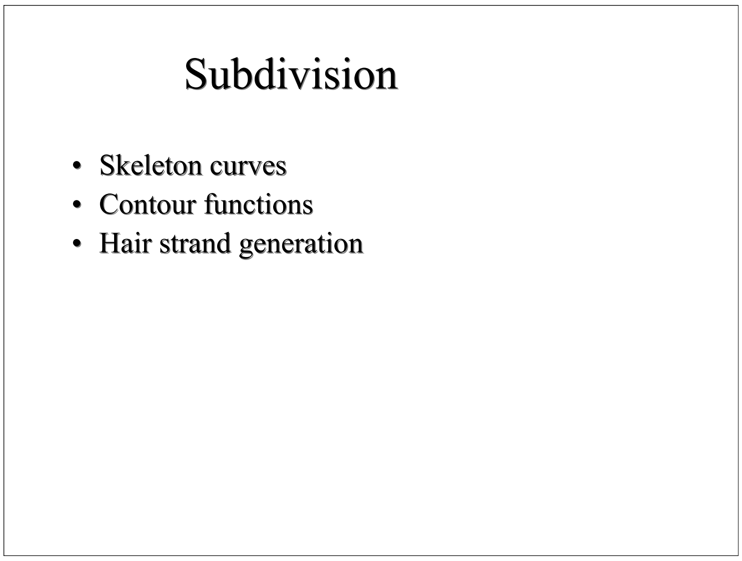
● A hair model

● Generalized cylinder

● Hair strand

- Subdivision allows more refined controls to the hair model → down to a hair strand

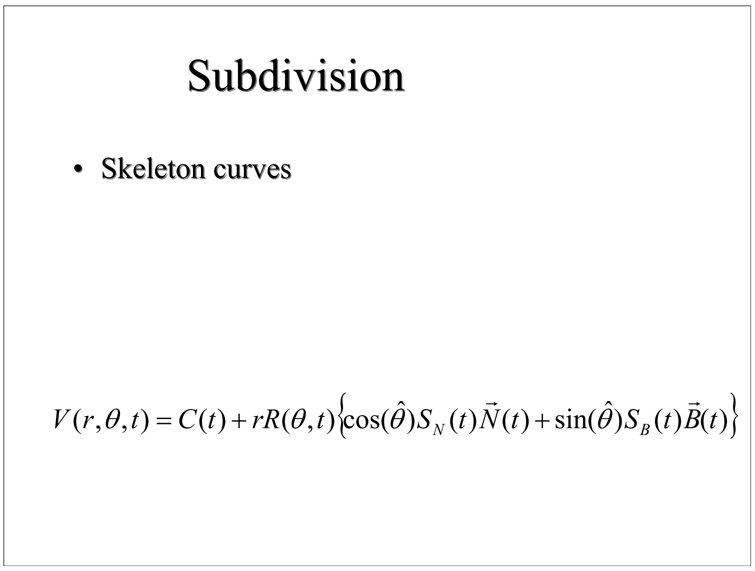
The parent-children relationship between clusters is stored in a tree structure that we call a hair tree. The entire hair model consists of a small number of top-level generalized cylinders and a large number of smaller GCs that stem from them. At the bottom of the hierarchy, each hair strand is controlled by these GCs. The user can subdivide a hair cluster repeatedly until the control over the desired level of detail is achieved.



Subdivision

- Skeleton curves
- Contour functions
- Hair strand generation

To subdivide a generalized cylinder, we need to create the skeleton curves, contour functions and other parameters of each child GC.



Subdivision

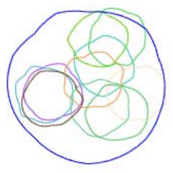
- Skeleton curves

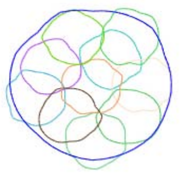
$$V(r,\theta,t) = C(t) + rR(\theta,t)\left\{\cos(\hat{\theta})S_N(t)\vec{N}(t) + \sin(\hat{\theta})S_B(t)\vec{B}(t)\right\}$$

The skeleton curve of each child GC is created using the generalized cylinder equation of the parent.

Subdivision

- Contour functions

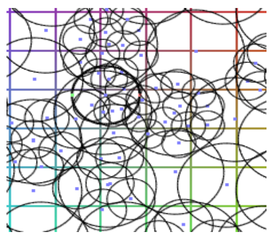

1. Random Positioning


2. Position Relaxation [Turk1991]

The contour functions of child GCs are first copied from the parent and scaled down and placed randomly inside the contour of parent GC. The positions of these contours are then evenly redistributed through a position relaxation step.

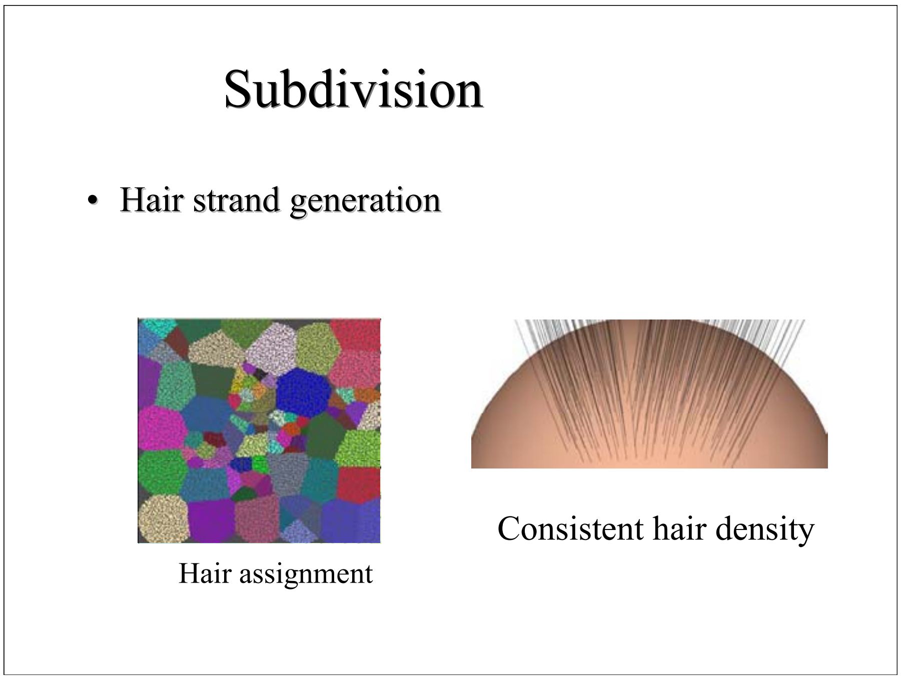
Subdivision

- Hair strand generation


Overlaps and holes

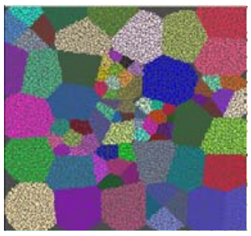
Uneven hair density

Since we allow arbitrarily shaped contours, these contours can overlap and leave holes between the boundaries of GCs. If we generate hair strands independently inside each GC, this can result in visually distracting hair density variation in the overlapping regions



Subdivision

- Hair strand generation


Hair assignment

Consistent hair density

To avoid this problem, we first uniformly sample the root position of hairs and then assign each hair strand to GCs. In the image on the left side, the root position of each hair strand on the scalp is drawn as colored dot. Colors indicate the GCs that controls the hair strands. This way, we can maintain consistent hair densities across the entire scalp. A pseudo-code to find the 'owner' GC of a hair strand is shown below.

```
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
```

Multiresolution Editing

Now we show an example of the user editing a hair model using the hierarchical control. Note that the details are preserved while a parent cluster is edited.

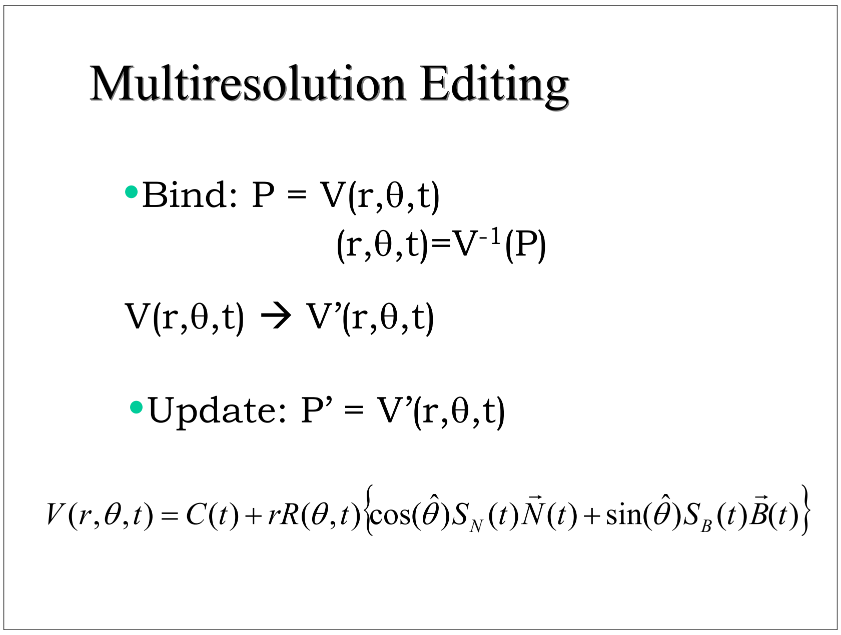
Multiresolution Editing

- Bind

  When a GC is selected for edit, its descendants are attached to the GC
- Update

  During editing, the children GCs are updated using GC equation of the parent

We perform the multiresolution editing operations in two steps. Whenever the user selects and edits a GC, we perform a bind operation that attaches its child nodes to the edited parent. During editing, we update the skeleton curves of its children using the GC equation of the parent.



Multiresolution Editing

- Bind: P = V(r,θ,t)

  (r,θ,t)=V^{-1}(P)

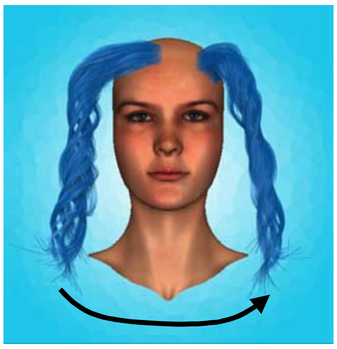
V(r,θ,t) → V'(r,θ,t)

- Update: P' = V'(r,θ,t)

$$V(r,\theta,t) = C(t) + rR(\theta,t)\left\{\cos(\hat{\theta})S_N(t)\vec{N}(t) + \sin(\hat{\theta})S_B(t)\vec{B}(t)\right\}$$

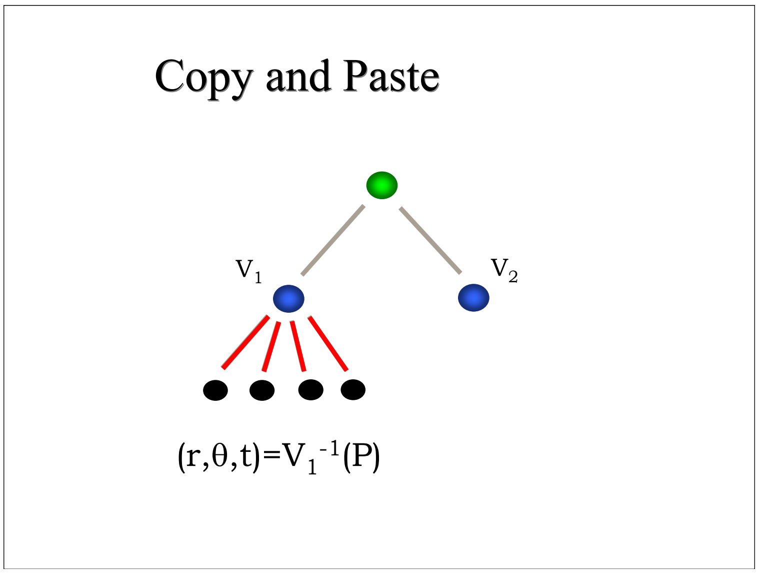
The bind procedure is the inverse transform of the GC equation. For each control point of the skeleton curve of each child, we compute the parametric coordinate r, θ and t of the GC equation. Whenever a parent GC changes by the user, we update the skeleton curves of child GCs with the precomputed parametric coordinates. Details of the binding operations can be found in the reprinted paper.

Copy and Paste

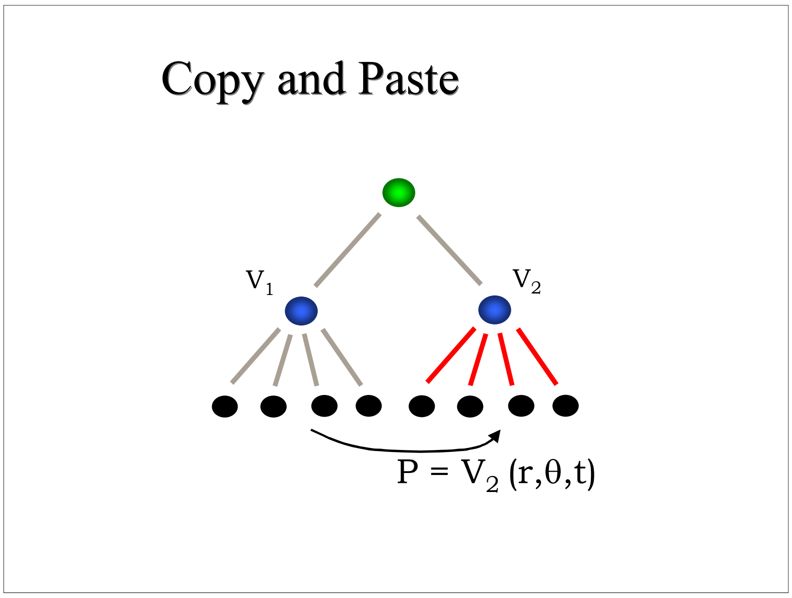


Transfer a *style* from one cluster to another

The bind and update process can be also used to transfer a style from one cluster to another cluster.

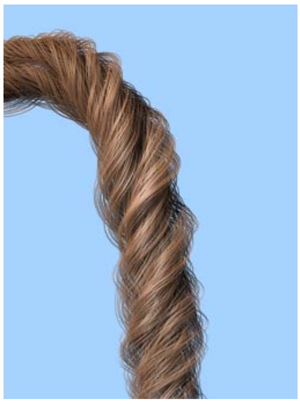


In this case, the parametric coordinates of child clusters of the source model are first computed using the bind operation as before



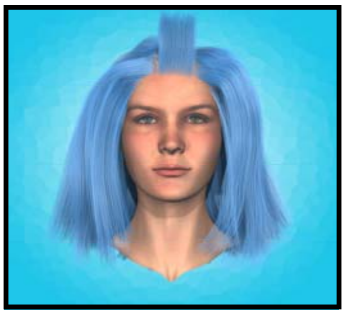
But this time, new child clusters are added to the target model and their skeleton curves are generated using the parametric coordinates computed in the bind operation.

Copy and Paste

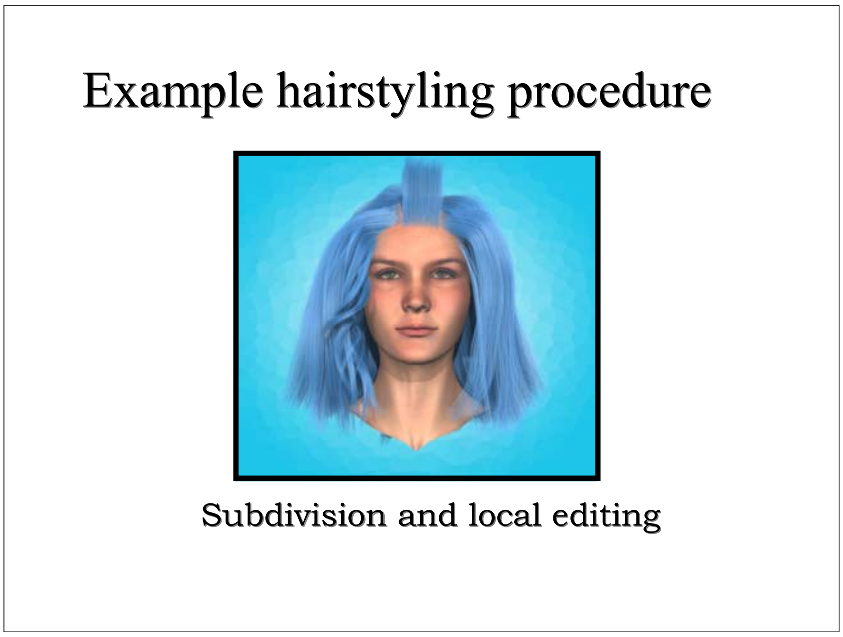


Using the hierarchy of the hair tree, the copy and paste operation can be performed at any level of detail. For example, if a highest level GC is copied, we can recursively copy all the children GCs too. The braid model shown above was created by copying a single braid model at three different levels of details.

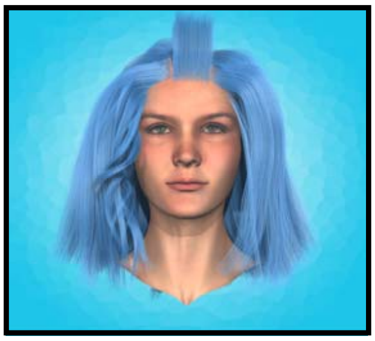
Example hairstyling procedure


Global hair shape modeling with GCs

Now we will go through an example hairstyling process. First, the user designs the overall shape of a hair model with about 20 generalized cylinders.

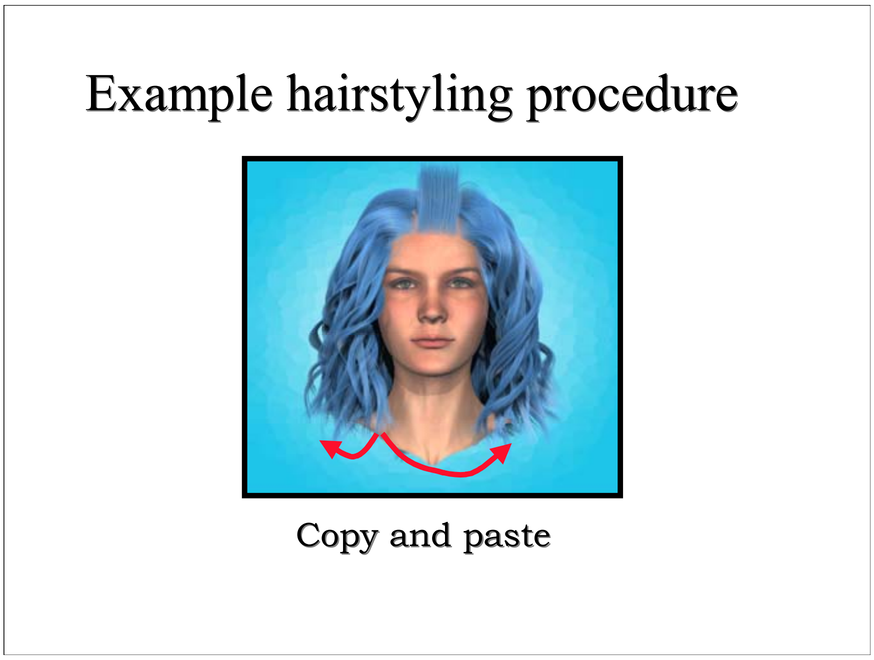


Example hairstyling procedure

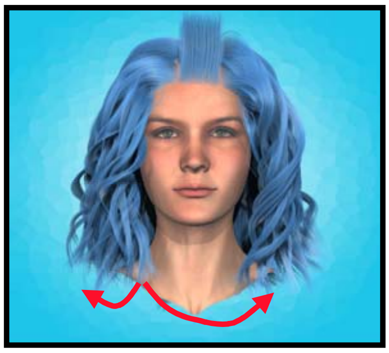


Subdivision and local editing

One hair cluster is subdivided and edited to be curly

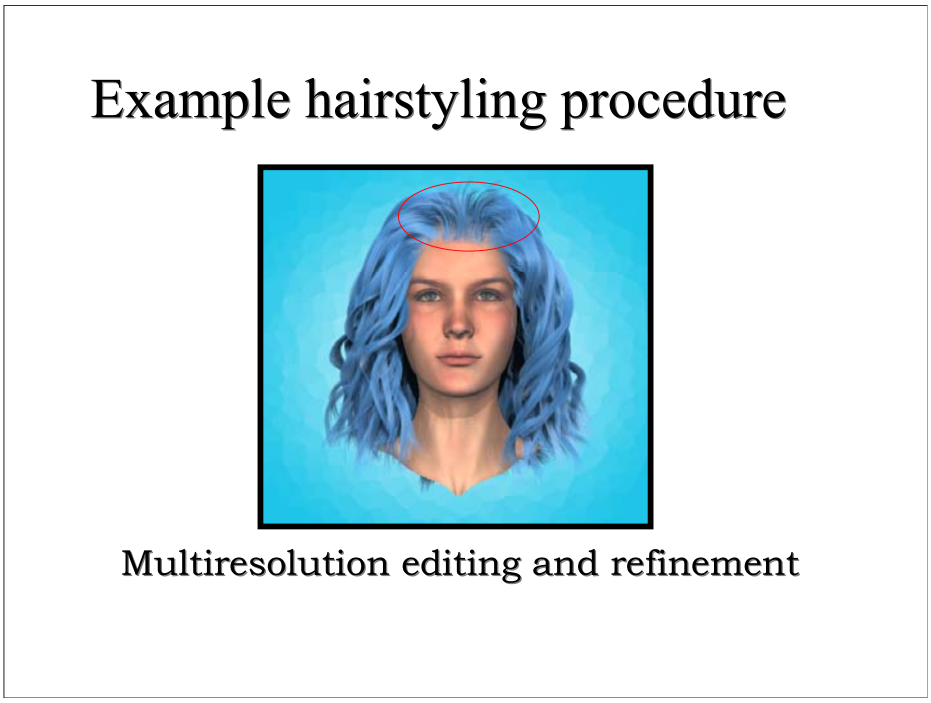


Example hairstyling procedure



Copy and paste

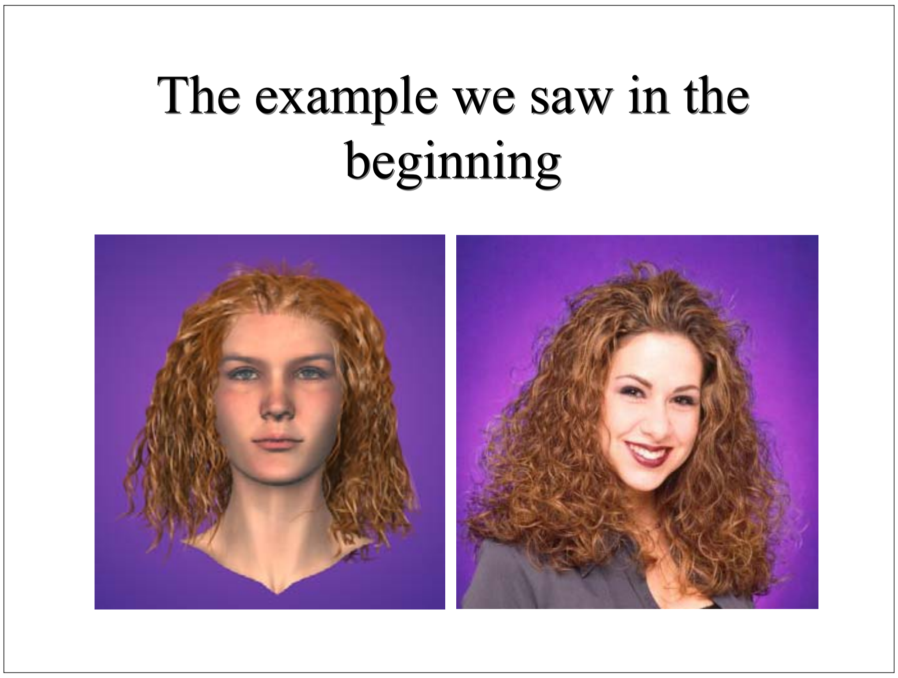
This curly cluster is copied onto all the other clusters on the sides.



Example hairstyling procedure

Multiresolution editing and refinement

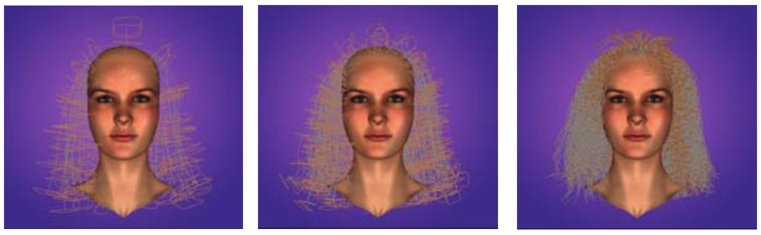
The bang hair cluster is subdivided and the model is further refined

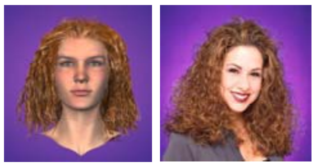


The example we saw in the beginning

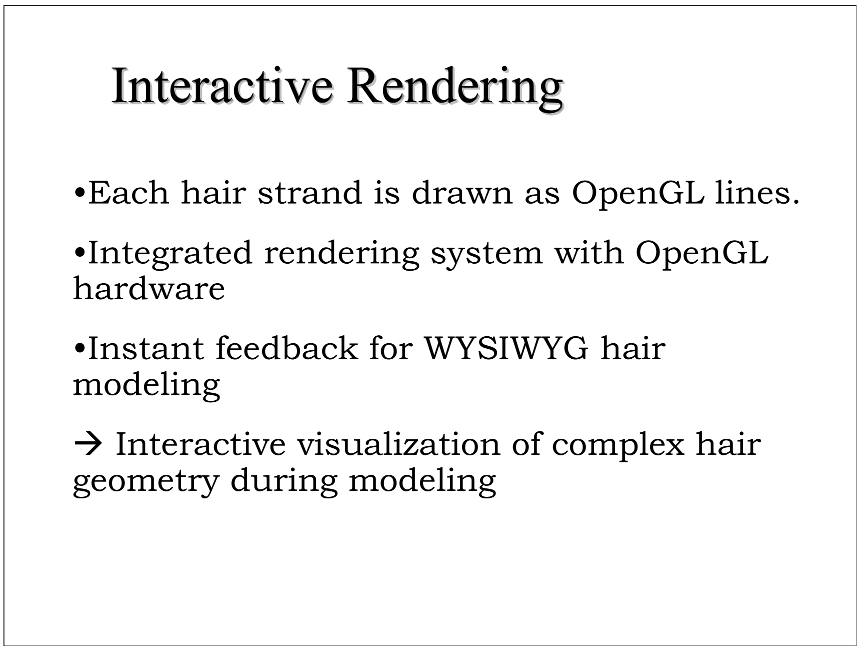
The example we saw in the beginning of this note was created with similar modeling steps.

Example





This model consists of about a thousand generalized cylinders, hierarchically defined at 4 levels of details. At the bottom of the hierarchy, we created about twenty thousand hair strands.



Interactive Rendering

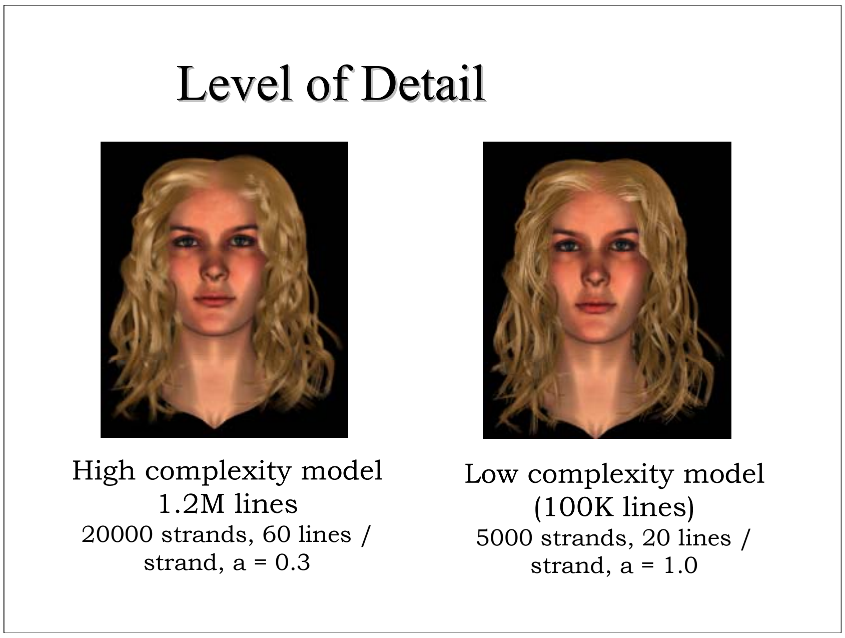
•Each hair strand is drawn as OpenGL lines.

•Integrated rendering system with OpenGL hardware

•Instant feedback for WYSIWYG hair modeling

→ Interactive visualization of complex hair geometry during modeling

Virtually, the shape of any hair strand can be explicitly modeled through the hierarchical control described so far. Since the geometry is explicitly maintained, our model can be rendered with any existing hair renderer. We want the hair renderer to be as fast as possible, to provide the user an instant visual feedback during modeling. To facilitate interactive viewing during modeling, we use integrated hair rendering algorithms that exploit existing openGL hardware as much as possible. For more details of these hair rendering algorithms, see the note 'algorithms for hardware accelerated hair rendering' in this course.

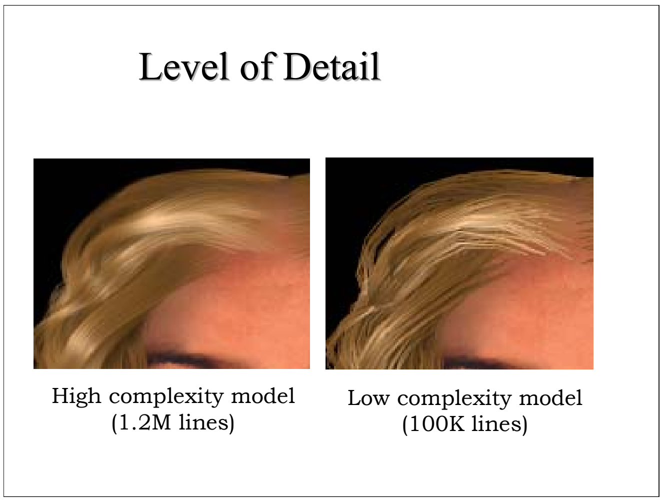


Level of Detail

High complexity model
1.2M lines
20000 strands, 60 lines / strand, a = 0.3

Low complexity model
(100K lines)
5000 strands, 20 lines / strand, a = 1.0

Using a low complexity model can increase the interactivity for distant viewing while a high complexity model can be used for a high quality renderer. The overall look of the model is similar since the 'structure' of the hair model is kept separate in the hierarchical control.



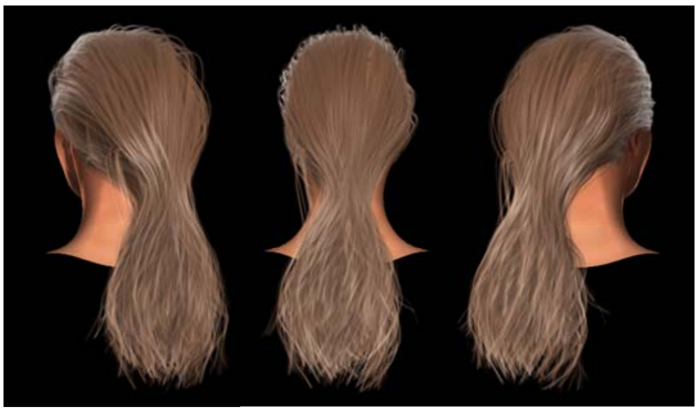
Level of Detail

High complexity model
(1.2M lines)

Low complexity model
(100K lines)

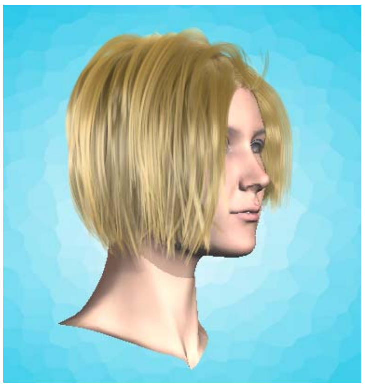
Below are example hair models that were created with the multiresolution hair modeling system. The same face model was used throughout these examples. Note how a hairstyle can change the impression and look on the same face model.

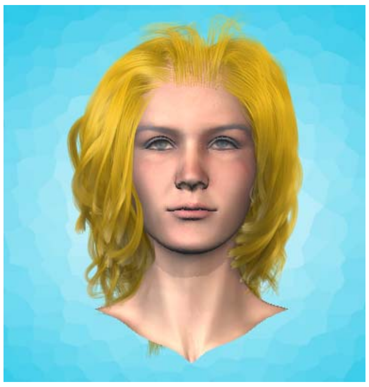
Results - Curly ponytail



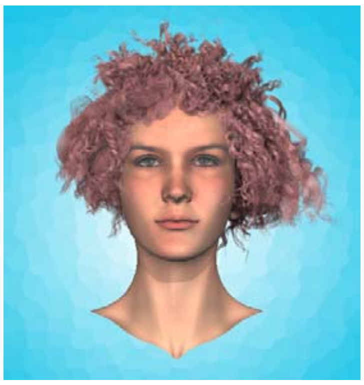
Results

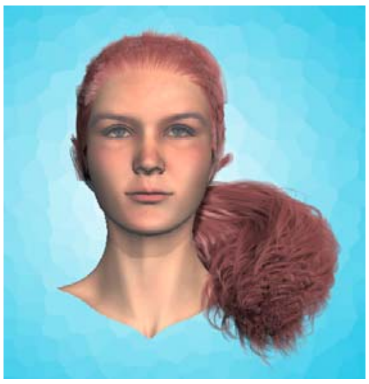
Results



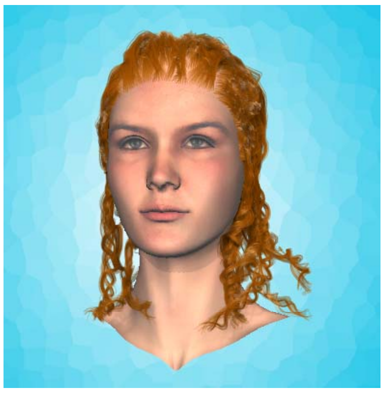


Results

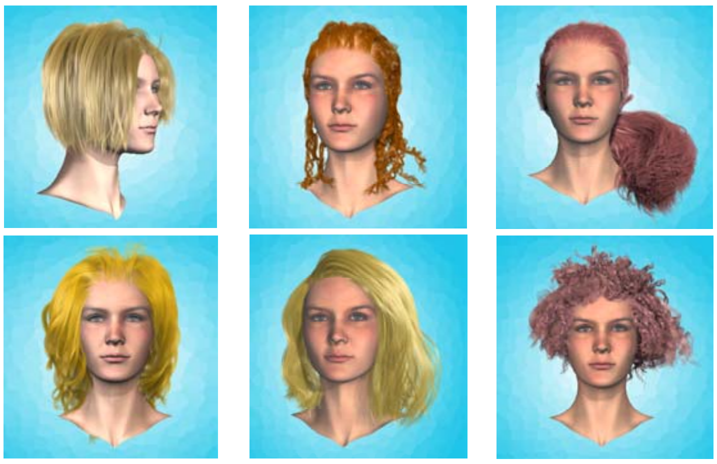




Results



See how a hairstyle changes the look on the same face model!



Hair Shape as Streamlines of Fluid Flow

Sunil Hadap[*]
R&D Staff, PDI/DreamWorks

Nadia Magnenat-Thalmann[†]
MIRALab, University of Geneva

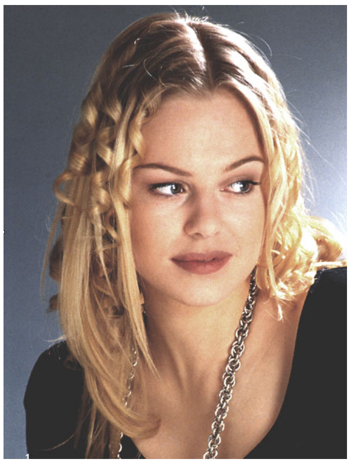


Figure 1: Intricate Hairstyle from “399 Great Hairstyles”

In the past two decades, there have been many advances in shape modeling, dynamics and rendering of general class of fuzzy objects – fur, hair, grass and particle systems [Gelder and Wilhelms 1997; Goldman 1997; Neyret 1998; Reeves 1983]. However, shape modeling of (especially long styled) hair remains an ever challenging problem. Hair shape needs a special attention because it has enormous geometric complexity at hair strand level. Yet, overall there seems to be a definite inherent simplicity and order, observe Figure 1. “399 Great Hairstyles” [Rudiger et al. 1998] is an exhaustive visual reference of various beautiful hairstyles.

For the complete discussion on the State-of-the-Art in Hair Simulation, we refer to [Magnenat-Thalmann et al. 2000; Hadap 2003]. In this section we take a quick recap. There are three main hair simulation tasks – shape modeling, animation and rendering. Each of these tasks can be associated with two broad model categories – explicit hair models [Daldegan et al. 1993; Anjyo et al. 1992; LeBlanc et al. 1991; Rosenblum et al. 1991] and volumetric textures [Neyret 1998; Kajiya and Kay 1989; Lewis 1989; Perlin and Hoffert 1989]. In the explicit hair models, each hair strand is considered for shape and dynamics. Daldegan *et al* [Daldegan et al. 1993] proposed that one could define a few characteristic hair strands and then populate the hair based on them. Similarly, Anjyo *et al* [Anjyo et al. 1992] developed a dynamic model that is based on individual hair strand. Though these models are intuitive and close to reality, they are tedious for shape modeling and numerically expensive for hair dynamics. These difficulties are partially overcome by considering a bunch of hair instead of individual hair strands in the case of the wisp models [Chen et al. 1999; Watanabe and Suenaga 1989; Watanabe and Suenaga 1992; Yang et al. 1999]. In the volumetric texture approach, hair is modeled as a volumetric density function [Kajiya and Kay 1989; Perlin and Hoffert 1989]. Though these models are extremely powerful, one does not have an explicit control over its global shape, they are not intuitive, and they do not represent physical reality. Thus, they are only effective in modeling short hair such as fur. Interestingly, Ebert *et al* [Ebert et al. 1994] introduced a method which enables the global shape control of the volumetric texture. We would like to propose a method in the same spirit.

[*]hadap@acm.org, This work is done at MIRALab, University of Geneva towards completion of PhD
[†]thalmann@miralab.unige.ch

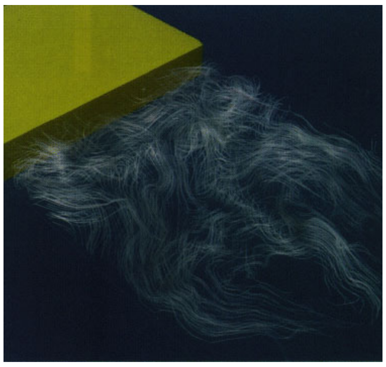


Figure 2: Illuminated Streamlines of Fluid Flow

The strengths of the explicit hair models are their intuitiveness and their ability to control the global hair shape, which are particularly important in the case of modeling long hair. On the contrary, the volumetric textures give a nice way of interpreting complexity in nature and they are rich in local details. We notice that the fluid flow has both the characteristics, which we would like to exploit for hair shape modeling. Some of the flow visualization images [Cabral and Leedom 1993; Zockler et al. 1996] (except for the choice of the colors) also suggest that there are similarities between hair shape and fluid flow, see Figure 2. Hence we develop a hair shape modeling method based on a fluid flow.

In this chapter, we present a novel virtual hairstyling method. The method draws on similarities between static hair shape and snapshots of fluid flow around an obstacle. Accordingly, the hair shape is modeled as streamlines of a fluid flow. The model offers an ability to control "overall" hair shape around the head. At the same time, it gives a possibility of modeling rich "details" such as waves and curls. Moreover, the continuum property of fluid flow gives a sound heuristics for modeling complex hair-hair interaction while hairstyling. The model leads to the development of a fast, intuitive and easy-to-use hair styler. A designer can create intricate hairstyles quickly, without worrying about hair-body and hair-hair interactions. However, as it will be discussed in the concluding section, one needs to develop an insight into the use of the modeling paradigm. The following section dwells on the similarities of hair shape and fluid flow. We choose *ideal flow* as a fluid model and in Section 2, we develop the hair shape modeling method based on ideal flow. Section 3 describes the implementation issues and extends the methodology to a practical hair shape modeler. Fluid definition being volumetric in nature, we can use many of the volumetric texture techniques to add realism. Section 4 discusses about enhancing realism.

1 Hair and Streamlines of Fluid Flow

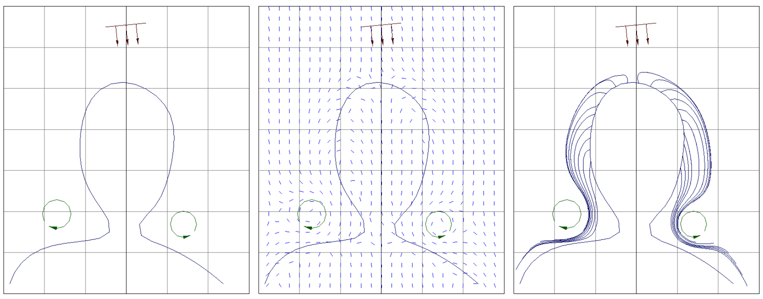


Figure 3: Modeling Hair as Streamlines of a Fluid Flow

Hair has many properties that are similar to a fluid flow. The comparison is only limited to static hair shape with the snapshot of a fluid flow, rather than comparing their dynamics. Figure 3 illustrates how we can think of hair as selective streamlines of a well set up fluid flow. Probably the first thing to notice is that the long hair has an overall tendency to go down under gravitational influence. This is similar to the *free stream flow condition* [Anderson 1991] – flow away from an obstacle has a constant velocity, downward in this case.

The hair-hair interaction is too prominent an issue to be ignored. Consequently, though hair is thin in geometry, collectively it has a volume. Hair strands tend to keep themselves apart and almost keep themselves parallel with respect to their neighbors. The hair-hair interaction is very similar to the *continuum property* of fluid – no two streamlines of a fluid flow intersect, however close they may get, with an exception of flow singularities. This observation is significant and gives sound heuristics to hair-hair interaction phenomenon.

Further, hair-body collision avoidance is the same as a flow around an obstacle. Flow component normal to an obstacle boundary is zero as the flow does not penetrate the obstacle. As depicted in Figure 3, near the neck and the shoulder, the hair follows a similar pattern. It is tangential to the body except from where it originates, e.g. on the scalp.

Hair strand tends to stand out on the scalp as it originates. This situation is different from the hair-body collision avoidance. In the flow context, we can visualize it as a secondary flow forcefully oozing out from within an obstacle boundary that is having fine holes. So the velocity component normal to the boundary is non-zero and constant. Greater the normal velocity component, more the streamlines will stand out.

Apart from this, hair shape has some other similarities with fluid flow such as hair curls being analogous to vorticity and hair waviness being analogous to the turbulence in the flow. Even the hair parting line appears as the *flow stagnation point* [Anderson 1991] in a flow around an obstacle. Observe in Figure 3 how the streamlines follow opposite directions at the stagnation point that is the top of the scalp.

In spite of the similarities discussed above, we would stress that, hair is not precisely like a fluid flow. For example, hair tends to form clumps, which is not true in the case of flow streamlines. In fact, some hair care products try to bring more streamline like order to one's hair, by reducing the clumpiness. Furthermore, though overall hair shape follows a fluid flow like structure, individual hair strand has perturbations as they "flow" and may even completely break away from the flow to become a stray hair strand. Section 5 discusses these issues in more details. First, in the next section, we develop a hair shape model based on the observations made so far.

2 Hair Shape Model

Choice of the fluid model for the purpose of modeling hair can be debatable. As we try to equate only a snapshot of fluid flow to hair, the natural choice would be to consider a stable flow. Stable flow is the flow that does not change over time. A good example of this kind of flow is the flow around a thin body in a very stable wind tunnel. We further choose the flow to be *ideal flow* [Wejchert and Haumann 1991] which is inviscid, hence stable, irrotational and incompressible. We find that this model is simple, fast and adequate. The ideal flow is governed by the following Laplace equation

$$\begin{aligned} \nabla^2 \Phi &= 0 \\ \vec{V} &= \vec{\nabla} \Phi \end{aligned} \tag{1}$$

where $\vec{V}$ is the velocity of the flow, a gradient of a scalar potential Φ.

As the equation 1 is a linear partial differential equation, the sum of the particular solutions is also a solution of the equation. There are a few interesting particular solutions that we would like to utilize.

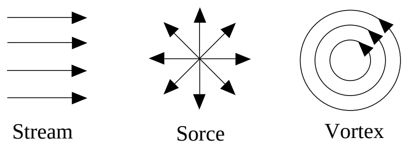


Figure 4: Ideal Flow Elements

Figure 4 illustrates these solutions of Laplace equation – the *ideal flow elements.* We briefly discuss them, and give an analytic formula for the velocity field induced by each element. For the detailed discussion on the ideal flow elements we refer to [Anderson 1991; Wejchert and Haumann 1991].

2.1 Stream

The velocity $\vec{V}_{stream}$ at any point $\bar{p}$ due to a *stream* is given by

$$\vec{V}_{stream}(\bar{p}) = \Gamma \hat{r} \tag{2}$$

where Γ is the strength of the stream and $\hat{r}$ is the unit vector in the direction of the stream. Thus the stream induces a constant velocity throughout the flow.

2.2 Source

The velocity $\vec{V}_{source}$ at any point $\bar{p}$ due to a *source* of strength Γ is given by

$$\vec{V}_{source}(\bar{p}) = \Gamma \frac{\hat{r}}{4\pi r^2} \tag{3}$$

where $\hat{r}$ is unit vector in the direction from the source location to the point $\bar{p}$. r is the distance between the source location and the point $\bar{p}$. Thus, the magnitude of the velocity follows an inverse square law and is always pointed away from the position of the source. The source essentially emits the flow from its position, whereas the negative source, also called as *sink*, sucks the flow towards its position point.

2.3 Vortex

A *vortex* swirls the flow around its position. The vortex for our purpose is actually a 3D vortex filament defined by the points $\bar{p}_1$ and $\bar{p}_2$ as shown in Figure 5.

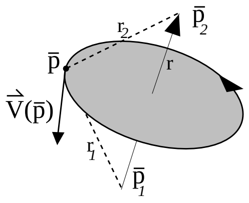


Figure 5: 3D Vortex Filament

The velocity $\vec{V}_{vortex}$ induced at point $\bar{p}$ by such a vortex filament [Ling et al. 1996] is given by

$$\begin{aligned} \vec{r}_1 &= \bar{p} - \bar{p}_1 \\ \vec{r}_2 &= \bar{p} - \bar{p}_2 \\ \vec{r}_0 &= \bar{p}_2 - \bar{p}_1 \end{aligned}$$

$$\vec{V}_{vortex}(\bar{p}) = \Gamma \frac{(\vec{r}_0 \cdot \vec{r}_1 / \, |\vec{r}_1| \, - \vec{r}_0 \cdot \vec{r}_2 / \, |\vec{r}_2|)}{4\pi \, |\vec{r}_1 \times \vec{r}_2|^2} \, \vec{r}_1 \times \vec{r}_2 \tag{4}$$

where Γ is the strength of the vortex.

There are few other interesting ideal flow elements such as source-sink pair i.e. *doublet* and *horse-shoe vortex*. However, these can be realized by using combination of the basic ideal flow elements.

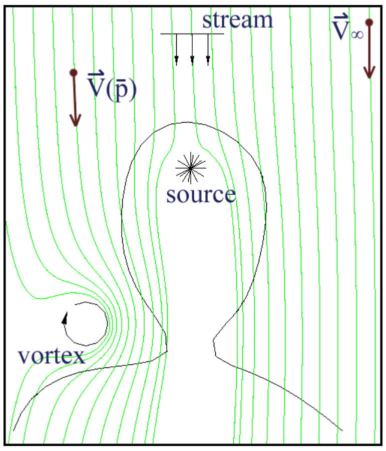


Figure 6: Linear Combination of Ideal Flow Elements

Figure 6 illustrates a typical complex flow by combination of the flow elements. Let the velocity induced at a point $\bar{p}$ in the flow by a user defined ideal flow element be $\vec{V}_{user}(\bar{p})$. Thus the linear combination of flow elements is given by

$$\vec{V}(\bar{p}) = \sum \vec{V}_{user}(\bar{p}) \tag{5}$$

We propose to model hair as streamlines of such a complex flow setup. Though there are numerous possibilities for the user to combine these elements, in order to get close to a hair shape we would design a flow setup taking the observations stated in Section 1 into consideration. First we put a *stream* to give the flow an overall downward direction.Then we may place a few other elements such as a *source* and a *vortex* as shown in figure 3. Note that the hair-hair interaction, is intrinsic to the fluid flow property hence it is always ensured. The streamlines are almost parallel and they never cross each other.

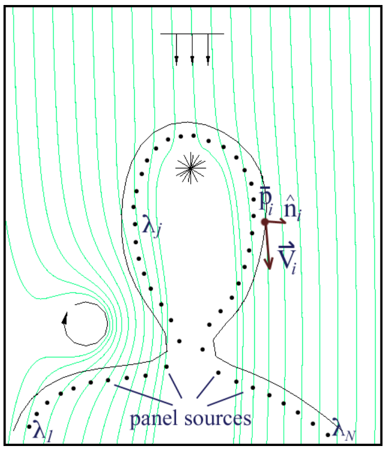


Figure 7: Placing Number of Sources under the Boundary

As depicted in Figure 7, a source in the flow has a property of generating an obstacle to the flow. Still, as shown in the figure, the streamlines penetrates the body. Inserting few more sources into the flow to achieve the obstacle avoidance is arbitrarily difficult. Instead, we setup a small number of sources along the the boundary to ensure that the flow does not penetrate the body. We find this setup to be adequate for the obstacle avoidance and allows fast interaction with the flow. Consider the obstacle, the body, placed in the flow field as shown in Figure 7. Let us approximate the boundary by descritizing it into a number of planar segments, namely *panels*. There are many variations of the method based on the choice of flow elements to be placed at each panel center. We choose simple *source panel method* [Anderson 1991; Ling et al. 1996]. We place one source per panel, slightly away from the center of the panel along the inward normal. Note that the source has a singularity at its center. The user can vary the *panel source offset* to achieve desired numerical accuracy. If the user places the source far away from the panel, the obstacle avoidance will not be accurate. However, if the user place the panel too close to the panel, she will induces large numerical inaccuracy because of the flow singularity. Also the gap between the two sources will allow some of the streamlines to escape into the boundary.

The panel source strengths are unknown. For N such panels, let the source strengths be $\lambda_1, \lambda_2, \ldots, \lambda_N$. We incorporate the panel sources into the equation 5.

$$\vec{V}(\bar{p}) = \sum \vec{V}_{user}(\bar{p}) + \sum_{j=1}^{N} \lambda_j \vec{S}_j(\bar{p}) \tag{6}$$

where $\vec{S}_j(\bar{p})$ is velocity induced by a unit source positioned near the panel j and is given by Equation 3. In order to solve for these unknown strengths, let us sample the flow at N panel centers and set up a boundary condition. The velocity at the i^{th} panel center $\bar{p}_i$ is

$$\vec{V}(\bar{p}_i) = \sum \vec{V}_{user}(\bar{p}_i) + \sum_{j=1}^{N} \lambda_j \vec{S}_j(\bar{p}_i) \tag{7}$$

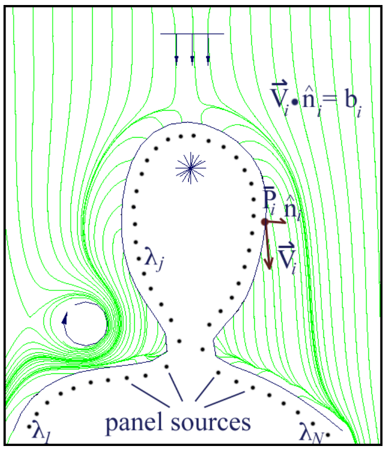


Figure 8: Source Panel Method for Obstacle Avoidance

For an inviscid flow, the velocity at the boundary can be non-zero. However, the velocity vector must be tangent to the surface, as the flow cannot penetrate the boundary surface. This ensures that the streamline will be parallel to the boundary surface. This is the typical situation of long hair rolling over the neck and shoulders. However, as stated in Section 1, we would also like to accommodate the cases where the streamline should ooze out from the boundary, just like the hair originating from the scalp. Both the cases are addressed by stating the wall boundary condition in terms of the velocity component normal to the boundary. If $\hat{n}_i$ is unit normal to the boundary surface at panel center i the wall boundary condition is

$$\vec{V}(\bar{p}_i) \cdot \hat{n}_i = b_i$$

$$(\sum \vec{V}_{user}(\bar{p}_i) + \sum_{j=1}^{N} \lambda_j \vec{S}_j(\bar{p}_i)) \cdot \hat{n}_i = b_i$$

$$\sum_{j=1}^{N}((\vec{S}_j(\bar{p}_i) \cdot \hat{n}_i)\lambda_j) = b_i - (\sum \vec{V}_{user}(\bar{p}_i)) \cdot \hat{n}_i \tag{8}$$

$$\sum_{j=1}^{N} S(i,j)\lambda_j = b_i - (\sum \vec{V}_{user}(\bar{p}_i)) \cdot \hat{n}_i \tag{9}$$

where b_i is the user specified magnitude of the normal velocity.

In Equation 9, the quantity $\vec{S}_j(\bar{p}_i) \cdot \hat{n}_i$ represents the velocity induced by a unit panel source near j^{th} panel on the center of the i^{th} panel. This is constant for a given geometry and a particular setup of the panel sources for that geometry. We call it the *flow boundary form* and denote it by $S(i, j)$. To solve the system of linear equation (Equation 9), in N unknowns (λ_j), we use LU decomposition followed by a back substitution. For a given constant flow boundary form matrix and its LU decomposition the computation takes of the order of minute. However, once the LU decomposed boundary form matrix is computed, one can quickly get a set of λ_j for the boundary condition b_i and variation of the complex user flow setup by back-substitution. This is particularly important for fast user interaction.

Figure 8 shows how the streamlines avoid the obstacle after setting appropriate panel source strengths. There are new streamlines standing out from the scalp as a result of the non-zero normal velocity specified on the scalp region. Elsewhere the streamlines go parallel to the body. By selecting the appropriate streamlines (the ones which ooze out from the scalp) we thus have a hairstyle, as shown in Figure 9.

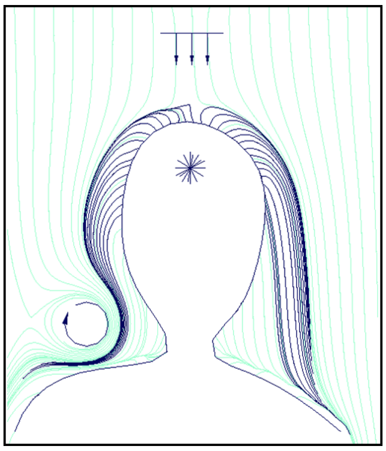


Figure 9: Hair as Streamlines of Fluid Flow

We carry out hair rendering by drawing individual hair strands. Even the dynamic hair model, discussed in the chapter – "Modeling Dynamic Hair as a Continuum", is based on the explicit hair model. This requires that we compute a large number of streamlines. We approximate a streamline by a polyline and we compute it by space marching the velocity field using small constant steps. Typically, we use 20,000 to 50,000 hair strands. Observing Equation 7, computation of velocity at each point, as we compute the streamline, involves the contribution of a large number of panel sources. We typically use 1000 to 2000 panels. This is numerically expensive considering a few ten thousand hair strands to be computed. To address this problem, we use subdivision scheme for fluid flow introduced recently [Weimer and Warren 1999]. The user defines a basic flow resolution as shown in Figure 10. We compute the flow field at each point of the coarse grid. We then use the subdivision flow in order to compute the flow at any point in the field. Thus we have reduced the computation of the flow at any point, resulting from a large fluid elements, to the computation of contributions by only few neighboring points on the coarse grid. For the detailed discussion on the subdivision flow we refer to [Weimer and Warren 1999].

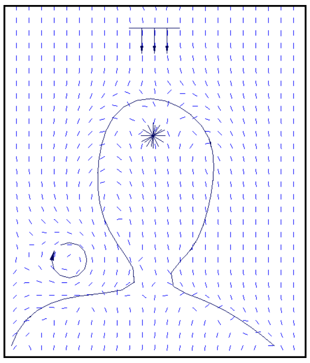


Figure 10: Subdivision Scheme for Ideal Flow

3 Interactive Hair Styler

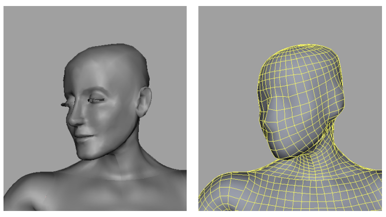


Figure 11: Polygon Reduced Geometry to Define Panels

In this section, we describe an interactive hair styler based on the theme "hair as a fluid flow". Here we only give a brief overview of the user interface to put the theory in perspective. For the complete details of the implementation, we refer to [Hadap 2003].

As described in Section 2, the flow computation with an boundary is done by the source panel method, which involves solving of a large, fully dense system of linear equations. To keep the computation within acceptable limits, the user starts modeling hair on a coarse mesh model as shown in Figure 11. It should be noted that the choice of a structured mesh is not a requirement for the method.

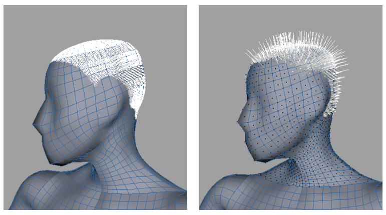


Figure 12: Hair Growth Map and Normal Velocity Map

The user then defines the overall hair density and paints the *hair growth map* directly on the model, as shown in Figure 12. The hair growth values can be varied smoothly by using pressure on the pressure sensitive stylus. From the density function in the form of the hair growth map, the placement of individual hair strand is pre-computed. In the white regions of the growth map, the hair number density will be equal to the overall hair density specified by the user. Whereas, in the gray regions it will be somewhat lower. In the black regions, there will be no hair. Once the hair growth map is defined, we use the Poisson-disc distribution to populate the hair roots. The Poisson-disk distribution is a random pattern that satisfies the Poisson-disk criterion – no two samples are closer together than some distance r_p.

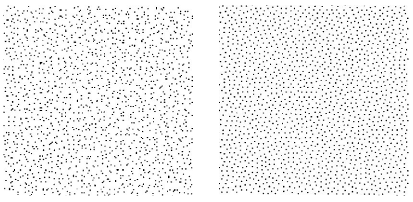


Figure 13: Distributing Hair on the scalp

Figure 13 shows a random distribution on the left. The random distribution results into a non-uniform placement of hair roots, where some hair roots get too close to each. Whereas, the Poisson-disc distribution, shown on the right in Figure 13 eliminates this defect by maintaining a minimum distance between any two hair roots, thus giving a more regular distribution. This distribution is close to what is observed on real scalp. In order to achieve the Poisson-disk distribution we first create a random distribution according to the hair growth map. Then we use a relaxation strategy described in the classic work by Don Mitchell's [Mitchell 1987] to smooth out the ill-placed hair roots. For further practical details on achieving Poisson-disk Distribution we refer to [Glassner 1995]. We then populate streamlines emerging from these hair roots to realize hairstyle. This will be discussed subsequently.

In order to place the sources corresponding to each panel, the user defines the panel source offset along the inward normal of the panel. Figure 12 shows the typical 3D configuration of panel sources. The user can finalize the source panel setup after visual examination. She has to take care that no two panel sources come close to each other to avoid the singularity. Upon acceptance, the flow boundary form is computed along with its LU decomposition as explained in Section 2. This typically takes around a minute for one thousand panels (on Pentium II 400MHz, 512MB RAM). As the geometry of the face remains constant throughout the hairstyling, this needs to be computed only once as long as the source panel setup does not change.

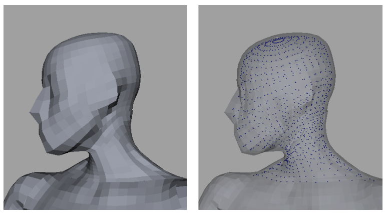


Figure 14: Placement of Panel Sources

To define the boundary condition of the flow described in Section 2, the user now paints the *normal velocity map*. During this, the hair growth map is also presented for visual feedback along with the display of normal velocity map being painted. The user will define zero normal velocity everywhere except for the scalp regions, from where the hair originates. She can control how the hair will stand out from the scalp by varying the magnitude of the normal velocity. Figure 14 shows the painted normal velocity map. The user even can paint negative velocity map, typically in the neck region. This would make hair curl-in towards the neck.

Further, the user is provided with 3D ideal flow elements – streams, sources and vortices. She can interactively place them around the model to define a hairstyle. After the user places a stream (the first fluid element in the hairstyle) in a downward direction to define the overall direction of the hair, a few hair strands are displayed for the interaction as shown in Figure 15a. The user can adjust the overall length of the hair. One can use a source from the toolbox and place it in front of the face to turn the hair away from the face as shown in Figure 15b. The other alternative is to trim hair on the face by using a trimming tool as shown in Figure 15c.

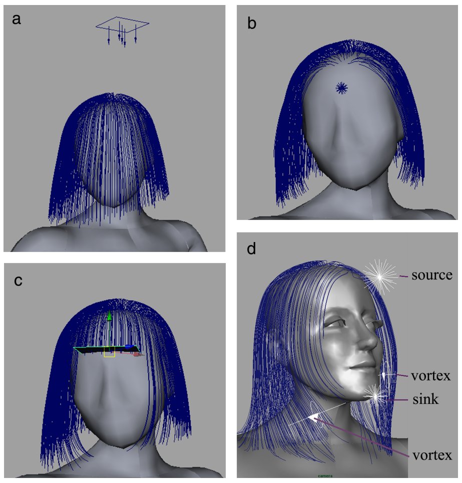


Figure 15: Simple Hairstyles using few Fluid Elements

The overall hair length along with the *hair length map* defines the length of individual hair strands. For deciding the hair length map, which defines the length of each streamline, the user is provided with a trim tool. Though the definition of the hair length map is similar to that of the hair growth map, it is not intuitive to paint the length map on the scalp as in the case of the hair growth map. Instead, the user takes any polygonal geometry, after adjusting it appropriately cutting across the displayed hair, she can trim the hair. The hair length map is recalculated by tracing back the streamlines, from the point of intersection with the trim tool.

The user can then place a few other fluid elements, mostly vortices, to add further details to the hairstyle. For every change in the flow setup and for the change in boundary condition, the source panel strengths are evaluated using the LU decomposed flow form factor. The effects of placing or moving the flow elements are computed in about a second. The computation speed does not permit the user to interact with the flow in real -time. However, this interactivity and only a few flow elements are sufficient to model a complex hairstyle. Figure 15d illustrates a hairstyle which is designed using only 5 fluid elements, 1 stream (not in the frame), 2 sources and 2 vortices. The user can also adjust the boundary velocity map to change the hairstyle appearance. Figure 15c is typical example of using negative boundary velocity value to curl-in hair towards the neck.

We present some illustrative hairstyles. Figure 16 is a simple but very realistic hairstyle obtained merely by two flow elements – one stream in the downward direction and one source to turn away the hair from the face (not

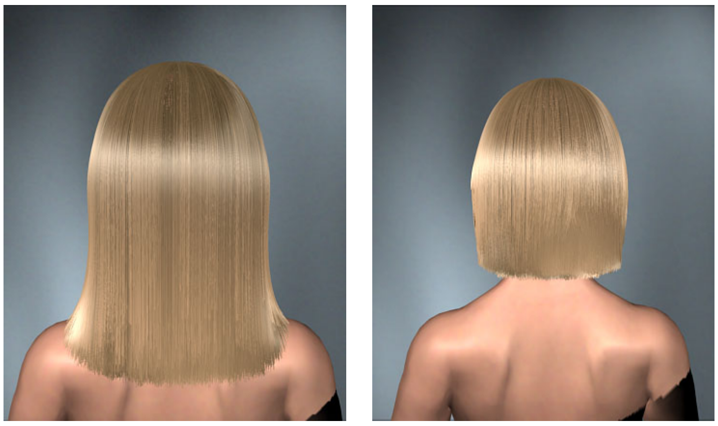


Figure 16: Hairstyles Involving with Simple Fluid Elements

visible). This set of hairstyle show the strength of the model in obtaining convincing hairstyle even with minimum effort.

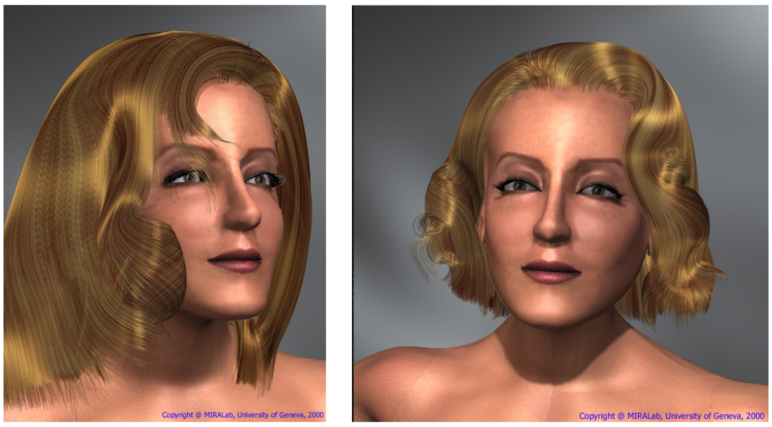

Copyright @ MIRALab, University of Geneva, 2000
Copyright @ MIRALab, University of Geneva, 2000

Figure 17: Hairstyles Involving Large Curls

Hairstyles in Figure 17 are result of setting up a complex ideal flow. Observe intricate large scale curls introduced by the vortices in the flow. Typically, the initial design of a basic hair style takes around an hour or two. It will take longer time to create a complex hairstyle depending upon the desired complexity.

4 Enhancing Realism

So far we have discussed the basic model for hairstyling using the fluid flow elements. As depicted by the hairstyle in Figures 16 and 17, the modeling of hair as streamlines of fluid flow followed by appropriate rendering scheme, gives results close to reality. Nevertheless, we try to break away from the theme and introduce variations inspired from the volumetric textures [Lewis 1989; Neyret 1998; Perlin and Hoffert 1989]. This will further add realism to hair. As explained in the Section 1, it is desirable to avoid the synthetic look of the hair, which is the result of a strict order of the individual streamlines. For this, we add an empirical model. The user defines a volumetric function such as noise and turbulence to define a volumetric perturbation to the fluid flow. A wide range of out-of-box volumetric textures such as fractals, waves, turbulence and noise are available for the purpose. However, we choose mainly two ways of adding perturbations to the synthetic uniformity – waviness and noise. In the case of waviness there is some underlying structure, whereas the noise is simply random but controlled variation. The straightforward way of achieving the perturbation is to define a function that adds the perturbation on top of the pre-computed individual hair strand shape obtained by the fluid flow setup. Following would be the list of parameters of such a function

1. The location where the hair strand originates from scalp

2. The length wise parameter of the hair strand from root to tip

3. The proximity of the hair strand with respect to its neighbors. After all, even though the perturbations are applied on per hair strand basis, there should be some inherent structure to the perturbations.

The added perturbation needs to have controllable inherent structure in spite of being random in nature. In the following of the section, we try to develop a well defined parametric space, which is followed by the perturbation scheme that is flexible and offers plenty of possibilities. Yet, the scheme will have only a small set of intuitive parameters. We begin by introducing the curvilinear coordinate system that is defined for each hair strand. We add the perturbations in the local coordinate space using the curvilinear coordinate system.

4.1 Curvilinear Coordinate System

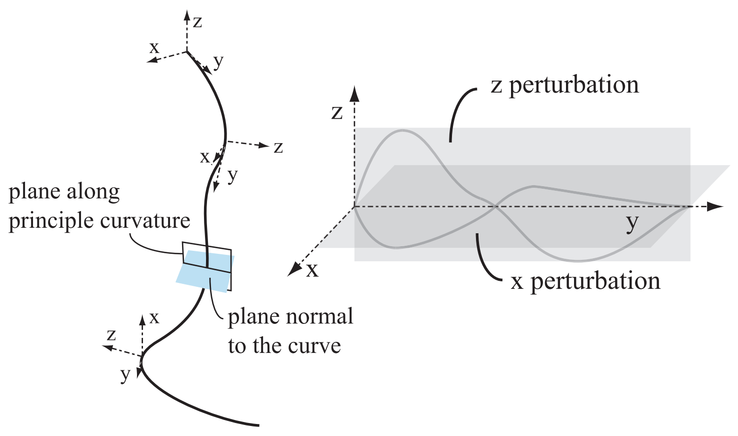


Figure 18: Curvilinear Coordinate System

The curvilinear coordinate system is illustrated in Figure 18. The y axis of the local coordinate system is aligned along the tangent of the streamline. The z axis is aligned along the principal normal. The principal normal is the vector formed by the intersection of the plane normal to the curve and the plane of the priciple curvature. The x axis is aligned to the bi-normal of the curve. Figure 18 illustrates how the coordinate frame orients as it moves from the root the tip. Once we establish the curvilinear coordinate system, it is possible to express the perturbations in the Cartesian coordinates along the x and z axis and apply it to the curve the hair strand shape as shown with Figure 18.

4.2 Root Variable Map

As described in the previous subsection, we apply the perturbation as a combination of the x perturbation and the z perturbation along the length. In order to obtain controlled variation (randomness) in the structure of waves as well as noise across the hairstyle, we introduce a scalar function *root variable map* defined on the scalp. The function is unit function defined over texture parameter space $[u, v]$ and is denoted by the following equation:

$$r = f(u, v) \tag{10}$$

The user defines the root variable map by painting a grayscale texture directly on the scalp in 3D as shown in Figure 19.

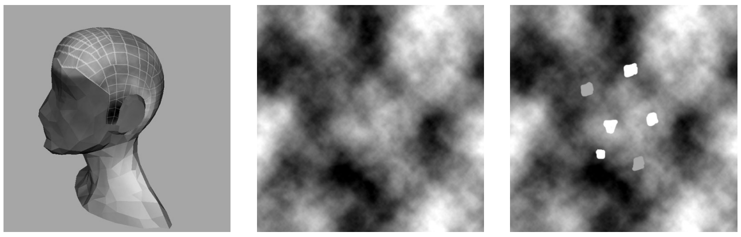


Figure 19: Root Variable Map

Alternatively, she can define it by editing the texture image in 2D using standard image editing software such as Photoshop. In most cases the user would define a smoothly varying random function. In the smoothly varying random function, the function value varies a lot for the distant $[u, v]$ points, whereas it does not vary that much for the points lying in the neighborhood. The smoothly varying random function is shown in the central figure of Figure 19, whereas the last figure illustrates some disjoint regions defined over the smoothly varying function to illustrate the flexibility. We would like to highlight that the root variable map thus defined in terms of an image rather than defined analytically, offers a lot of possibilities in defining the character of the volumetric perturbation. These gray scale values of the image affects various parameters of the wave and noise functions, which is defined subsequently. We also give a few illustrative examples of how a particular definition of the root variable map achieves a particular effect, such as overall waviness, clumping and stray hair strands.

4.3 Waviness

The waviness can be typically represented by a generalized sine wave, hence we chose the sine function as the basis of our waviness model. For a point p on the hair strand located at the normalized distance parameter l from the root, we define waviness in terms of sine wave function, randomly varying across the hairstyle:

$$\begin{aligned} W_x(l) &= A_x(l)\ sin(\Phi_x + T_x(l)\ l) \\ W_z(l) &= A_z(l)\ sin(\Phi_z + T_z(l)\ l) \end{aligned} \tag{11}$$

The randomness in the wave function per hair strand is achieved by randomly defining the amplitude A, frequencey T and phase Φ.

In order to define how the amplitude varies along the length of the hair, the user defines the amplitude characteristics at four distinct points equally spaced from the $\frac{1}{4}^{th}$ position to the tip position. The amplitude of the wave $A_x(l)$, all along the length from root to tip, then gets defined by:

$$A_x(l) = CR(A_x^i, l) \tag{12}$$

where CR is a Catmull-Rom interpolation function defined through $A_x^i, i = 0, 1, 2, 3, 4$ with the length wise parameter $l \in [0, 1]$. The value of A_x^0 is set to zero as the perturbation is always zero at the hair strand root.

The user defines the amplitude characteristics A_x^i as distinct stochastic processes using Perlin's noise function [Perlin 1985] *random*, each defined at one of the four hair length positions. We deliberately name the function as *random* instead of traditional function name – *noise*, to avoid the confusion with the noisiness introduced to the hair strand subsequently. These four stochastic functions have a mean $A_{x,mean}$, a variance $A_{x,var}$ and a scale A_{scale} defined by the user. Thus the amplitude characteristics A_x^i are:

$$A_x^i(r) = A_{x,mean}^i + A_{x,var}^i\ random(r\ A_{x,scale}^i) \quad i = 1, 2, 3, 4 \tag{13}$$

The *random* function returns a value of the Perlin's noise function whose randomness is parameterized by its arguments – r times $A_{x,scale}^i$ in our case. A high value of $A_{x,scale}^i$ signifies white noise, whereas a low value attribute to a smoothly varying random function. As a special case, the value of the function is 0 at 0; $random(0) = 0$. $A_z(l)$ is defined in the same manner by user defining four A_z^i amplitude characteristics in terms of amplitude, phase and scale. Even, the same definition is used for the frequency T:

$$T_x^i(r) = T_{x,mean}^i + T_{x,var}^i\ random(T_{x,scale}^i\ \ r) \quad i = 1, 2, 3, 4 \tag{14}$$

The eight values $T_x^i(r)$ and $T_z^i(r)$ are defined in a similar manner, and $T_x(l)$ and $T_z(l)$ are interpolated values across them using Catmull-Rom interpolation. Unlike the amplitude and the frequency, the phase is defined only at the root of each hair strand:

$$\Phi_x = \Phi_{x,mean} + \Phi_{x,var}\ random(r\ \Phi_{x,scale})$$
$$\Phi_z = \Phi_{z,mean} + \Phi_{z,var}\ random(r\ \Phi_{z,scale}) \tag{15}$$

Notice the presence of r in all the above equations. This directly links the *root variable map* variation explained previously to the variation in amplitude, frequency and phase. The value of r changes from hair strand to hair strand according to the definition of the root variable map. Based on this variation, user can implicitly define how randomly the amplitude, the frequency at four distinct regions and the phase at the root gets defined for the individual wave.

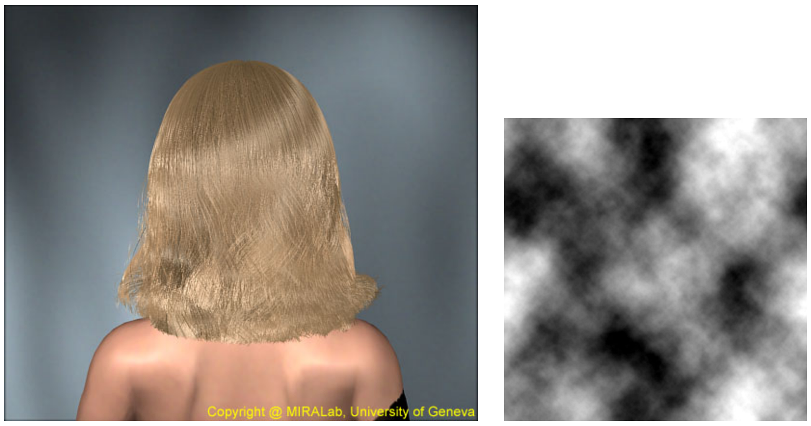


Figure 20: Hairstyle using Waviness

Figure 20 shows a hairstyle formed by setting up simple fluid elements followed by the waviness added by the method described in this section. The corresponding root variable map is also presented. Observe that the root variable map is smoothly varying, which results in to the wave structure smoothing varying across the hair strands. Also observe that the overall amplitude and frequency characteristic changes from root to tip. The waves has more amplitude as well as frequency towards the tip.

Figure 21 illustrates the wave model. As discussed earlier, the mean, the variance and the scale values at four different positions for both x and z directions gets defined by the user, along with two initial phases. The generated wave for the hair strand thus will be confined to the amplitude envelop formed by the randomly generated values having the above defined characteristics, as shown in the figure.

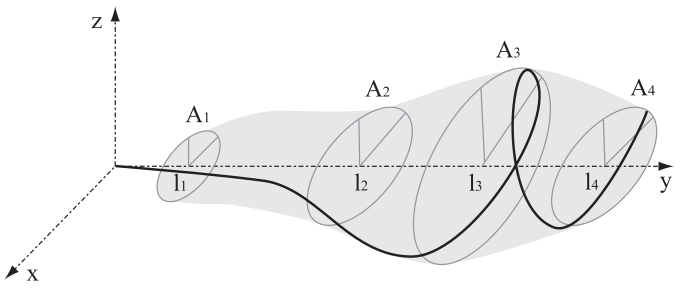


Figure 21: Hairstyle with Waviness

One can use the same wave model to generate curls. In order to generate curl, a helix, a phase difference of $\frac{\pi}{4}$ should be maintained between Φ_x and Φ_z. In the illustrative example, Figure 21, thus the generated helix is confined to the amplitude envelop. Notice that the frequency increases towards the tip of the hair. All these characteristics are randomly varied across the hair strands.

4.4 Noise

Noise is similar to waviness. Instead of the generalized sine wave perturbation applied to hair strand, in the case of noise we use Perlin's *noise* function along the length parameter l.

$$
\begin{aligned}
N_x(l) &= A_x(l)\ noise(\Phi_x + T_x(l)\ l) \\
N_z(l) &= A_z(l)\ noise(\Phi_z + T_z(l)\ l)
\end{aligned} \tag{16}
$$

Compare this equation to Equation 11. The amplitude, period and phase gets defined in exactly same way as explained in the previous section but with different set of amplitude, frequency characteristics. Even, the overall variation is related to a separate root variable map meant specifically for the noise perturbations. That way the user can have separate control over the waviness and noise and can mix them for desired effect.

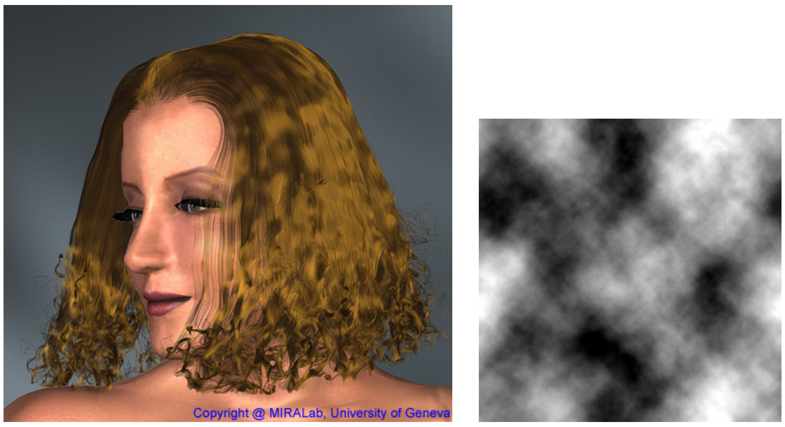


Figure 22: Hairstyle using Noise

Figure 22 shows a hairstyles formed by setting up simple fluid elements followed by the application of noise perturbation. The corresponding root variable map shows the smoothly varying random function. This results in individual hair following random directions, however while maintaining overall structure. Observe that the amplitude of the noise increases from root to tip. The increase in the frequency, unlike in the case of waviness, is manifested in change in the quality of the noise – towards white noise near tip.

4.5 Clumpiness

Fluid like hair may be a dream of everybody. Though in reality, under cohesive/adhesive forces, hair tend to form clumps – a set of hair strands get close to follow a common path. The clumpiness can be obtained in the same framework of the volumetric perturbation using the waviness or the noise or the both. The trick is to define distinct regions having constant value in the root variable map. Figure 23 illustrates such a root variable map.

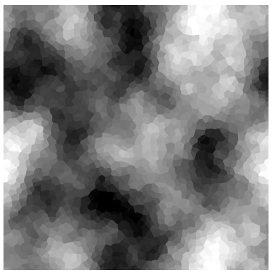


Figure 23: Map Definition for Clumpiness

The root variable map is in fact derived from the previously defined smoothly varying random map. However, observe that instead of defining a continuously varying map, the image defines distinct small regions or crystals having a constant values. The value varies from region to region in a smoothly varying random fashion. One can obtain such a map using out-of-box tools ("crystallize") supported by the image editing software such as Photoshop.

Effectively, we are creating a bunch of hair by means of having the same r value for the hair strands belonging to the same region. Subsequently, the volumetric perturbations applied to all these hair strands will be constant. The size of such regions can be controlled by the user depending upon the desired size of the clumps. Figure 24 shows an enlarged view of "crystallized" root variable map. Observe how a set of hair strand roots belong to a particular region.

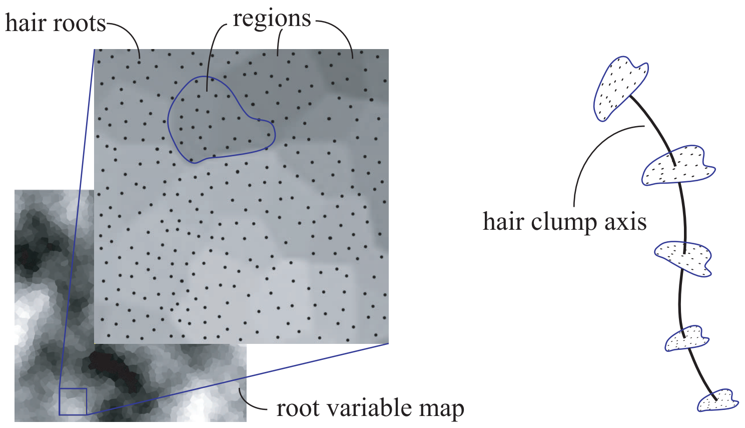


Figure 24: Forming Clumps

Unfortunately, just by defining bunch of hair strands to follow a common path via common volumetric perturbation would not suffice. The hair strands need to come close to each other as they flow from root to tip. For that purpose, we need to explicitly identify a bunch of hair strands. We carry out a Delaunay triangulation of the hair roots in the texture space $[u, v]$. Then we follow each edge of the triangulation to observe change in the root variable map. If the root variable map varies across the edge, we delete that edge. Once we have followed all the edges, we are left with closed regions in the triangulated mesh. If a particular region is too small this may result in only a single hair root resulting into a single hair strand belonging to the clump. In this case we are creating a stray hair strand as discussed in the next section. All the hair strands in one clump follow some what parallel path defined by the fluid flow. Further, exactly the same perturbation gets added on top of the flow for each of the hair strand belonging to the clump. This can be visualized as the small region being swept all along the length of the *hair clump axis* to form the clump. Figure 24 illustrates how the clump is formed by sweeping the small region along the hair clump axis. We elect the centroidal $[u, v]$ value as the root of the hair clump axis. Then we compute the geometry of only the hair clump axis by tracing the streamline in the fluid flow followed by application of the volumetric perturbation. Then we sweep the clump region from root to tip, in the process we scale the region to create shrinkage in the clump.

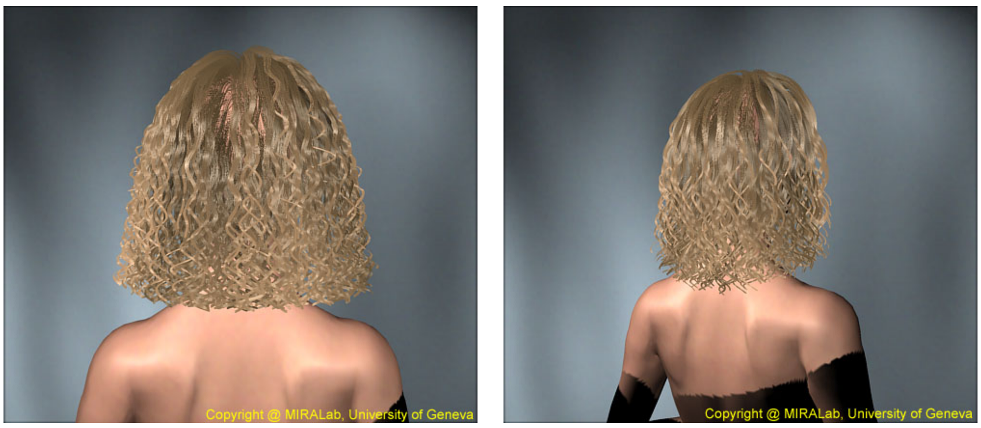

Copyright @ MIRALab, University of Geneva
Copyright @ MIRALab, University of Geneva

Figure 25: Hairstyle with Waviness and Clumpiness

Figure 25 shows such hairstyle as a result of simple fluid flow, waviness and clumpiness. The root variable map used is as depicted in Figure 23. The user defines the clump dilation characteristics across the hairstyle in the same manner by defining mean, variance and scale values for four different sections along the length. Observe that hair has controlled randomness in amplitudes, frequencies and the scalings in the clumps throughout the hairstyle.

4.6 Stray Hair

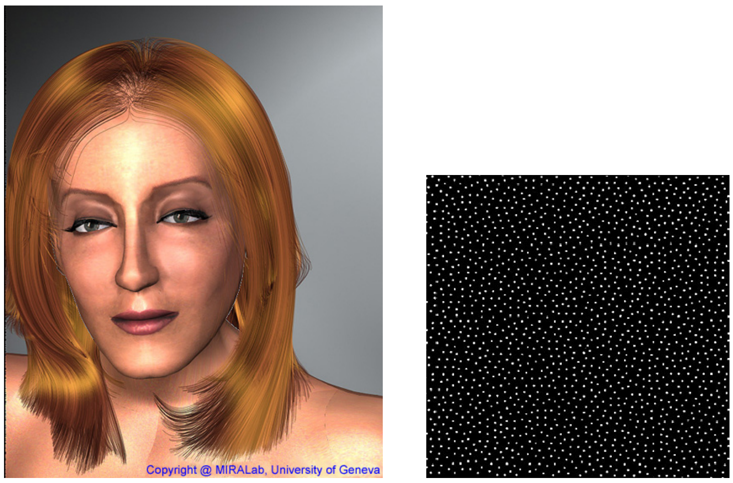


Figure 26: Individual Hair Breaking Away From Flow

So far, we have explained a structured perturbation in the hairstyle using a well-defined model of waviness and clumpiness. However, in the spirit of explicit hair model, we now demonstrate how to add a random noise in the hair structure. This simply means that the user is able to define certain individual hair strands that break away from the defined fluid flow and controlled volumetric perturbation, and follow a random path. Figure 26 demonstrates how few individual stray hair strands break away from overall flow and follow the random directions. The stray hair strands can be defined by changing the root variable map as shown in the figure. We simply add bright spots on the root variable map. In this example, the root variable map is uniformly dark, indicating no waviness or noise everywhere except at the texture positions of the white spots. As a result, the overall hairstyle is very smooth except for the few stray hair strands whose texture coordinates $[u, v]$ happen to fall on these white spots. We use mix of waviness and noise to achieve the random directions of the stray hair strands.

5 Hair Animation as Time-varying Fluid Flow

The dynamics of hair is different from the dynamics of a fluid flow. Therefore, for the purpose of hair animation, one might not be able to extend the static hair shape model based on a snapshot of fluid flow, to the dynamics of the flow. Nevertheless, in this section we would present an interesting possibility. We discuss how a time varying ideal flow setup can be used for reasonable animations of simple hair styles. The technique can be used only for simple hair styles and specific types of animations.

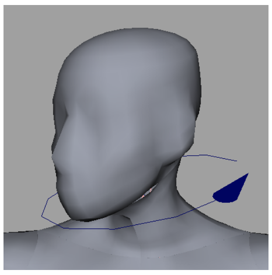


Figure 27: Time-varying Vortex around the Head

The ideal flow is stable. Thus for a given setup of ideal flow elements, the flow does not change over time. We use the quasi-steady property of the ideal flow for the animation of hair. We set up time varying fluid elements, which give (being an ideal flow) a time varying stable fluid. Figure 28 is an animation of a simple hairstyle with the head movement. The hairstyle is composed of only a stream in a downward direction with a set of panel sources beneath the head. As the head moves, the inertial forces of hair are simulated by setting up a time varying vortex around the head, as shown in Figure 27, to give hair a momentum. The strengths of these fluid elements are animated by key-framing at appropriate times. If there is a vertical head movement in the animation, one could have added even the bounce to hair by varying the strength of the stream which gives hair the downward direction.

In an another situation, consider hair blown by wind or hair under water. In this class of hair animation, the external forces such as wind or water dominates the dynamics of hair. In such situation, one can readily animate hair, hair being streamlines of fluid flow. The hair shape can be altered using time varying volumetric perturbations [Dischler 1999] as discussed in section 5. One can simulate the turbulent wind fields [Sakas 1993; Stam and Fiume 1993] as a stochastic process. These being linear in nature we can add the fields to the field defining hair shape to give a hair animation dominated by wind. The source panel method will ensure the collision avoidance of hair with body in any complex flow setup. During computing the animation, hair calculations for a frame depend only on the flow setup for that frame. This is a great merit of the method for parallel computation.

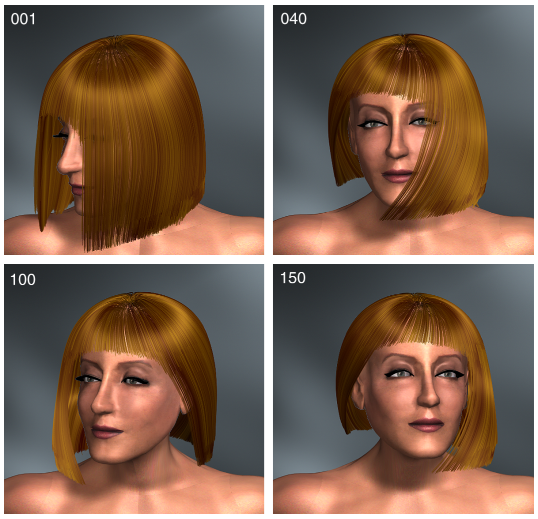


Figure 28: Hair Animation using Time-varying Ideal Flow

Soon we realized that, although interesting, the method has limitations. It can't represent accurately the stiffness dynamics and is limited as compared to numerous kind of motions hair undergo. In the chapter – "Modeling Dynamic Hair as a Continuum", we develop a detailed dynamic hair model which is essentially inspired from this method. However, the dynamic hair model has nothing in common with the fluid model presented for hairstyling.

6 Summary

In this chapter we have described a powerful hair shape modeling paradigm, which is implemented as a plugin to Maya – MIRAHairSimulation. This work is part of doctoral research [Hadap 2003] at MIRALab, University of Geneva. The highlights of the proposed method are as follows

- Hairstyling – We have developed a powerful interactive hair styler based on the theme – hair as streamlines of fluid flow. The shape modeling paradigm gives strong heuristics to gravity, hair stiffness, hair-hair interaction and hair-body collisions, which are otherwise very complex physical phenomena. As the user does not have to pay a special attention to these qualities, she can interactively shape complex and visually plausible hairstyles quickly and easily.

- Volumetric Textures – Although strong in adding rich local details, the volumetric textures were not controllable to achieve the desired global shape. Thus they were unsuitable for modeling of long styled hair, they are great for modeling of short hair like fur. The volumetric nature of the proposed shape modeling paradigm provides a structured way of adding overall shape control to the volumetric texture definition. We have merged the two approaches, which enables the user with immense possibilities of adding rich local volumetric details such as noise and turbulence, while she can still controll the overall shape.

Overall, MIRAHairSimulation proves to be a powerful system for achieving convincing hairstyles in the typical creative environment. Figures 16, 17, 17, 20, 22, 25 and 26 illustrate the effectiveness of the proposed hair shape modeling methodology based on fluid flow followed by volumetric perturbations, in creating a variety of hairstyles. We also give results from the usage of MIRAHairSimulation by the animators at MIRALab – University of Geneva. Figures 29 and 30 are hairstyles by Samuel Kong and Nedjma Kadi. It is inappropriate to quantify the time involved in creating a hairstyle as the time involved highly depends on the complexity of the hairstyle and desired precision. The typical efforts involved in designing hairstyle are moderate and it typically take 2-3 hours to define a complex hairstyle. Apart from the ease of the hairstyle definition, the examples clearly show the possible variety in the hairstyle shape. Albeit, the hairstyles does not involve knots and braids.

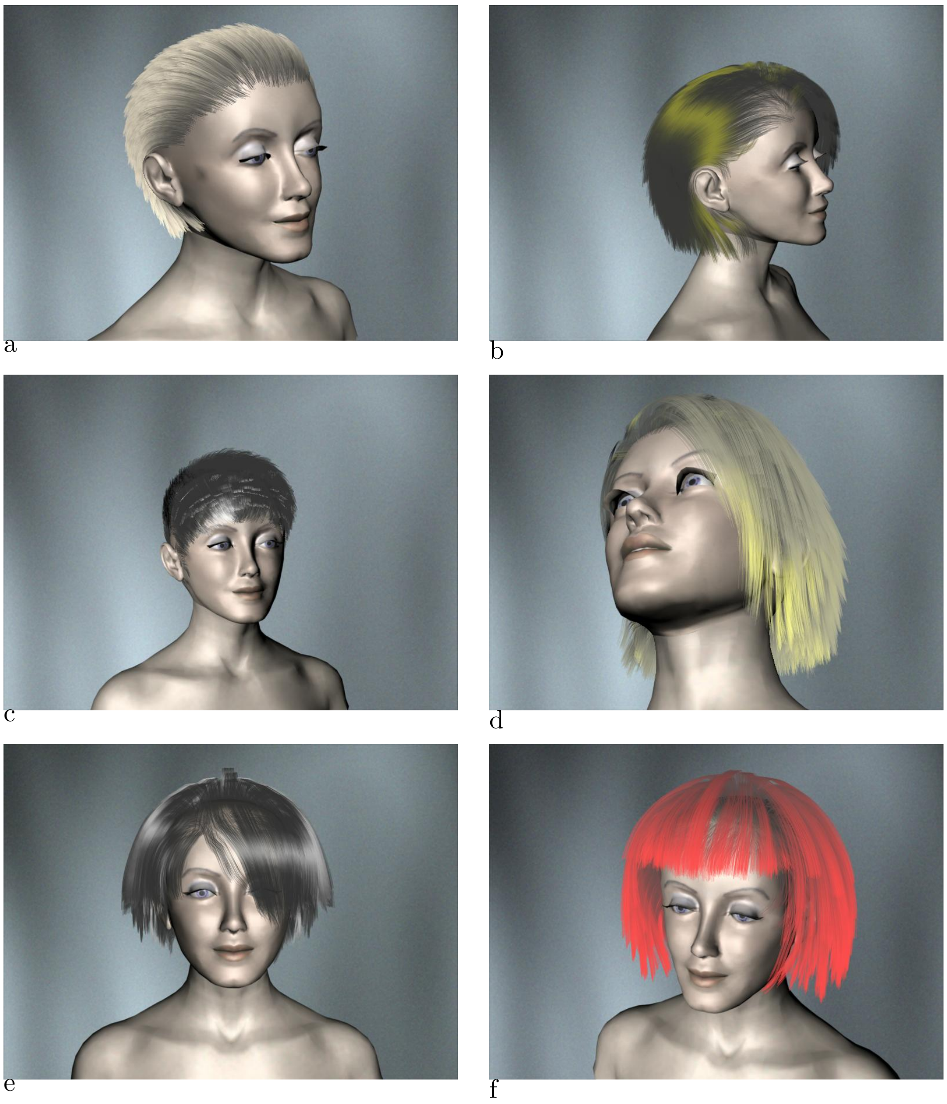
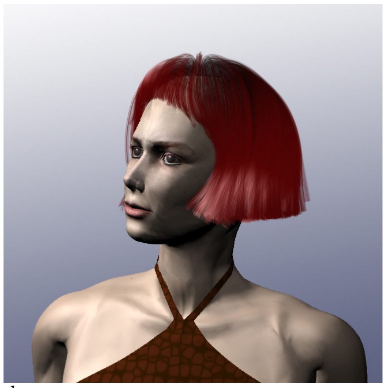


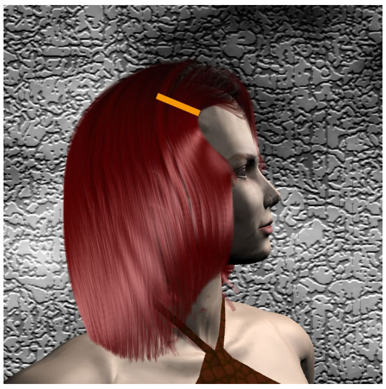
Figure 29: Hairstyles by the student Samuel Kong using MIRAHairSimulation

a

b

c


d


e

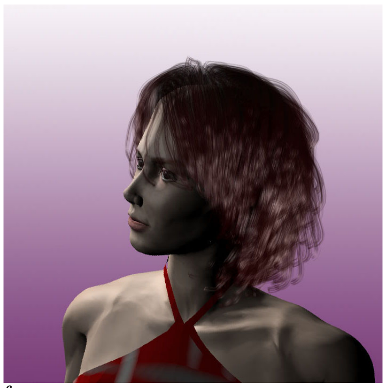

f

Figure 30: Hairstyles by the animator Nedjma Kadi using MIRAHairSimulation

7 Limitations and Future Work

- The shape modeling paradigm for hairstyling based on fluid flow is both powerful and enjoyable. However, it might take some time before the user fully grasps the possibilities of the tool. The user has to develop a special insight into the theme – hairstyle as streamlines of fluid flow.
- Although the editing of the fluid flow for the purpose of hairstyling is fast, it is not real-time. Depending on the flow complexity, the interactive flow update takes from 0.1 to 4 seconds. This interactivity is sufficient. However, faster interaction speeds due to ever increasing computing power, would definitely improve the overall grasp of the shape modeling paradigm.
- The tool does not provide the means of precisely crafting a hairstyle from a real example. The use of explicit hair models is more suited for this purpose. In the proposed methodology, fluid flow being a global phenomenon,

one can not introduce local shape changes without affecting the global shape. In using our method, the user can only capture the essence of the desired example hairstyle and then explore the wide range of possibilities within the limits.

To this end, we would like to explore the possibility of a computer vision based techniques to acquire the precise definition of the flow from a real photograph to automatically populate the hairstyle.

- The source panel method used for the boundary avoidance of fluid flow requires that the underlying mesh is smooth and regular. Currently the animators have to spend extra time to define the boundary mesh that approximates the detailed face geometry. This is a time consuming task. We would like to implement an appropriate mesh reduction scheme which gives very smooth and regular mesh.
- It is not possible to model wide range of clips, braids, knots and fashion accessories in hairstyle using the proposed methodology. The method only offers some primitive possibilities as shown in Figure 30c and 30e.

References

Anderson, J. D. 1991. *Fundamentals of Aerodynamics.* McGraw-Hill. ISBN 0-07-001679-8.

Anjyo, K., Usami, Y., and Kurihara, T. 1992. A simple method for extracting the natural beauty of hair. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques (SIGGRAPH)*, ACM SIGGRAPH.

Cabral, B., and Leedom, L. 1993. Imaging vector fields using line integral convolution. In *Proceedings of SIGGRAPH 93*, Computer Graphics Proceedings, Annual Conference Series, 263–272. ISBN 0-201-58889-7.

Chen, L., Saeyor, S., Dohi, H., and Ishizuka, M. 1999. A system of 3d hair style synthesis based on the wisp model. *The Visual Computer 15*, 4, 159–170.

Daldegan, A., Magnenat-Thalmann, N., Kurihara, T., and Thalmann, D. 1993. An integrated system for modeling, animating and rendering hair. *Computer Graphics Forum, Proceedings of Eurographics 12*, 3, 211–221.

Dischler, J.-M. 1999. A general model of animated shape perturbation. In *Proceedings of Graphics Interface*, 140–147.

Ebert, D., Carlson, W., and Parent, R. 1994. Solid spaces and inverse particle systems for controlling the animation of gases and fluids. *The Visual Computer 10*, 4, 179–190. ISSN 0178-2789.

Gelder, A. V., and Wilhelms, J. 1997. An interactive fur modeling technique. In *Proceedings of Graphics Interface.*

Glassner, A. 1995. *Principles of Digital Image Synthesis*, vol. 1. Morgan Kaufmann, ch. Sampling and Reconstruction Techniques, 427–430.

Goldman, D. 1997. Fake fur rendering. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques (SIGGRAPH)*, ACM SIGGRAPH, 127–134.

Hadap, S. 2003. *Hair Simulation.* PhD thesis, MIRALab, CUI, University of Geneva. No 3416, Science Faculty.

Kajiya, J., and Kay, T. 1989. Rendering fur with three-dimensional textures. In *Proceedings of the 16th annual conference on Computer graphics and interactive techniques (SIGGRAPH)*, ACM SIGGRAPH, 271–280.

LeBlanc, A., Turner, R., and Thalmann, D. 1991. Rendering hair using pixel blending and shadow buffers. *The Journal of Visualization and Computer Animation 2*, 3, 92–97.

Lewis, J.-P. 1989. Algorithms for solid noise synthesis. In *Proceedings of the 16th annual conference on Computer graphics and interactive techniques (SIGGRAPH)*, ACM SIGGRAPH, 263–270.

Ling, L., Damodaran, M., and Gay, R. K. L. 1996. Aerodynamic force models for animating cloth motion in air flow. *The Visual Computer 12*, 2, 84–104. ISSN 0178-2789.

Magnenat-Thalmann, N., Hadap, S., and Kalra, P. 2000. State of the art in hair simulation. In *Proceedings of International Workshop on Human Modeling and Animation*, Korea Computer Graphics Society, Seoul, Korea, 3–9.

Mitchell, D. P. 1987. Generating antialiased images at low sampling densities. In *Computer Graphics (Proceedings of SIGGRAPH 87)*, vol. 21, 65–72.

Neyret, F. 1998. Modeling animating and rendering complex scenes using volumetric textures. *IEEE Transactions on Visualization and Computer Graphics 1*, 4 (Jan-Mar).

Perlin, K., and Hoffert, E. M. 1989. Hypertexture. In *Proceedings of the 16th annual conference on Computer graphics and interactive techniques (SIGGRAPH)*, ACM SIGGRAPH, 253–262.

Perlin, K. 1985. An image synthesizer. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques (SIGGRAPH)*, ACM SIGGRAPH, 287–296.

Reeves, W. T. 1983. Particle systems - a technique for modeling a class of fuzzy objects. *ACM Transactions on Graphics 2*, 2 (April), 91–108.

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 (October-December), 141–148.

Rudiger, M., Samson, R. V., and Samson, R. V. 1998. *388 Great Hairstyles.* Sterling Publications.

Sakas, G. 1993. Modelling and animating turbulent gaseous phenomena using spectral synthesis. *The Visual Computer 9*, 4.

Stam, J., and Fiume, E. 1993. Turbulent wind fields for gaseous phenomena. In *Computer Graphics (Proceedings of SIGGRAPH 93)*, vol. 27, 369–376.

Watanabe, Y., and Suenaga, Y. 1989. Drawing human hair using wisp model. In *Proceedings of Computer Graphics International*, Springer-Verlag, 691–700.

Watanabe, Y., and Suenaga, Y. 1992. A trigonal prism-based method for hair image generation. *IEEE Computer Graphics and Application 12*, 1 (January), 47–53.

Weimer, H., and Warren, J. 1999. Subdivision schemes for fluid flow. In *Proceedings of SIGGRAPH 99*, ACM SIGGRAPH / Addison Wesley Longman, Los Angeles, California, Computer Graphics Proceedings, Annual Conference Series, 111–120. ISBN 0-20148-560-5.

Wejchert, J., and Haumann, D. 1991. Animation aerodynamics. In *Computer Graphics (Proceedings of SIGGRAPH 91)*, vol. 25, 19–22. ISBN 0-201-56291-X.

Yang, X. D., Xu, Z., Yang, J., and Wang, T. 1999. The cluster hair model. *Journal of Graphics Models and Image Processing*.

Zockler, M., Stalling, D., and Hege, H.-C. 1996. Interactive visualization of 3d-vector fields using illuminated streamlines. In *Proceedings of IEEE Visualization*, 107–144.

Modeling Dynamic Hair as a Continuum

Sunil Hadap[*] Nadia Magnenat-Thalmann[†]
R&D Staff, PDI/DreamWorks MIRALab, University of Geneva

In computer graphics, there are numerous novel models developed for animation of synthetic and natural objects, animals, and virtual humans. Many of the models do not reflect the physical reality, but the mere visual resemblance. For example, a digital actor may not walk in accordance with the accurate dynamics of the body, rather she will follow the footsteps "key-framed" by the animator. In this particular case, the underlying animation model may very well be complex. However, it does not reflect the reality. One of the highlights of such an approach is – it leaves the animator with a complete control of the result. On the other hand, when it comes to animation of fluids, explosions, solid fracture and hair, computer graphics has opted for "direct numerical simulation". These models tend to be more and more physically based. Here the animators are not given the explicit control over the result. The control is only cursory and by means of setting up boundary conditions and defining external force fields. However, the models being accurate, they produce very convincing results. Our approach to hair animation is in the same spirit. For us, the cloth simulation system at MIRALab – University of Geneva and its approach is very inspirational to this regard.

We have discussed the state-of-the-art in hair simulation in [Hadap 2003; Magnenat-Thalmann et al. 2000]. We have identified the difficulties in hair dynamics along with previous attempts, their advantages and limitations. To start with, we take only a quick recap. In this chapter, our focus is mainly the dynamics of long styled hair, as against fur. In this regard, the explicit hair models (or the wisp models) are most effective. In these models, each and every hair (or wisp) is explicitly considered for the dynamics. This makes these models intuitive and close to reality. However, the shape intricacies, the thin geometry and the relatively high stiffness along with the high degree of damping make dynamics of individual hair strand difficult. Further more, the shear number of hair strands demands very careful balance between adequate details in the elastic models and the numerical complexity. Anjyo *et al* [Anjyo et al. 1992] pioneered and Lee *et al* [Lee and Ko 2001] developed on their work in which they diligently model hair inertial and stiffness dynamics as projective two dimensional cantilever dynamics. In our opinion there is plenty of scope for more complex models considering the current computing power. On the other hand, Rosenblum *et al* [Rosenblum et al. 1991] used a mass-spring-hinge model to control the position and the orientation of the hair strand. Unlike the cantilever models, the spring-mass-hinge models are truly three dimensional. However, they give rise to stiff differential equations of motion. They are also inappropriate to model the dynamics of non-straight neutral shape of hair along with the twisting motion. Many of these attempts used approximate collision detection. They replace the head and body geometry by simple analytic shapes such as ellipsoids. We strongly feel that recent developments in real-time computer graphics with regard to collision detection and response certainly facilitate accurate hair-body collision handling. Appreciably, none of the previous attempts considered hair-hair and hair-air interactions - until very recently. Work by Plante *et al* [Plante et al. 2001] and Chang *et al* [Chang et al. 2002] addressed the problem by considering only interaction between wisps. In these novel propositions, the configuration of the wisps remains rather constant and hair does not break away from one wisp and join another.

In recent years the computing power has grown many times. Supercomputing power of the past is becoming increasingly available to the animator's workstations. There is a need to develop new hair dynamics models in light of current and future computing advances. In this chapter, we hope to have developed enough "food for computing" by attempting hair-hair and hair-air interaction with elaborate elastic dynamics of individual hair stand. While making a paradigm shift, to model hair-hair and hair-air interactions, we propose to consider hair as continuum. Subsequently, we treat the hair-hair interaction dynamics and hair-air interaction dynamics to be fluid dynamics. This proves to be a strong as well as viable approach for an otherwise very complex phenomenon. We use smoothed particle hydrodynamics (SPH) as the numerical model. However, for the realization of the shape memory and the rendering, we still need to retain the notion of individual hair strand. In that regard, we develop an elaborate stiffness and inertial dynamics of the individual hair strand. We treat it as a serial rigid multi-body chain. This being a reduced coordinate formulation, the stiffness dynamics is numerically stable and fast. Finally, we unify the continuum interaction dynamics and the stiffness dynamics to realize a strong hair animation framework.

The outline of the chapter is as follows. In the next section, we develop the basic continuum hair model. Realizing the need to retain the individual character of hair, Section 2 gives a detailed model of stiffness dynamics for single

[*]hadap@acm.org, This work is done at MIRALab, University of Geneva towards completion of PhD
[†]thalmann@miralab.unige.ch

hair. Section 3 explains the integration of two seemingly disparate approaches, hair volume as a continuum and dynamics of an individual hair. Section 4 demonstrates how hair-body interactions can be modeled in an unified way as the fluid boundary condition. We also discuss various details about collision detection and response in the section. To add hair-air interaction to the model, Section 5 extends the idea of hair as a continuum to a mixture of hair and air. We outline our scheme for the numerical integration in Section 6. Section 7 address some of the implementation issues that make proposed hair dynamics viable. Finally we show the results that demonstrate the effectiveness of the developed hair dynamics model in animating long hair.

1 Hair as a Continuum

Hair has many properties similar to fluid flow. These similarities were identified and exploited in the chapter – "Modeling Hair Shape as Streamlines of Fluid Flow" for the effective static hair shape modeling. The hair is modeled as streamlines of well setup ideal flow. Unfortunately, in this novel approach, no analogy could be developed between fluid flow and the dynamics of hair. We take inspiration from this approach and in this section explore the possibility of modeling complex hair dynamics as fluid dynamics. We consider if and how we can extend the idea of hair being streamlines of fluid flow by associating shape memory to streamlines.

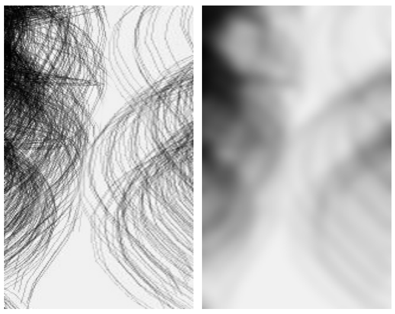


Figure 1: Hair as a Continuum

Hair-hair interaction is very important and the most difficult problem in achieving visually pleasing hair animation. Only recently, Chang *et al* [Chang et al. 2002] and Plante *et al* [Plante et al. 2001] developed hair-hair interaction models based on wisps. They carried out explicit collision detection and response between the wisps of hair. Although these methods are clever and effective, they have the limitation that hair cannot break away from one wisp and join another. We would like to model interactions on hair-hair basis. There are many advances in collision detection and response as compiled by Lin *et al* [Lin and Gottschalk 1998]. However, they are simply unsuitable for the problem at hand because of shear number complexity of hair. This problem warrants to take a radical approach – consider hair as a continuum, see Figure 1. Let us start the discussion by defining the continuum. In the continuum the physical properties of the medium such as pressure, density and temperature are defined at each and every point in the specified region. Fluid dynamics regards liquids and gases as a continuum and even elastic theory regards solids as such, ignoring the fact that they are still composed of individual molecules. Indeed, the assumption is quite realistic at a certain length scale of the observation but at smaller length scales the assumption may not be reasonable. While considering hair as a continuum, it can be argued that hair-hair spacing is not at all comparable to inter molecular distances. However, individual hair-hair interaction is of no interest to us apart from its end effect. Hence, we treat the size of individual hair and hair-hair distance much smaller than the overall volume of hair, justifying the continuum assumption. Panton [Panton 1995] gives an interesting discussion on the continuum assumption. As we develop the model further, it will be apparent that the above assumption is not just about approximating the complex hair-hair interaction. An individual hair is surrounded by air. As it moves, it generates a boundary layer of the air. The boundary layer influences many other hair strands in motion. This aerodynamic form of friction is comparable to mere hair-hair contact friction. In addition, there are electrostatic forces to take part in the dynamics. It is not feasible to model these complex multiple forms of interactions accurately. This inspires us to consider interaction of individual hair strand with the other surrounding strands, in a macroscopic manner, through the continuum assumption. That way, we hope to have a sound model for an otherwise very complex phenomenon.

As we start considering hair as a continuum, we define the properties of such a medium, namely the hair medium. There are two possibilities – hair medium could be considered as a solid or a liquid. This depends on how it behaves under shearing forces. Under shearing stresses, the solids deform till they generate counter stresses. If the shearing stresses are removed, the solids exhibit ability of retaining their original shape. The liquids are not able to withstand any shearing stresses. Under the influence of the shearing stresses they continue to deform indefinitely and they don't have any shape memory. In case of hair, if we apply a lateral shearing motion it acts like a liquid. At the same time, length wise, it acts as a solid. They even have bending rigidity. Thus, there is a duality in the behavior of hair as a continuum.

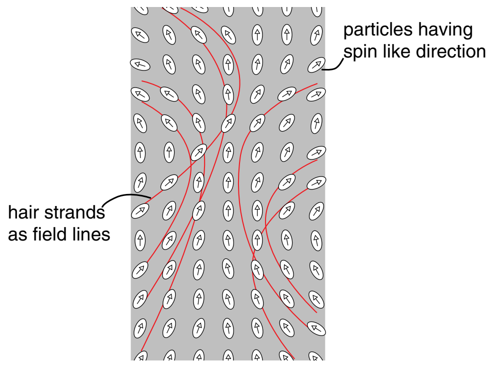


Figure 2: Hair as Field Lines of Oriented Molecules

Interestingly, there are certain liquids that too exhibit this duality – liquid crystals. Liquid crystal molecules have certain preferred orientations as shown in Figure 2, which form a field. The molecules can no more freely demonstrate the fluid like motion because of their preferred directions. Inspired from liquid crystals, we tried to develop a hair dynamics model based on the assumption that each particle of the hair medium will have a spin like pseudo direction, which defines a field. Then the field lines would be synonymous to individual hair strands. The hair dynamics can be formulated based on the anisotropy due to particle orientations. Further, we would have associated some deformation energy to the field lines to simulate the shape memory associated with the hair strand, discussed earlier in this section. However, we realized that this kind of dynamics mimics hair dynamics only instantaneously. Although hair can get sheared laterally, this cannot happen indefinitely. Soon the global, lengthwise effects would come into effect to restrict the lateral motion. Secondly, as the particles of the hair medium move, they form new field lines hence new hairs. This leads to the problem of frame coherency in the model. This would be visually quite disturbing in successive frames of animation. Even from mere rendering point of view, we cannot treat hair solely as a continuum, unless the viewpoint is far enough and individual hair movement is not perceived. Thus, we have to retain the individual character of hair as well, while considering hair as a continuum. Finally, we realize the model by splitting hair dynamics into two parts:

- Hair-hair, hair-body and hair-air interactions, which are modeled using continuum dynamics, and more precisely fluid dynamics

- Individual hair geometry and stiffness, which is modeled using the dynamics of an elastic fiber

Interestingly, this approach even addresses the solid-liquid duality effectively. The model can be visualized as a bunch of hair strands immersed in a fluid. The hair strands are kinematically linked to fluid particles in their vicinity. The individual hair has its own stiffness dynamics and it interacts with the environment through the kinematical link with the fluid. The stiffness dynamics of an individual hair is quite straight forward, which is developed in the next section.

In order to develop the continuum model further, let us start identifying various physical quantities involved in fluid dynamics. Density, pressure and temperature are the basic constituents of fluid dynamics. The density of the hair medium is not precisely the density of individual hair. It is rather associated with the number density of hair in an elemental volume. In Figure 1, observe that the density of the hair medium is less when the number density of the hair is less. The density of the hair medium is thus defined as the mass of the hair per unit occupied volume and is denoted as ρ. The notion of density of the hair medium enables us to express the conservation of mass (it is rather conservation of the number of hair strands) in terms of the continuity equation [Panton 1995]

$$\frac{1}{\rho}\frac{d\rho}{dt} = -\nabla \cdot \vec{v} \tag{1}$$

where, $\vec{v}$ is the local velocity of the medium. Note that the fluid dynamics equations are in the Langrangian form, unlike more popular Eulerain form. This is more explained in section 3. The continuity equation states that the relative rate of change of density ($\frac{1}{\rho}\frac{d\rho}{dt}$), at any point in the medium, is equal to the negative gradient of the velocity field at that point ($-\nabla \cdot \vec{v}$). This is the total outflux of the medium at that point. The physical interpretation of the continuity equation in our case is that, as the hair strands start moving apart, their number density, and hence the density of the hair medium drops and vice a versa.

The pressure and the viscosity in the hair medium represent all the forces due to various forms of interactions of hair strands as described previously. If we try to compress a bunch of hair, it develops a pressure such that the hair strands will tend to move apart. The viscosity would account for various forms of interactions such as hair-hair, hair-body and hair-air. These are captured in the form of the momentum equation [Panton 1995] of fluid.

$$\rho \frac{d\vec{v}}{dt} = \nu \nabla \cdot (\nabla \vec{v}) - \nabla p + F_{bd} \tag{2}$$

The acceleration of the fluid particles $\frac{d\vec{v}}{dt}$ with the spatial pressure variation $-\nabla p$ would be such that it will tend to even out the pressure differences and as the fluid particles move, there will be always resistance $\nu \nabla \cdot (\nabla \vec{v})$ in the form of the friction. The body forces F_{bd}, *i.e.* the inertial forces and gravitational influence are also accounted for in the equation.

Temperature considerably affects the properties of hair. However, we do not have to consider it in the dynamics. We treat the hair dynamics as an isothermal process unless we are trying to simulate a scenario of hair being dried with a hair dryer. Secondly, the temperature is associated with the internal energy of the fluid, which is due to the continuous random motion of the fluid molecules. At the length scale of our model *i.e.* treating hair as a continuum, there is no such internal energy associated with the hair medium. Subsequently, we drop the energy equation of fluid, which is associated with the temperature and the internal energy.

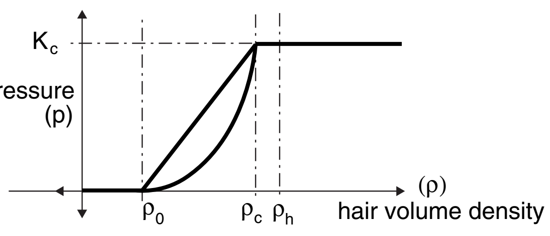


Figure 3: Equation of State

The equation of state (EOS) [Panton 1995] binds together all the fluid equations. It gives a relation between density, pressure and temperature. In our case of hair-hair interaction, the EOS plays a central role along with the viscous fluid forces. The medium we are modeling is not a real medium such as gas or liquid. Hence, we are free to "design" EOS to suit our needs. The following equation is our proposition:

$$p = \begin{cases} 0 & \text{if } \rho < \rho_0, \\ K_c(\frac{\rho-\rho_0}{\rho_c-\rho_0})^n & \text{if } \rho_0 \leq \rho < \rho_c, \\ K_c & \text{if } \rho_c < \rho \end{cases} \tag{3}$$

We define the hair rest density ρ_0 as a density below which statistically there is no hair-hair collisions. In addition, we define hair close packing density as ρ_c that represents the state of the hair medium in which hair strands are packed to the maximum extent. This density is slightly lower than the physical density of hair, ρ_h. Figure 3 illustrates the relation between the density and the pressure of the hair medium. In the proposed pressure/density relationship, notice that there is no pressure built up below the hair rest density ρ_0. As one starts squeezing the hair volume, pressure starts building up. As a consequence, the hair strands are forced apart. At the hair compaction density ρ_c, the pressure is maximum. K_c is the interaction constant of the hair volume. The power n refers to the ability of hair volume to get compressed. If the hair is well aligned, the power will be high. In this case, as we compress the hair volume, suddenly the hair strands start to form close packing and build the pressure quickly. On the contrary, if hair is wavy and not very well aligned, the pressure build up is not abrupt. This will lead to power n towards one.

Instead of modeling the collisions of individual hair strand with the body, we model them, in a unified way, as a boundary condition of the fluid flow. There are two forms of the fluid boundary conditions a) flow tangency condition - the fluid flow normal to the obstacle boundary is zero. b) flow slip condition - the boundary exerts a viscous pressure proportional to the tangential flow velocity. The formulation of the flow boundary condition is deferred to Section 3, where we will introduce the numerical fluid model. It will be apparent that although we model the hair-body interactions as fluid boundary condition, after discretization, the model directly falls under traditional collision detection and response techniques.

Having developed the groundwork for hair-hair and hair-body interactions, in the next section we develop an elaborate stiffness dynamics of the individual hair strand.

2 Single Hair Dynamics

In the previous section we discussed how we could think of hair-hair interaction as fluid forces by considering hair volume as a continuum. However, for the reasons explained there, we still need to retain the individual character of a hair strand. The stiffness dynamics of an individual hair is discussed in this section.

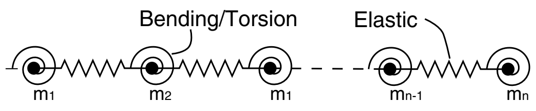


Figure 4: Hair Strand as an Oriented Particle System

In the case of single hair dynamics, as discussed in [Hadap 2003; Magnenat-Thalmann et al. 2000], there are two approaches so far – cantilever dynamics and mass-spring-hinge dynamics. We have seen that the cantilever dynamics is not truly three dimensional. Thus, we sincerely feel that it has limited potential in the light of current computational power. We discuss the spring-mass-hinge model and highlight its limitations, which leads to the development of our model. In a very straightforward manner, one models hair as a set of particles connected by tensile, bending and torsional springs [Daldegan et al. 1993; Rosenblum et al. 1991], as shown in Figure 4. If the hair strand is approximated by a set of n particles, then the system has $6n$ degrees of freedoms (DOFs) attributed to three translations, two bendings and one twist per particle. Treating each particle as a point mass, we can setup a set of governing differential equations of motion and try integrating them. Unfortunately this is not a viable solution. Hair is one of the many interesting materials in nature. It has remarkably high Elastic Modulus of 2-6GPa. Moreover, being very small in diameter, it has very large tensile strength as compared to its bending and torsional rigidity. This proves to be more problematic in terms of the numerics. We are forced to choose very small time steps due to the stiff equations corresponding to the tensile mode of motion, in which we are hardly interested. In fact, the hair fiber hardly stretches by its own weight and body forces. It just bends and twists.

Hence, it is better to choose one of the following two possibilities. Constrain the differential motion of the particles that amounts to the stretching using *constrained dynamics* [Baraff 1996]. Alternatively, reformulate the problem altogether to remove the DOFs associated with the stretching, namely a *reduced coordinate formulation* [Featherstone 1987]. Both methods are equally efficient, being linear time. Parameterizing the system DOFs by an exact number of generalized coordinates may be extremely hard or even impossible for the systems having complex topology. In this case, a constrained method is preferred for its generality and modularity in modeling complex dynamical systems. However, for the problem at hand, the reduced coordinate formulation is a better method for the following reasons:

- Reduced coordinates are preferred when in our case the $3n$ DOFs remaining in the system are comparable to the $3n$ DOFs removed by the elastic constraints.

- The system has fixed and simple topology where each object is connected to maximum of two neighbors. We can take advantage of the simplicity and symbolically reduce the most of the computations.

- Reduced coordinate formulation directly and accurately facilitates the parametric definition of bending and torsional stiffness dynamics. The geometry of spring-mass-hinge system is one dimensional and the masses are point masses. Thus it is difficult to accurately formulate the bending and torsional dynamics as one can not effectively resolve the orientations that facilitate definition of bending and torsion in three dimensions.

Subsequently we model an individual hair strand as a *serial rigid multi-body chain.*

2.1 Hair as Serial Rigid Multi-body Chain

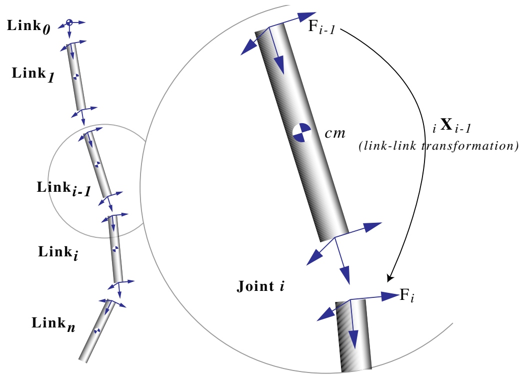


Figure 5: Hair Strand as Rigid multi-body Serial Chain

The first step is to clearly define the serial rigid multi-body system that approximates the motion of individual hair strand. We divide the strand into n segments of equal length as shown in Figure 5. The advantages of defining segments of equal length will be made clear, subsequently. The n segments are labeled as $link_1$ to $link_n$. Each link is connected to two adjacent links by a three DOF spherical joint forming a single un-branched open-loop kinematic chain. The joint between $link_{i-1}$ and $link_i$ is labeled $joint_i$. The position where the hair is rooted to scalp is synonymous to $link_0$ and the joint between head and hair strand is $joint_1$.

Further, we introduce n coordinate frames $\mathfrak{F}_i$, each attached to the corresponding $link_i$. The coordinate frame $\mathfrak{F}_i$ moves with the $link_i$. The placement of coordinate system is largely irrelevant to the mathematical formulations, but they do have an important bearing on efficiency of computations, which is discussed subsequently. Having introduced the link coordinates, we introduce the *spatial transformations* that enable us to transform *spatial entities* defined in the coordinate frame of one link, in terms of the coordinate frame of the adjacent link. ${}_i\hat{\mathbf{X}}_{i-1}$ is an adjacent-link coordinate *spatial transformation* which operates on a *spatial vector* represented in coordinate frame $\mathfrak{F}_{i-1}$ and produces a representation of the same spatial vector in coordinate frame $\mathfrak{F}_i$. For comprehensive discussion on spatial vector algebra and it's application to rigid body dynamics along with the peculiar notations, we refer to pioneering work by Featherstone [Featherstone 1987].

We use the notations introduced by Featherstone. Small case letters such as l denote scalars and bold face letters such as $\mathbf{v}$ denote cartesian vectors. Spatial 6×1 vectors and spatial 6×6 matrices are denoted by bold face small and capital letters, respectively, having a hat on top, e.g. $\hat{\mathbf{v}}$. Subscript on an entity denotes the associated link, e.g. $\hat{\mathbf{v}}_i$ denote the spatial velocity of the i^{th} link. Entities with dash, e.g. $\hat{\mathbf{I}}'$, denote that they are defined in the local coordinate frame. The spatial transpose operator is denoted by a superscript S, e.g. $\hat{\mathbf{X}}^S$.

Figure 5 illustrates the definition of a hair strand as a serial multi-body rigid chain. The spatial transformation ${}_i\hat{\mathbf{X}}_{i-1}$ is composed of a pure translation, which is constant as the length of the segment is constant, and a pure orientation which is variable. We use a unit quaternion $\mathbf{q}_i$ to describe the orientation of each link with respect to the previous link. Then, we augment the components of n quaternions, one per joint, to form $\mathbf{q} \in \Re^{4n}$, the system state vector. Note that, additional n unit quaternion constraints, *i.e.* $|\mathbf{q}_i| = 1$, make system have $3n$ coordinates. Thus system is optimally represented to have $3n$ DOFs. Moreover, the angular velocity across the spherical joint is described by conventional 3×1 angular velocity vector $\mathbf{w}_i$. These form $\mathbf{w} \in \Re^{3n}$, the derivative state vector of the system. The spatial motion of the rigid body, $link_i$ in our case, is fully characterized by its 6×1 *spatial velocity* $\hat{\mathbf{v}}_i$, 6×1 *spatial acceleration* $\hat{\mathbf{a}}_i$ and 6×6 *spatial inertia tensor* $\hat{\mathbf{I}}_i$. In the next subsections, we will formulate the spatial dynamics of serial rigid multi-body chain in terms of the system state variables $\mathbf{q}$ and $\mathbf{w}$ and their respective derivatives $\dot{\mathbf{q}}$ and $\dot{\mathbf{w}}$, using the physical quantities $\hat{\mathbf{v}}_i$, $\hat{\mathbf{a}}_i$ and $\hat{\mathbf{I}}_i$ for dynamics.

2.2 Kinematics of Hair Strand

A 6×3 motion sub-space $\hat{\mathbf{S}}$ relates the angular velocity $\mathbf{w}_i$ to spatial velocity across the joint, which is the only allowed motion by the spherical joint. Since the position of the link in its own coordinate frame remains fixed, we can express the motion sub-space $\hat{\mathbf{S}}$ as a constant matrix.

$$\hat{\mathbf{S}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{4}$$

Subsequently, the velocity and acceleration across the spherical joint are given by the following equations:

$$\begin{aligned} \hat{\mathbf{v}}_i &= {}_i\hat{\mathbf{X}}_{i-1}\hat{\mathbf{v}}_{i-1} + \hat{\mathbf{S}}\mathbf{w}_i \\ \hat{\mathbf{a}}_i &= {}_i\hat{\mathbf{X}}_{i-1}\hat{\mathbf{a}}_{i-1} + \hat{\mathbf{v}}_i \hat{\times}\, \hat{\mathbf{S}}\mathbf{w}_i + \hat{\mathbf{S}}\dot{\mathbf{w}}_i \end{aligned} \tag{5}$$

Given joint angular velocities $\mathbf{w}_i$ and joint angular accelerations $\dot{\mathbf{w}}_i$, Equations 4 and 5 enable us to recursively compute the link velocities $\hat{\mathbf{v}}_i$ and the link accelerations $\hat{\mathbf{a}}_i$, with $\hat{\mathbf{v}}_0$ and $\hat{\mathbf{a}}_0$ as a starting point. In our case $\hat{\mathbf{v}}_0$ and $\hat{\mathbf{a}}_0$ are the spatial velocity and the spatial acceleration of hair root, *i.e.* the scalp. We need to successively integrate the derivative vectors of the system *i.e.* integrating $\dot{\mathbf{w}}_i$ into $\mathbf{w}_i$ and $\mathbf{w}_i$ into $\mathbf{q}_i$. One can notice that the angular velocity $\mathbf{w}_i$ can not be integrated directly into joint variables $\mathbf{q}_i$. However, the following equation relates the joint variable rates $\dot{\mathbf{q}}_i$ expressed as quaternions to the angular velocities $\mathbf{w}_i$

$$\begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = 1/2 \begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & -q_3 & q_2 \\ q_3 & q_0 & -q_1 \\ -q_2 & q_1 & q_0 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} \tag{6}$$

$$q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1 \tag{7}$$

Next step is to identify various external forces acting on the links, which induce the joint angular accelerations and make hair strand bend and move.

2.3 Forward Dynamics of Hair Strand

Before we discuss the dynamics of single hair strand, we tabulate the physical properties of a typical human hair strand. Especially note the formulas for the moment of area, the polar moment of area, bending spring constant and torsional spring constant. The detailed discussion on the hair properties is covered in [Hadap 2003].

Number of hair strands on human scalp	90-120 thousand, we use 20-40 thousand for animation purpose which are "data-amplified" to around 50-60 thousand for rendering purpose
Typical diameter ($D = 2R$)	60-100 μm
Cross section	Circular to elliptical, we assume circular
Typical distance between hair on scalp	1mm
Linear density of hair strand	30-100 μgm/cm
Density of hair material	1.2 gm/cm^3
Elastic modulus (E)	2-6 GPs
Moment of area (I)	$\pi R^4/4$
Polar moment of area (I_p)	$\pi R^4/2$
Equivalent bending spring constant (K_b)	EI/l, where l is length of the hair segment
Equivalent torsional spring constant (K_t)	$\frac{1}{2(1+\nu)}\frac{EI_p}{l}$, where ν is Poisson ratio

Table 1: Typical Physical Properties of Human Hair

A number of forces act on each link apart from the gravitational influence $\hat{\mathbf{g}}$. The explicit definition of the point of action of the spatial force on link is irrelevant as it is embedded in the definition of the spatial force, thus resulting in a very compact representation.

- The gravitational influence is accommodated by giving the base of the zeroth link representing the root a fictitious additional negative gravitational acceleration, *i.e.* by subtracting $\hat{\mathbf{g}}$ from $\hat{\mathbf{a}}_0$.
- The inertial dynamics plays a pivotal role in the case of dynamics of hair strand, even though the hair strand is thin in geometry. Inertia of the individual segment is indeed small as compared to its stiffness. However, it is the first and second moments of inertia that govern the dynamics. The serial rigid-multibody chain formulation facilitates us to accurately account for the inertial dynamics of the hair strand. The *spatial momentum* of each link is composed of the spatial velocity $\hat{\mathbf{v}}_i$ and the *spatial inertia* $\hat{\mathbf{I}}_i$, a 6x6 matrix. Since the position of the link in its own coordinate frame remains fixed, we can express the spatial inertia $\hat{\mathbf{I}}_i$ as a constant. Further, by proper choice of coordinate system, $\hat{\mathbf{I}}_i$ assumes a rather simple form.

$$\hat{\mathbf{I}} = \begin{bmatrix} \mathbf{H}^T & \mathbf{M} \\ \mathbf{I} & \mathbf{H} \end{bmatrix} \tag{8}$$

where $\mathbf{M}$, $\mathbf{H}$ and $\mathbf{I}$ are the 3x3 matrix representations of zeroth, first and second moments of mass of the link around the origin of its own frame as described above. Notice that although the mass of the individual link is small and subsequently the mass matrix $\mathbf{M}$ tend to be singular, due to well conditioned $\mathbf{H}$ and $\mathbf{I}$, $\hat{\mathbf{I}}$ is not singular. Table 1 gives the expressions for moment of area and polar moment of area of the cylindrical hair segment, which enables us to formulate the second moment of inertia $\mathbf{I}$.

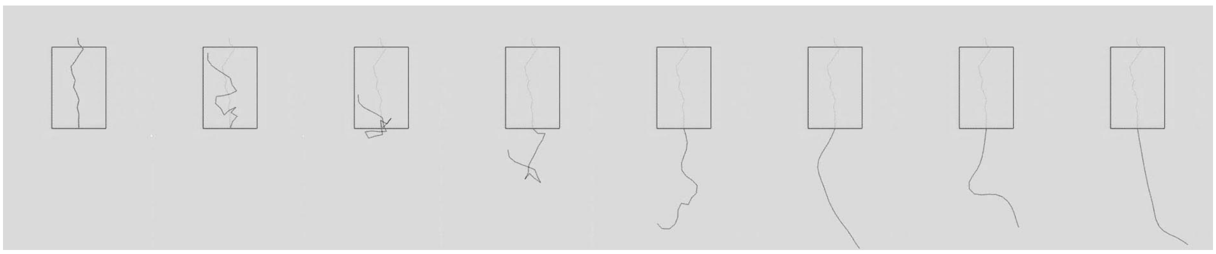


Figure 6: Free-fall of Hair Strand – No Stiffness

Figure 6 illustrates the motion of free falling hair strand without stiffness. Thus the motion is solely governed by the gravity and inertial dynamics of links. This motion is similar to that of a chain. Needless to state that the elongation constraint is always maintained as the system does not have any corresponding DOF in the definition. We defer the details of the algorithm that computes the motion till the next section, where we will have defined all the forms of forces acting on the hair strand.

- In order to account for the bending and torsional rigidity of the hair strand, the $joint_i$ exerts an actuator force $\mathbf{Q}_i^a \in \Re^3$ on both $link_{i-1}$ and $link_i$ in opposite directions. The actuator force is not a spatial force but rather a force expressed in joint motion space. The joint actuator force is a function of joint variables $\mathbf{q}_i$ incorporating the bending and torsional stiffness constants. To realize the joint actuator force, we uniquely decompose the joint variable $\mathbf{q}_i$ into a pure bending component θ_i^b around the axis $\mathbf{b}_i$ followed by a pure twist component θ_i^t around link axis. We would like to highlight that this unique decomposition is only possible due to the accurate representation of the orientation of the adjacent links via joint variable $\mathbf{q}_i$. Similarly, the neutral hair strand shape defines a set of neutral orientations $\mathbf{q}_i^0$ which are decomposed into the neutral bending component θ_i^{b0} around axis $\mathbf{b}_i^0$ and the pure twist component θ_i^{t0}. From θ_i^b, $\mathbf{b}_i$, θ_i^{b0}, $\mathbf{b}_i^0$, θ_i^t and θ_i^{t0}, given equivalent bending spring constant and equivalent twist spring constant listed in Table 1 along with respective damping constants, one can formulate the actuator force $\mathbf{Q}_i^a$. The details of the formulation needs the discussion on how to represent quantities in the joint space, instead we refer to discussion in [Featherstone 1987].

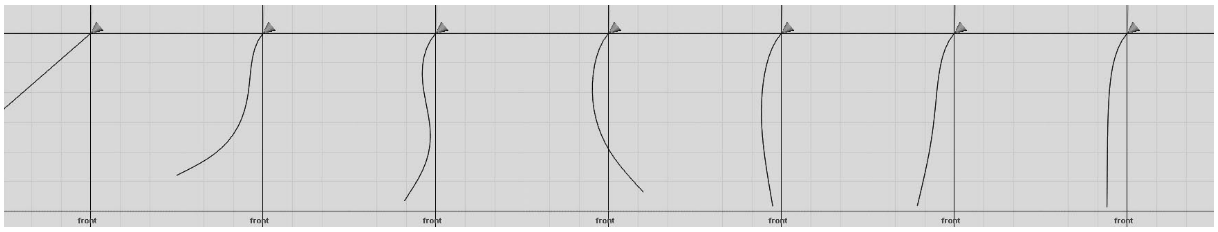


Figure 7: Shape-memory of Hair Strand

Figure 7 illustrates the stiffness dynamics of a hair strand. Under cantilever action, the hair strand bends in the direction of the gravity. Due to (primarily) bending stiffness, it tries to retain its neutral shape – straight line in this case.

- Force $\hat{\mathbf{f}}_{ci}$ is the interaction spatial force (aggregate of line force and torque) on $link_i$ coming from the kinematic link with the hair medium as discussed in Section 1. The actual form of $\hat{\mathbf{f}}_{ci}$ is given in Sections 3, 4 and 5. This force accounts for all the interaction effects such as hair-hair collision, hair-body collision and hair-air drag.

Given the set of forces acting on the system, we now need to calculate the motion of the hair strand. This evolves calculation of the induced joint angular accelerations $\dot{\mathbf{w}}_i$ followed by the integration. This is a forward dynamics problem involving a rigid multi-body system. We use Articulated-Body Method to solve the hair strand forward dynamics. This method has a computational complexity of $O(n)$. The detailed discussion of this algorithm is beyond the scope of this chapter. It is comprehensively covered in [Featherstone 1987; Mirtich 1996]. In the next section we give a brief outline of the method.

2.4 Articulated-Body Forward Dynamics Algorithm

We use Articulated-Body method to solve the hair strand dynamics stated in the previous section. This method has a computational complexity if $O(n)$ as compared with $O(n^3)$ methods such as Composite-Rigid-Body method. For the detailed discussion of these algorithm refer to [Featherstone 1987].

A collection of rigid bodies connected by joints is called an articulated body. To define an articulated body inertia, we single out a particular member of the articulated body, called the handle, and define the articulated inertia as a relationship between a test force $\hat{\mathbf{f}}$ applied to the handle and the acceleration $\hat{\mathbf{a}}$ of the handle according to

$$\hat{\mathbf{f}} = \hat{\mathbf{I}}^A \hat{\mathbf{a}} + \hat{\mathbf{p}} \tag{9}$$

$\hat{\mathbf{I}}^A$ is the articulated-body inertia and $\hat{\mathbf{p}}$ is the bias force, which is the value of the test force that must be applied to the handle in order to give it zero acceleration. Then, the basic idea of the articulated-body algorithm is to treat n-joint multi-body system as a one-joint system whose only moving link is in fact the handle of an articulated body comprising all the remaining links. Then we find the acceleration of the first joint solving the forward dynamics of a single joint robot, which is relatively simple. Having solved for acceleration of joint 1, we can treat link 1 as the (moving) base of an $n-1$ joint robot and repeat the process for joint 2, and so on.

The algorithm is a 3 step process.

Common sub-expressions

$$\hat{\mathbf{c}}_i = \hat{\mathbf{v}}_i \hat{\times}\, \hat{\mathbf{S}}\mathbf{w}_i \quad (10)$$

$$\hat{\mathbf{h}}_i = \hat{\mathbf{I}}_i^A \hat{\mathbf{S}} \quad (11)$$

$$d_i = \hat{\mathbf{S}}^S \hat{\mathbf{h}}_i \quad (12)$$

$$u_i = \hat{\mathbf{S}} Q_i^a - \hat{\mathbf{h}}_i^S \hat{\mathbf{c}}_i - \hat{\mathbf{S}}^S \hat{\mathbf{p}}_i \quad (13)$$

Step 1

$$\hat{\mathbf{v}}_i = {}_i\hat{\mathbf{X}}_{i-1} \hat{\mathbf{v}}_{i-1} + \hat{\mathbf{S}}\mathbf{w}_i \quad (14)$$

Step 2

$$\hat{\mathbf{p}}_i^v = \hat{\mathbf{v}}_i \hat{\times}\, \hat{\mathbf{I}}_i \hat{\mathbf{v}}_i \quad (15)$$

$$\hat{\mathbf{I}}_i^A = \hat{\mathbf{I}}_i +_i \hat{\mathbf{X}}_{i+1}(\hat{\mathbf{I}}_{i+1}^A - \frac{\hat{\mathbf{h}}_{i+1}\hat{\mathbf{h}}_{i+1}^S}{d_{i+1}})_{i+1}\hat{\mathbf{X}}_i, \quad (\hat{\mathbf{I}}_n^A = \hat{\mathbf{I}}_n) \quad (16)$$

$$\hat{\mathbf{p}}_i = \hat{\mathbf{p}}_i^v +_i \hat{\mathbf{X}}_{i+1}(\hat{\mathbf{p}}_{i+1} + \hat{\mathbf{I}}_{i+1}^A \hat{\mathbf{c}}_{i+1} + \frac{u_{i+1}}{d_{i+1}}\hat{\mathbf{h}}_{i+1}), \quad (\hat{\mathbf{p}}_n = \hat{\mathbf{p}}_n^v) \quad (17)$$

Step 3

$$\dot{\mathbf{w}}_i = \frac{u_i - \hat{\mathbf{h}}_i^S \; {}_i\hat{\mathbf{X}}_{i-1}\hat{\mathbf{a}}_{i-1}}{d_i} \quad (18)$$

$$\hat{\mathbf{a}}_i = {}_i\hat{\mathbf{X}}_{i-1}\hat{\mathbf{a}}_{i-1} + \hat{\mathbf{c}}_i + \hat{\mathbf{S}}\dot{\mathbf{w}}_i \quad (19)$$

§1 Start from $\hat{\mathbf{v}}_0$, the velocity of the base *i.e.* that of the hair strand root. Using current value of joint angular velocities $\mathbf{w}_i, i = 1 \ldots n$, compute all the link velocities $\hat{\mathbf{v}}_i, i = 1 \ldots n$ from $\hat{\mathbf{v}}_{i-1}$, using equation 14.

§2 Given link spatial inertias $\hat{\mathbf{I}}_i, i = 1 \ldots n$, start from the last link's articulated-body inertia $\hat{\mathbf{I}}_n^A = \hat{\mathbf{I}}_n$ and bias force $\hat{\mathbf{p}}_n = \hat{\mathbf{v}}_n \hat{\times}\, \hat{\mathbf{I}}_n \hat{\mathbf{v}}_n$. Compute all the articulated-body inertias $\hat{\mathbf{I}}_i^A, i = n-1 \ldots 1$ and the bias forces $\hat{\mathbf{p}}_i, i = n-1 \ldots 1$ from $\hat{\mathbf{I}}_{i+1}^A$ and $\hat{\mathbf{p}}_{i+1}$, using equations in Step 2.

§3 Once we know all the articulated-body inertias and bias forces we start from the link 1. Given all the external forces acting on links (see Section 2.3), we compute the joint angular accelerations $\dot{\mathbf{w}}_i, i = 1 \ldots n$ using equations in Step 3. We update the link acceleration $\hat{\mathbf{a}}_i$ before we move on to the next link.

The time evolution of the hair strand shape is broken into discrete steps in time. At each time step, we evaluate the joint angular accelerations $\dot{\mathbf{w}}_i, i = 1 \ldots n$ followed by the numerical integration. The details of numerical integration are covered in Section 6 after the details of all the forces acting on the hair strand are covered in the subsequent sections.

3 Fluid Hair Model

In the previous section we described the precise dynamics of individual hair strand. We considered bending and torsional stiffness of hair along with body forces *viz.* inertia and gravitational influence. In this section, we develop on the proposed continuum model for hair-hair interactions. As discussed in Section 1, the density and the pressure of the hair medium form the basic constituents of the fluid-hair model. The continuity equation (Eq. 1), the momentum equation (Eq. 2) and the equation of state (Eq. 3) capture the overall dynamics of hair-hair interaction. Establishing

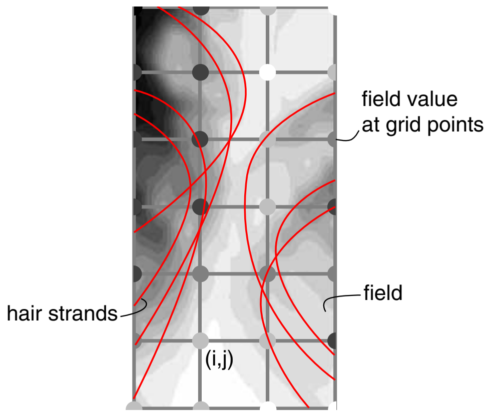


Figure 8: Fluid Dynamics – Eulerian viewpoint

the kinematical link between the dynamics of the individual hair strand and the dynamics of interactions is a crucial part of the algorithm, which is addressed in this section.

The conventional fluid dynamics formulation uses Eulerian viewpoint. One way to think of Eulerian method is to think of an observer watching the fluid properties such as density, temperature and pressure change at a certain fixed point in space, as fluid passes through this point. In the numerical simulations, the space is discretised using a rectangular grid or a triangular mesh to define these few observation points for computations, as shown in Figure 8. Hence using the Eulerian viewpoint, we will ultimately get fluid forces acting at this fixed set of points. We would like to transfer the fluid force at each of these points onto the individual hair, which is in the vicinity of the point. There is no trivial correlation between the grid points and the hair strands, unless they coincide. Also the hair strand will be in the vicinity of new set of grid points every time it moves. This makes it difficult to formulate the kinematical link between the two. There are methods such as the particle-in-cell method introduced by Hockney *et al* [Hockney and Eastwood 1988], which try to do the same. However, we opted for the other, less popular but effective, Langrangian formulation of fluid dynamics. We explain the benefits subsequently.

In Langrangian formulation, the physical properties are expressed as if the observer is moving with the fluid particle. *Smoothed Particle Hydrodynamics* (SPH), invented by Monaghan [Monaghan 1992], is one of the Langrangian numerical methods, that utilizes space discretisation via number of discrete points that move with the fluid flow. One of the first applications of SPH in computer animation was done by Gascuel *et al* [Gascuel et al. 1996]. For a good overview of SPH, we refer to [Morris 1995].

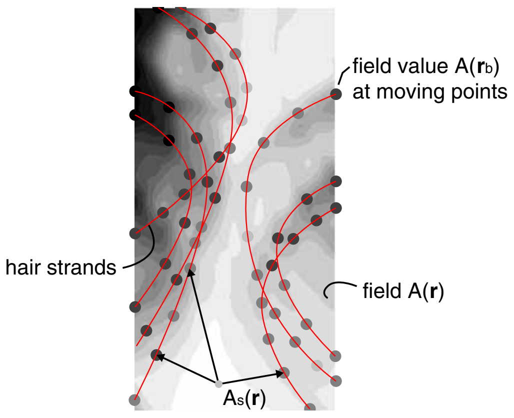


Figure 9: Fluid Dynamics – Langrangian viewpoint

Figure 9 illustrates the concept of smoothed particles. The physical properties are expressed at the center of each of these smoothed particles. Then the physical property at any point in the medium is defined as a weighted sum of the properties of all the particles.

$$A_s(\mathbf{r}) = \sum_b A_b \frac{m_b}{\rho_b} W(\mathbf{r} - \mathbf{r}_b, h) \tag{20}$$

The summation interpolant $A_s(\mathbf{r})$ can be thought of as the smoothed version of the original property function $A(\mathbf{r})$. The field quantities at particle b are denoted by a subscript b. Thus, the mass associated with particle b is m_b and density at the centre of the particle b is ρ_b, and the property itself is A_b. We see that the quantity $\frac{m_b}{\rho_b}$ is the inverse of the number density (*i.e.* the specific volume) and is, in some sense, a volume element. The function W is the weight function referred as interpolating kernel in SPH. Details of the interpolating function are covered subsequently.

To exemplify, the smoothed version of density at any point of medium is

$$\rho(\mathbf{r}) = \sum_b m_b W(\mathbf{r} - \mathbf{r}_b, h) \tag{21}$$

Figure 9 illustrates how density is recorded onto each particle, denoted by varying degree of gray scale values of the dots. The field is defined at each and every point in the region by weighted sum of the field values of the surrounding particles, which is denoted by the continuous gray tones in the region.

Similarly, it is possible to obtain an estimate of the gradient of the field, provided W is differentiable, simply by differentiating the summation interpolant

$$\nabla A_s(\mathbf{r}) = \sum_b A_b \frac{m_b}{\rho_b} \nabla W(\mathbf{r} - \mathbf{r}_b, h) \tag{22}$$

The interpolating kernel $W(\mathbf{r} - \mathbf{r}', h)$ has the following properties

$$\int W(\mathbf{r} - \mathbf{r}', h) d\mathbf{r}' = 1 \tag{23}$$

$$\lim_{h \to 0} W(\mathbf{r} - \mathbf{r}', h) = \delta(\mathbf{r} - \mathbf{r}') \tag{24}$$

The choice of the kernel is not important in theory as long as it satisfies the above kernel properties. However, for practical purposes we need to choose a kernel, which is simple to evaluate and has compact support. The *smoothing length h* defines the extent of the kernel. We use the cubic spline interpolating kernel.

$$W(\mathbf{r}, h) = \frac{\sigma}{h^\nu} \begin{cases} (1 - \frac{3}{2}s^2 + \frac{3}{4}s^3) & \text{if } 0 \le s \le 1, \\ \frac{1}{4}(2 - s)^3 & \text{if } 1 \le s \le 2, \\ 0 & \text{otherwise} \end{cases} \tag{25}$$

Where $s = |\mathbf{r}|/h$, ν is the number of dimensions and σ is the normalization constant with values $\frac{2}{3}$, $\frac{10}{7\pi}$, or $\frac{1}{\pi}$ in one, two, or three dimensions, respectively. We can see that the kernel has a compact support, *i.e.* its interactions are exactly zero at distances $|\mathbf{r}| > 2h$. Evaluating the field at each point involves computation of the contribution due to all the particles in the region. The compact support, i.e. the kernel has zero value outside the smoothing length, drastically reduces the computational overhead as we need to consider only the neighboring particles within the smoothing length in order to evaluate the function at a point. Figure 10 illustrates a typical kernel having a compact support. We keep the smoothing length h constant throughout the simulation to facilitate a speedy search of neighborhood of the particles. The nearest neighbor problem is well known in computer graphics. Section 7 gives the strategy for the linear time neighbor search.

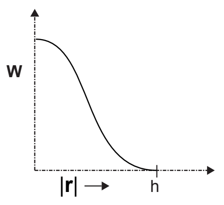


Figure 10: SPH Kernel having Compact Support

There is no underlying grid structure in the SPH method, which makes the scheme suitable for our purpose. We are free to choose the initial positions of the smoothed particles as long as their distribution reflects the local density depicted by Equation 21. Eventually the particles will move with the fluid flow. In order to establish the kinematical link between the individual hair dynamics and the dynamics of interactions, we place the smoothed particles directly onto the hair strands as illustrated in the Figure 9. We keep the number of smoothed particles per hair segment constant, just as we have kept the hair segment length constant, for the reasons of computational simplicity. As the smoothed particles are glued to the hair strand, they can no longer move freely with the fluid flow. They just exert forces arising from the fluid dynamics onto the corresponding hair segment and move with the hair segment (in the figure, the hair strand is not discretised to show the segments). Thus, we have incorporated both, the elastic dynamics of individual hair and the dynamics of interactions into hair dynamics.

Apart from providing the direct kinematical link, the SPH method has other numerical merits when compared to a grid-based scheme:

- As there is no need for a grid structure, we are not defining a region of interest to which the dynamics must confine to. This is very useful considering the fact that, in animation the character will move a lot and the hair should follow it.
- No memory is wasted in defining the field in the region where there is no hair activity, which is not true in the case of grid-based fluid dynamics.
- As the smoothed particles move with the flow carrying the field information, they optimally represent the fluctuations of the field. In the case of grid-based scheme, it is necessary to opt for tedious adaptive grid techniques to achieve similar computational resolution, within given memory footprint.

In the rest of the section, we discuss the SPH versions of the fluid dynamics equations. Each smoothed particle has a constant mass m_b. The mass is equal to the mass of the respective hair segment divided by the number of smoothed particles on that segment. Each particle carries a variable density ρ_b, variable pressure p_b and has velocity $\mathbf{v}_b$. The velocity $\mathbf{v}_b$ is actually the Cartesian velocity of the point on the hair segment where the particle is located, expressed in the global coordinates. $\mathbf{r}_b$ is the global position of the particle, *i.e.* the location of the particle on the hair strand in the global coordinates. Once, initially, we have placed the particles on the hair strands, we compute the particle densities using Equation 21. Indeed, the number density of hair at a location reflects the local density, which is consistent with the definition of the density of the hair medium given in Section 1.

For brevity, we introduce the notation $W_{ab} = W(\mathbf{r}_a - \mathbf{r}_b, h)$. Similarly, let $\nabla_a W_{ab}$ denote the gradient of W_{ab} with respect to $\mathbf{r}_a$ (the coordinates of particle a). The quantities such as $\mathbf{v}_a - \mathbf{v}_b$ shall be written as $\mathbf{v}_{ab}$.

The density of each particle can be always found from Equation 21, but this equation requires an extra loop over all the particles, which means the heavy processing of nearest neighbour finding, before it can be used in the calculations. A better formula is obtained from the smoothed version of the continuity equation, Equation 1.

$$\frac{d\rho_i}{dt} = \rho_i \sum_{j=1}^{N} \frac{m_j}{\rho_j} \mathbf{v}_{ij} \cdot \nabla_i W_{ij} \tag{26}$$

Using this formula, we now can update the particle density without going through the particles just by integrating the above equation. However, we would have to correct the densities from time to time using Equation 21, to avoid the density being drifted due to numerical inaccuracies.

The smoothed version of the momentum equation, Equation 2, without the body forces, is as follows

$$\frac{d\mathbf{v}_i}{dt} = -\sum_{j=1}^{N} m_j \left(\frac{p_j}{\rho_j^2} + \frac{p_i}{\rho_i^2} + \prod_{ij}\right) \nabla_i W_{ij} \tag{27}$$

The reason for dropping the body force F_{bd} is that, the comprehensive inertial and gravitational effects are already incorporated in the stiffness dynamics of the individual strand. Otherwise, we would be duplicating them.

As the particles are glued to the respective hair segment, they cannot freely attain the acceleration $\frac{d\mathbf{v}_i}{dt}$ given by the momentum equation. Instead, we convert the acceleration into a force by multiplying both the sides of Equation 27 with the mass of the particle m_i. Thus, instead of particle accelerating according to the governing equations of motion, they merely apply forces, arising from the fluid dynamics, onto the hair strand. In the previous section, we referred this total of all the fluid forces due to each particle on the segment as the interaction force $\hat{\mathbf{f}}_{ci}$. Although, we need to convert the Cartesian form of the force into the spatial force in order to incorporate it in the spatial dynamics – this is straightforward.

In Equation 27, $\prod_{ij}$ is the viscous pressure, which accounts for the frictional interaction between the hair strands. We are free to design it to suit our purpose, as it is completely artificial, taking inputs from the artificial viscosity form for SPH proposed by [Morris 1995], we set it to

$$\begin{aligned}\prod_{ij} &= \begin{cases}\frac{-c\mu_{ij}}{\bar{\rho}_{ij}} & \text{if } \mu_{ij} < 0\\ 0 & \text{if } \mu_{ij} \geq 0\end{cases}\\ \mu_{ij} &= h\frac{\mathbf{v}_{ij}\cdot\mathbf{r}_{ij}}{|\mathbf{r}_{ij}|^2 + h^2/100}\\ \bar{\rho}_{ij} &= (\rho_i + \rho_j)/2 \qquad (28)\end{aligned}$$

Here, the constant c is the speed of sound in the medium. However, in our case, it is just an animation parameter. We are free to set this to an appropriate value that obtains satisfactory visual results. The term incorporates both bulk and shear viscosity, and in totality accounts for all the dissipative interactions amongst the hair strands due to the friction and the boundary layer around the hair strands.

At each step of the integration, first we obtain the density at each particle ρ_i using Equation 26. To correct numerical errors from time to time, we use Equation 21. The only unknown quantity so far is the pressure at each particle p_i. Once we know the particle densities ρ_i, the equation of the state (Equation 3), directly gives the unknown pressure. This is the central theme of the algorithm. Subsequently, we compute the fluid forces acting on each particle using the momentum equation (Equation 27). We know now the interaction forces $\hat{\mathbf{f}}_{ci}$ for each hair segment and we are ready to integrate the equation of the motion for individual hair strand, which is covered in Section 6.

The complete validation of the model remains to be done through systematic virtual experiments, backed by empirical study. We believe that the model has good scientific potential – we leave this aspect as a future work. However, the model in the existing form is adequate to capture the hair-hair interactions for the animation purpose.

4 Hair-body Interactions as Fluid Boundary Condition

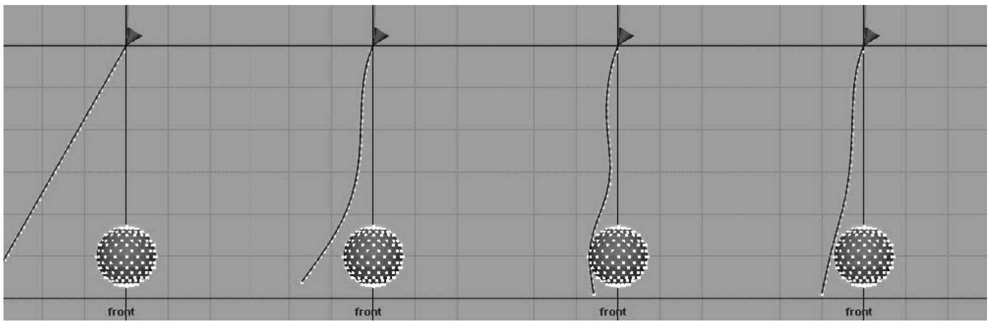


Figure 11: Hair-obstacle Collision – Interaction with Boundary Particles

As discussed in Section 1, we model hair-body interactions as the fluid boundary condition. It is quite straightforward to model solid boundaries, either stationary or in motion, using special boundary particles. We place the boundary particles along the geometry as shown in Figure 11. The boundary particles do not contribute to the density of the fluid and they are inert to the forces coming from the fluid particles. However, they exert a boundary force onto the neighboring fluid particles. A typical form of boundary force is as follows and is given by Morris [Morris 1995]. Each boundary particle has an outward pointing unit normal $\mathbf{n}$ and exerts a force

$$\begin{aligned}\mathbf{f}_n &= K_n \;\; f_1(\triangle\mathbf{r}\cdot\mathbf{n}) \;\; P(\triangle\mathbf{r}\cdot\mathbf{t}_r) \;\; \mathbf{n}\\ \mathbf{f}_t &= -K_f \;\; |\mathbf{f}_n|/K_n \;\; (\triangle\mathbf{v}\cdot\mathbf{t}_v) \;\; \mathbf{t}_v \qquad (29)\end{aligned}$$

where, $\triangle\mathbf{r}$ is the position vector from the boundary particle to the colliding fluid particle. The tangent $\mathbf{t}_r$ is the unit projection of the position vector $\triangle\mathbf{r}$ onto the tangent plane at the position of the boundary particle. Similarly, the tangent $\mathbf{t}_v$ is the unit projection of the relative approach velocity of the fluid particle $\triangle\mathbf{v}$, onto the tangent plane at the position of the boundary particle. Function f_1 is any suitable unit function, which will repel the flow particle away. P is Hamming window, which spreads out the effect of the boundary particle to neighbouring points in the boundary. That way, we have a continuous boundary defined by discrete set of boundary particles. The coefficient of friction K_f determines the extent of the tangential flow slip force $\mathbf{f}_t$, whereas coifficient K_n determines the extent of collision force $\mathbf{f}_n$. Figure 11 demonstrates that the boundary particles are quite effective in achieving collision response. As the boundary particle method is within the framework of SPH, one need not use exclusive collision detection and response for the purpose. However, this method have a significant drawback. The method works effectively only if the boundary particles are placed uniformly on the obstacle geometry. Secondly, if the separation between the boundary particles is large, hair will slip into the boundary though the separation. These drawbacks demand extra work by animators to define a clean uniform geometry as the boundary particles are placed on the vertices of the geometry. We developed a more detailed collision detection and response technique, although the theme remains the same – penalize the smoothed particles approaching the boundary with a collision force.

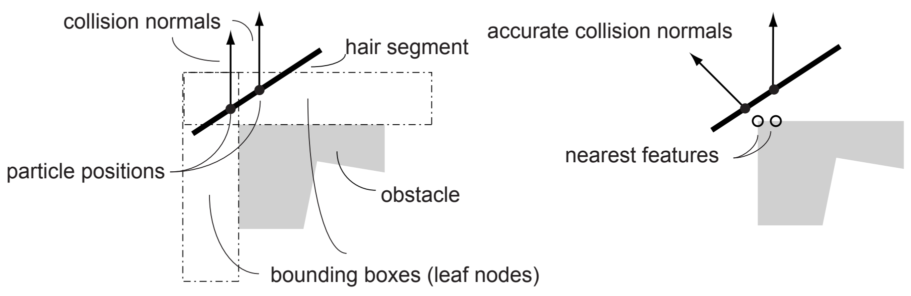


Figure 12: Inaccurate vs Accurate Collision Normal and Nearest Feature

Computer graphics has extensive methods for collision detection. For the state-of-the-art in collision detection, we refer to the nice overview by Lin *et al* [Lin and Gottschalk 1998]. Choosing a right collision detection strategy for hair simulation required a lot of deliberation. The large number of hair strands in the simulation (typically 5,000 to 25,000) pose the challenge. We need to detect the collision of the very large number of smoothed particles with large number of mesh polygons. Many popular collision detection techniques such as AABB Tree and OBB Tree methods are local in nature. They only may detect the proximity of the particle with a mesh polygon and particularly fail to give exact collision normal. One can assign the normal of the colliding mesh polygon as the collision normal, as shown in Figure 12a. However, the collision normal should point away from the nearest feature of the colliding polygon, as shown in Figure 12b. The accurate collision normal is required for effective computation of the collision force and particularly the frictional force. The closes feature tracking method [Cohen et al. 1995] gives the accurate nearest feature of mesh, be it a polygon, an edge or a vertex. However, this method is quite slow for our purpose.

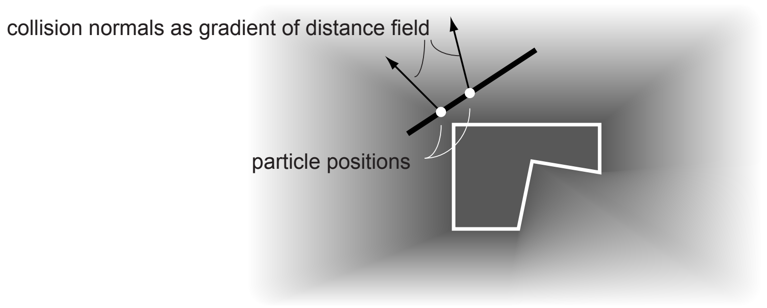


Figure 13: Collision Detection using Distance Field

We use novel *adaptively sampled distance field* (ADF) method by Frisken *et al* [Frisken et al. 2000]. Distance field is the scalar field in a region surrounding the obstacle. Figure 13 illustrates the distance field around the obstacle. The value of the distance field at any point in the region defines the nearest distance from that point to the obstacle geometry. We would like to highlight a property of the distance field – the gradient of the distance field at any point always points away from the obstacle. Whereas, the scalar field value directly determines how close is the point from the obstacle. As shown in the figure, once we determine that the particles are colliding by examining the distance field at the particle positions, the collision normals can be assigned to the gradients of the distance field at those positions. ADF encodes the distance field in a very memory efficient manner. Further, evaluating ADF at arbitrary point along with the gradient is very fast. Constructing the ADF is computationally intensive, which we do it for each time step. However, the overall method of determining collisions of smoothed particles along with collision normals is very fast and accurate using the ADF. For the numerous details on constructing ADF, we refer to [Frisken et al. 2000].

For the collision response, we use simple *penalty method.* The collision force is similar to Equation 29 and is given by the following equation

$$\begin{aligned}\mathbf{f}_n &= K_n \;\; f_1(r) \;\; \mathbf{n} \\ \mathbf{f}_t &= -K_f \;\; |\mathbf{f}_n| / K_n \;\; (\triangle\mathbf{v} \cdot \mathbf{t}_v) \;\; \mathbf{t}_v \end{aligned} \tag{30}$$

where, r is the nearest distance from the fluid particle to the boundary, which is directly read from the ADF. $\mathbf{n}$ is the unit collision normal defined as the gradient of the distance field at the fluid particle position. Whereas $\mathbf{t}$ is unit projection of approach velocity of the fluid particle $\triangle\mathbf{v}$ onto the tangent plane. After experimentation with various forms of the penalty function f_1, we found Perlin's *gain* function [Perlin 1985] to be most suitable. The following figure illustrates the gain function

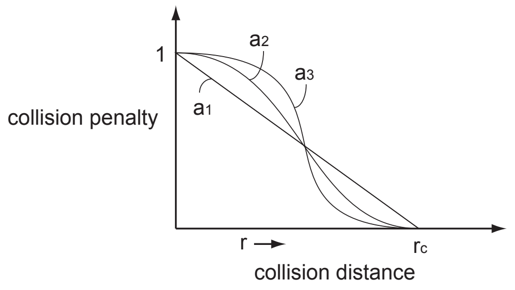


Figure 14: Penalty Function as Perlin's gain function

The collision penalty is zero for r above the collision distance r_c, whereas for $r < r_c$ it increases to one. One can adjust how fast the penalty increases with the decrease in r by varying the gain parameter a.

5 Hair-air, a mixture

We considered hair-hair and hair-body interactions in Sections 3 and 4. In this section, we address the hair-air interactions. Hair-air interactions are important for the following reasons:

- We would like to animate hair blown by wind.
- As the mass of an individual hair strand is very small compared to the skin friction drag created by its surface, the air drag is quite significant. Thus, most of the damping in hair dynamics comes from the air drag. The internal damping pertaining to dissipation in the deformation is quite negligible as compared to the air drag.
- Air plays a major role in hair-hair interaction. As a hair strand moves through air, it generates a boundary layer, which influences the neighboring hair strand even if the physical contact is minimum.
- Most importantly, hair volume affects the air field. Hair volume is not completely porous. Thus it acts as a partial obstacle to the wind field, to alter it.

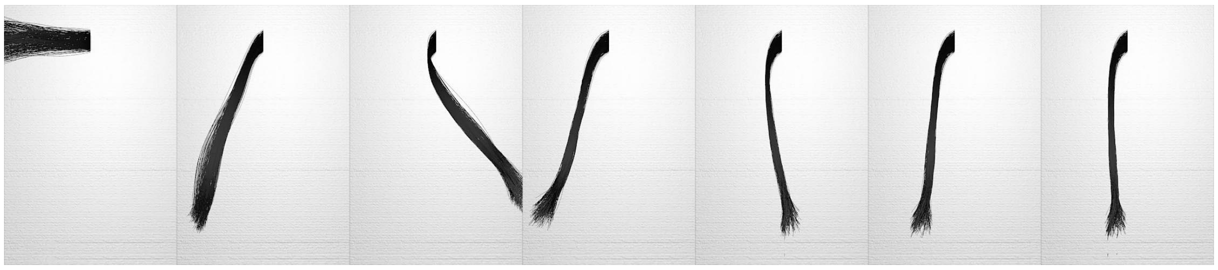


Figure 15: Cantilever of Hair – Moderate Air Drag

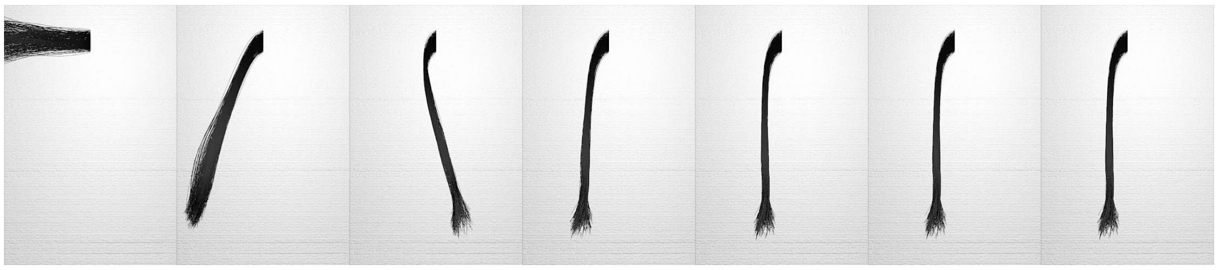


Figure 16: Cantilever of Hair – High Air Drag

Initially we thought of a very simple model for adding wind effects in the hair animation. There are significant advances in computer graphics for modeling turbulent gaseous fields [Stam 1997]. These models are mostly empirical. We incorporate the effect of the field by adding extra force to each of the smoothed particles $\hat{\mathbf{f}}_{di} = \mu(\hat{\mathbf{v}_{w}}_{i} - \hat{\mathbf{v}}_i)$ in addition to the interaction force $\hat{\mathbf{f}}_{ci}$. Here, $\hat{\mathbf{v}_{w}}_{i}$ is the local wind velocity at particle i, expressed as the spatial vector and $\hat{\mathbf{v}}_i$ is the velocity of the particle i. μ is the drag coefficient, which is an animation parameter. Figures 15 and 16 illustrate how variation in the drag coefficient affects the motion of hair. For that purpose, we let the bunch of hair fall under gravity undergoing cantilever action. The two examples illustrate the successive frames of the animations corresponding to each cycle of the oscillation. Observe that in the first example, the bunch of hair makes two oscillations before coming to rest, whereas in the second example it makes hardly one oscillation. The air drag coefficient μ is double in the second example. In both the examples, hair exhibit high degree of damping as seen in real-life.

However, this strategy is a passive one. As mentioned early in the section, hair volume should also affect the wind field for more realistic animations, as it is not completely porous. Subsequently, we extend the hair continuum model. We postulate that the hair medium is a mixture of hair material and air. There are many ways to model a mixture; we model it in a very straightforward way. Let the two fluids, hair medium and air, have their own fluid dynamics. We just link the two by adding extra drag force to the momentum equation. Thus, the SPH forms of the equations for the hair-air mixture are:

$$\frac{d\rho_i^h}{dt} = \sum_{j=1}^{N} m_j^h (\mathbf{v}_i^h - \mathbf{v}_j^h) W_{ij} \quad \text{hair continuity} \tag{31}$$

$$\frac{d\rho_k^w}{dt} = \sum_{l=1}^{M} m_l^w (\mathbf{v}_k^w - \mathbf{v}_l^w) W_{kl} \quad \text{air continuity} \tag{32}$$

$$\frac{d\mathbf{v}_i^h}{dt} = \frac{\mu(\mathbf{v}^a(\mathbf{r}_i^h) - \mathbf{v}_i^h)}{m_i} - \sum_{j=1}^{N} m_j^h \left(\frac{p_j^h}{{\rho^h}_j^2} + \frac{p_i^h}{{\rho^h}_i^2} + \prod_{ij}^{h}\right) \nabla_i W_{ij} \tag{33}$$

$$\frac{d\mathbf{v}_k^a}{dt} = \frac{\mu(\mathbf{v}^h(\mathbf{r}_k^a) - \mathbf{v}_k^a)}{m_k} - \sum_{l=1}^{M} m_l^a \left(\frac{p_l^a}{{\rho^a}_l^2} + \frac{p_k^a}{{\rho^a}_k^2} + \prod_{kl}^{a}\right) \nabla_k W_{kl} \tag{34}$$

Observe that there are two different sets of smoothed particles corresponding to two constituents of the mixture. $\mathbf{v}^a(\mathbf{r}_i^h)$ is air velocity experienced by the hair particle i, *i.e.* the velocity estimate of air at point $\mathbf{r}_i^h$ and vice versa for the air particle. Thus there is a coupling by the drag coefficient μ between the two fluid dynamics. For zero drag coefficient, hair and air will move without affecting each other. This model, although computationally expensive, results in very good animation of hair blown by wind – discussed in Section 8.

6 Numerical Integration

We assume that the reader is already conversant with numerical integration issues. For an extensive discussion on numerical methods we refer to standard text book – "Numerical Recipes in C: The Art of Scientific Computing" [Press et al. 1993]. The sole purpose of developing hair stiffness dynamics as dynamics of serial rigid multi-body chain is to have dynamical equations which are non-stiff. Thus it is quite sufficient to use an explicit numerical integration scheme. We use fifth order Runge-Kutta integration method with adaptive time stepping via error control [Press et al. 1993] for the purpose. The higher order integration scheme is not only more accurate, it has even better stability. So the user can choose large time steps which compensate for the extra computational overheads involved in higher order schemes. As the animator is readily given visual feedback, she can immediately judge the discrepancy in the results due to instability. She thus can choose appropriate constant time step. The more technically oriented animator, can have a more detailed control over the time step by setting appropriate minimum and maximum bounds on the time step. The adaptive integrator will choose the time step in the user specified range after analyzing the error estimate arising from the previous time step. Figure 17 illustrates the typical instability in the hair animation.

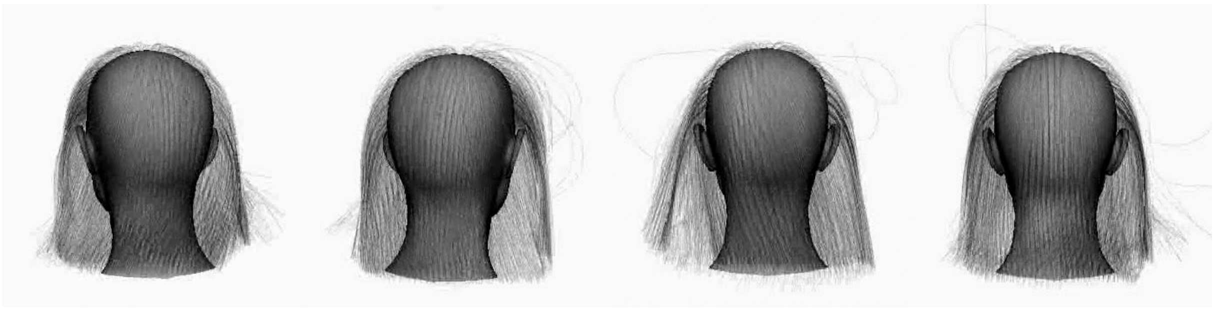


Figure 17: Instability in Hair Dynamics

Interestingly, we have observed that in the current implementation, where we use floating point precision, the user can not set the hair segment length l less than one centimeter for earth's gravity of 980cm/sec^2, no matter how small the time step is. We suspect this is related to numerical precision rather than the stability region of the dynamics.

However, the choice of explicit integration method has a major drawback. These methods have very narrow stability region. To achieve non-straight neutral shape of the hair we need to assign considerably high bending and torsional rigidities. Moreover, hair motion is highly damped demanding high degree of air-drag. Using the explicit integration methods, the hair simulation is almost always operating near bounds of the stability region. As a result, we are able to simulate straight to wavy hair, even though the model is able to accommodate the case of very curly hair.

We would like to use implicit integration methods, which we leave it as future work. In the case of chosen stiffness dynamics model, it is non-trivial to formulate the implicit integration schemes. It is not possible to express the Jaccobian of the stiffness dynamics using Featherstone's algorithm. We need to investigate alternative implicit methods that use only approximate estimation of the Jaccobian. We list this drawback in detail in the concluding section along with the future research possibilities.

Another source of instability that might occur is due to usage of relatively crude penalty method for the collision response. If hair moving with high velocity collides the boundary, one needs to apply large penalty force to avoid the penetration. This would demand smaller time steps to avoid instability. However, we would like to point out that the collision avoidance is part of the fluid boundary condition. Thus as the hair strand approaches the boundary, hair-hair interaction "makes the strand aware of the boundary" due to the fluid dynamic forces. Thus the hair-body collision is never a hard collision. In future we would like to replace the collision response by a more elaborate method – impulse dynamics. We full heartedly refer to the pioneering work by Brian Mirtich [Mirtich 1996], where he integrated the rigid-body impulse dynamics into the rigid multi-body dynamics framework.

7 Implementation Issues

The developed models for individual hair dynamics and hair-hair, hair-body, hair-air interactions are quite elaborate. Naturally, they are computationally intensive. Thus, a meticulous implementation is desired to make the methodology viable even with today's ever growing desktop computing power. In this section we discuss some of the implementation issues.

7.1 Data-parallel Implementation of Single Hair Dynamics

Modeling a hair strand as a rigid multi-body serial chain accurately captures all the relevant modes of motion and stiffness dynamics. Formulating only the exact number of relevant DOFs, *i.e.* bending and torsion, we have removed the source of the stiff equations of motion associated with the high tensile rigidity of the hair strand. Hence, we obtain an advantage in terms of possible large simulation time steps, even though the dynamics calculations are a bit involved.

We keep the length of the hair segment per hair strand constant. We also align the hair segment's local coordinate system to the principal inertial axis. That way the 6x6 spatial inertia tensor takes a simple form with many zeros and is constant. Length being constant, the only variable part in the coordinate transformation, from one link to another, is rotation. Exploiting the special multi-body configuration of hair, we have symbolically reduced most of the Articulated Rigid Body Dynamics calculations and have fine-tuned it to be most efficient. The time complexity of algorithm is linear. This puts no restriction on the number of rigid segments we can have per hair strand.

Finally, we parallelize the task of the hair strand computation. We exploit four-way parallel Single Instruction Multiple Data (SIMD) capability of Pentium III processors. We sort the hair strands by their number of hair segments. Then we club four hair strands, equal in number of segments, as far as possible or we trim a few. We then can compute four hair strands at a time on a single processor. Additionally, we assume two to four CPUs are available to use. Using these strategies, we are able to simulate 10,000 hair strands, having 30 segments on an average, in less than 2 minutes for each frame.

7.2 Efficient implementation of Smoothed Particle Hydrodynamics

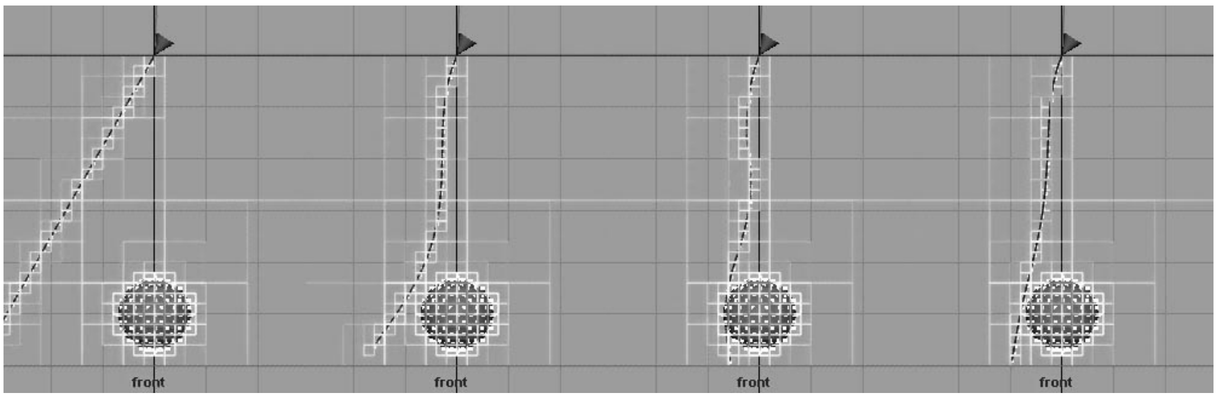


Figure 18: Hair-obstacle Collision – Use of Octree to Track Particle Interactions

The smoothed particle's kernel has compact support. The particle influences only its near neighbours, more precisely the ones that are in the circle of smoothing length. Thus the time complexity of the fluid computation is O(kn), where n is the total number of particles and k is the typical number of particles coming under influence of one particle. We still have to ensure that we use an efficient algorithm to locate the neighbours. There are many strategies for collision detection and neighbour search. For a detailed survey, refer to Lin and Gottschalk [Lin and Gottschalk 1998]. However, smoothed particles being point geometries, we use the Octree space partitioning. Vemuri *et al* [Vemuri et al. 1998] used the Octree for granular flow which is very similar to our application.

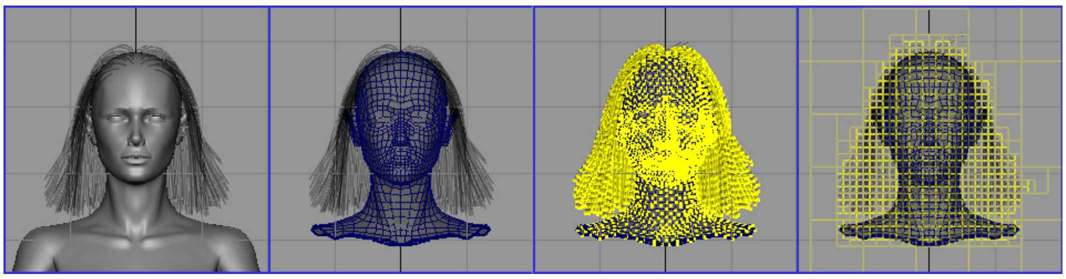


Figure 19: Approximate geometry, Smoothed particles, Octree

8 Results

We report three short animations using the described methodology. They are in increasing order of scene complexity. However, they utilize the same underlying models discussed so far. The simplest of the animations highlight a multitude of the dynamics in minute detail and the more complex ones illustrate the effectiveness of the methodology in animating real life hair animations. In the end we discuss animating hair for a dance sequence.

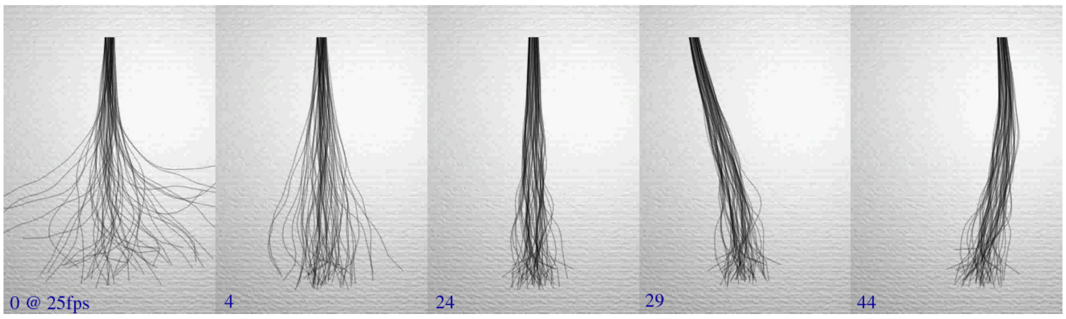


Figure 20: Starting from the initial spread, individual hair strands collapse under gravity. As they get close, the pressure built up in the “hair fluid” retains the volume (frame 24). In the subsequent frames, the body forces and hair-air interaction is prominent

In the first animation, from the initial spread, individual hair strands collapse under gravity. Hair strands have their shape memory working against gravity. Otherwise they would have straightened up at frame 24. Also, as the hair strands get close, the pressure builds up due to increase in the number density in the ”hair fluid”, which further retains the volume, throughout the animation, by keeping individual hair apart. The inertial forces and the influence of air are evident in the oscillatory motion of hair. The air drag is most effective towards the tip of hair strands. Observe the differential motion between the tips. Hair strands on the periphery experience more air drag than the interior ones. This is only possible due to the fluid-hair mixture model; the movement of hair does set air in motion like a porous obstacle.

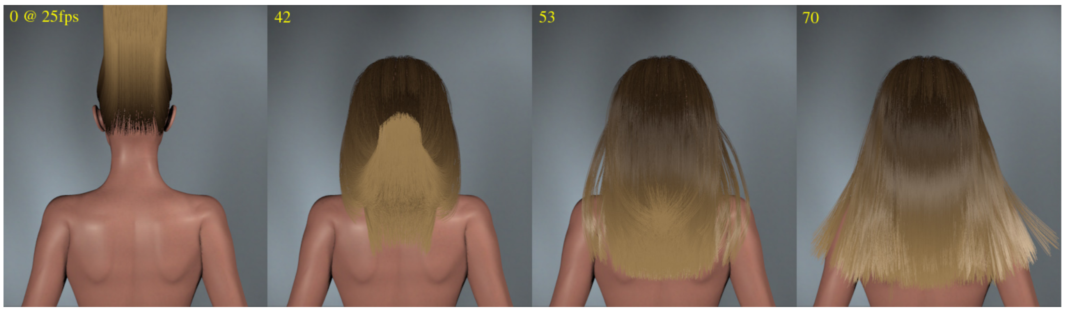


Figure 21: Free fall of hair – Hair volume, modeled as fluid, falls freely under gravity. However, the individual hair’s length constraint quickly restricts the free falling motion to give it a bounce. At the same time, “hair fluid” collides with the body and bursts away sidewise.

The second animation scenario is to illustrate the “fluid” motion of hair without loosing the character of individual hair. The hair volume starts falling freely under gravity. Quickly, the individual hair’s length constraint and stiffness restricts the free falling motion to give it a bounce, towards the end of the free fall (frame 53). At the same time, ”hair fluid” collides with the body and bursts away sidewise (frame 70). The air interaction gives an overall damping. Observe that the hair quickly settles down, even after the sudden jerk in the motion, due to air drag and hair friction with the body.

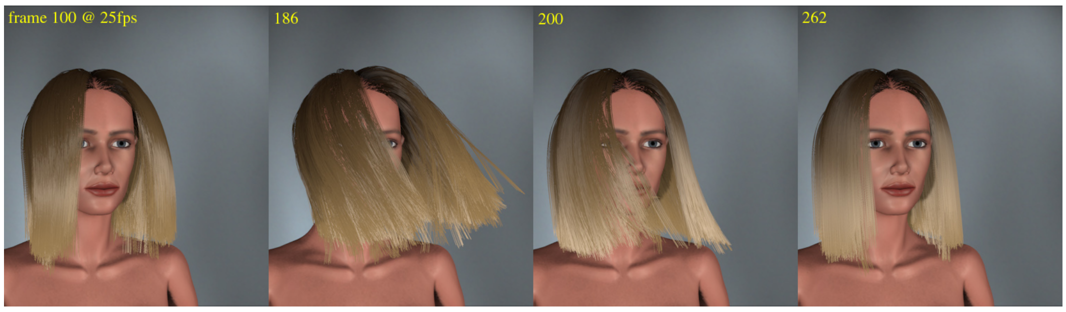


Figure 22: Hair blown by wind – The “fluid hair” model is extended to “hair-air mixture”. Indeed, the complex hair-hair, hair-air and hair-body interactions are modeled under a single framework.

The third animation in Figure 22 exclusively illustrates the effectiveness of the model in animating hair blown by wind. Needless to say that there is an influence of airfield on individual hair. More importantly, body and hair volume acts as a full and partial obstacle to air altering its flow.

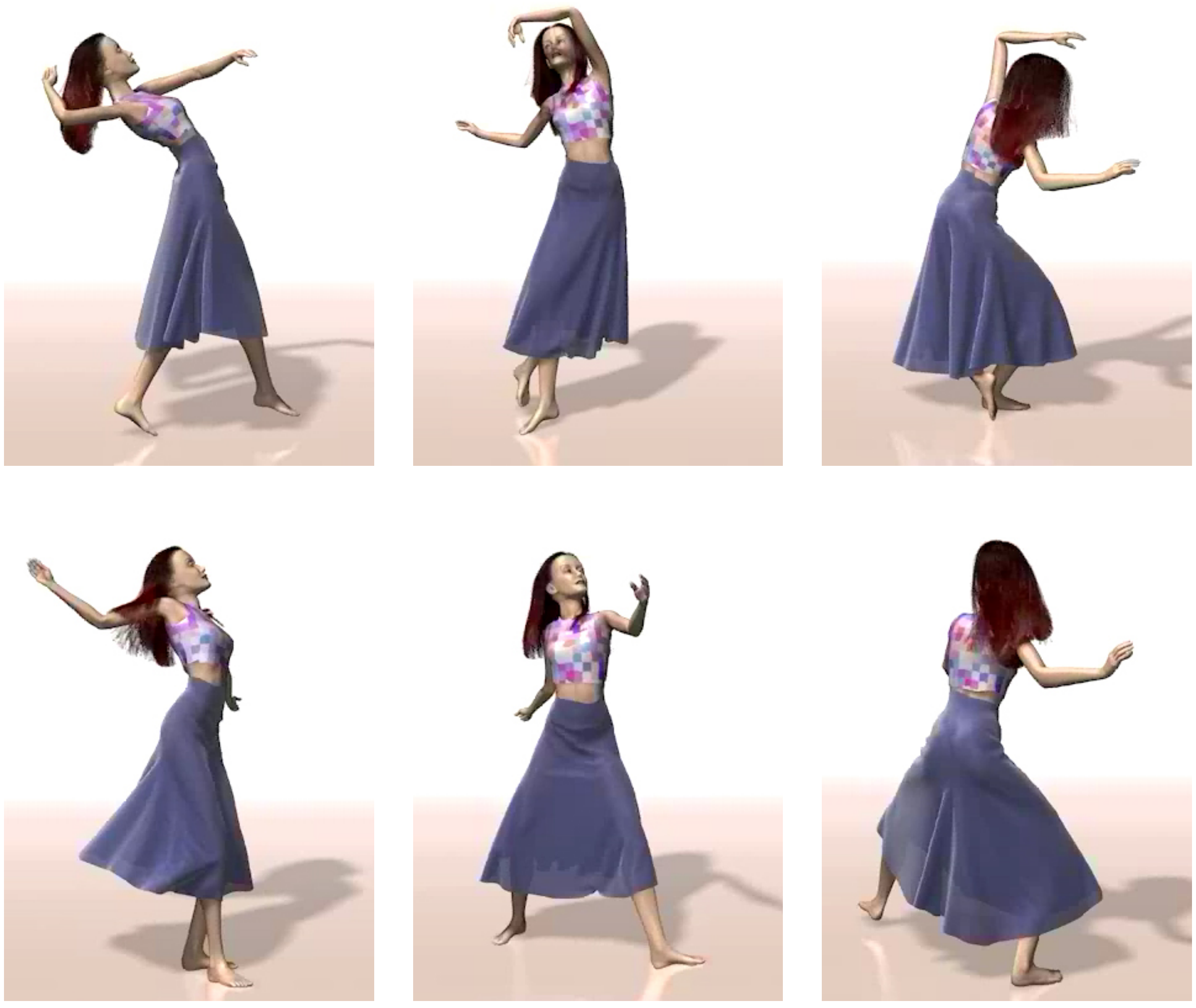


Figure 23: Dance Sequence Demonstrating Hair Animation by Nedjma Kadi and Sunil Hadap

The animation methodologies are implemented in a Maya plugin – MIRAHairSimulation. We next present a representative animation sequence which is the result of typical usage of MIRAHairSimulation by animators at MIRALab, University of Geneva.

Figure 23 is a dance sequence of around 1 minute. The dance is motion captured using Vicon8 optical motion tracking system. The animators used 3ds max for setting up the body deformations and the Fashionizer, MIRALab's flagship cloth simulation system for achieving the cloth animation. The resulting animated mesh sequence was imported into Maya for adding hair animation. The process of setting up hair animation and computing hair simulation including hair rendering towards the satisfactory results took around 2 weeks. The dynamic hairstyle has around 8,000 hair clumps and the total number of polygons for the dress and the body is around 20,000. It took on an average 243 seconds for computing one frame of the animation, whereas the rendering of the sequence took on an average 370 seconds per PAL frame. We used RenderMan for rendering of the dance sequence. The simulation was computed on a workstation having Intel Xeon 2.2MHz processor with 2GB RAM.

9 Summary

We have developed a powerful hair dynamics model.

- Stiffness Dynamics – We have given an elaborate model for the stiffness and inertial dynamics of an individual hair strand. We treat the hair strand as a serial rigid multi-body system. This reduced coordinate formulation gives very accurate and effective representation for the dynamics of non-straight (wavy) hair by providing precise parametric definition of the bending and the torsion in three dimensions. The formulation also partly eliminates the stiff numerical equations enabling large time-steps, thus faster simulations.

- Interaction Dynamics – The hair-hair, the hair-air interactions and the accurate hair-body collisions were one of the few unsolved problems in computer graphics – until recently. We have exclusively addressed this problem by making a paradigm shift and treating hair as a continuum. We model the hair-hair, the hair-body and the hair-air interactions in a unified way using fluid dynamics. The continuum assumption proves to be a very strong model for otherwise very complex interaction phenomena.

10 Limitations and Future Work

- We have successfully attempted to capture the detailed dynamics of straight to wavy hair typically found in moderately complex hairstyles. However the problem of animating complex hairstyles, i.e. the hairstyles involving curly hair or the hairstyles having intricate geometry, still eludes us. By adopting the reduced coordinate formulation, we hoped to have completely eliminated the stiff differential equations. However, we learned that very high bending and torsional rigidity is required to firmly maintain the intricate geometric definition of the complex hairstyle, under gravity. At the same time the hair motion is highly damped. Both these problems clearly put the inherent stability of the implicit integration methods in high demand.

  Unfortunately, it is not possible to express the Jaccobian of the system using the articulated rigid body dynamics that we have used. This is due to the iterative nature of the inertial dynamics formulation. In future we would like to explore the possibility of expressing the approximate Jaccobian that corresponds to the stiff part of the differential equations and use implicit integration methods based on that.

  Another possibility is to follow the constrained dynamics. The constrained dynamics has the best possibilities of using implicit integration methods. These methods may also provide better collision response possibilities. However, the constrained dynamics methods severely suffer from the numerical inaccuracies in terms of drift in the constraints. One needs to use sophisticated numerical methods to avoid the problem associated with the drift.

  The original idea of using the spring-mass-hinge systems still remains to be one of the strong possibilities we would like to explore. The current advances in the implicit integration methods fade away the limitations of these methods of being "stiff" in nature due to elongation constraints that are expressed as stiff springs. However, we would have to investigate the ways of expressing the stiffness dynamics of bending and torsion, in the framework of the spring-mass-hinge system, which in our primary opinion is complex.

- We have implemented the collision response which is essentially the penalty method. This method does not have good stability characteristics. We have barely managed to handle various hair-body collision situations arising in real-life complex motions such as the dance sequence.

  We would like to explore the Brian Mirtich's work of impulsed based collision response [Mirtich 1996], which is unconditionally stable, although it is numerically expensive.

References

Anjyo, K., Usami, Y., and Kurihara, T. 1992. A simple method for extracting the natural beauty of hair. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques (SIGGRAPH)*, ACM SIGGRAPH.

Baraff, D. 1996. Linear-time dynamics using lagrange multipliers. *Proceedings of SIGGRAPH 96* (August), 137–146.

Chang, J., Jin, J., and Yu, Y. 2002. A practical model for mutual hair inteactions. In *Proceedings of Symposium on Computer Animation*, ACM SIGGRAPH, San Antonio, USA.

Cohen, J., Lin, M., Manocha, D., and Ponamgi, K. 1995. I-collide: An interactive and exact collision detection system for large-scaled environments. In *Proceedings of ACM Symposium on Interactive 3D Graphics*, 189–196.

Daldegan, A., Magnenat-Thalmann, N., Kurihara, T., and Thalmann, D. 1993. An integrated system for modeling, animating and rendering hair. *Computer Graphics Forum, Proceedings of Eurographics 12*, 3, 211–221.

Featherstone, R. 1987. *Robot Dynamics Algorithms.* Kluwer Academic Publishers.

Frisken, S., Perry, R., Rockwood, A., and Jones, T. 2000. Adaptively sampled distance fields: A general representation of shape for computer graphics. In *Proceedings of ACM SIGGRAPH*, 249–254.

Gascuel, J., Cani, M., Desbrun, M., Leroy, E., and Mirgon, C. 1996. Smoothed particles: A new paradigm for animating highly deformable bodies. In *6th Eurographics Workshop on Animation and Simulation'96*, Paris.

Hadap, S. 2003. *Hair Simulation.* PhD thesis, MIRALab, CUI, University of Geneva. No 3416, Science Faculty.

Hockney, R. W., and Eastwood, J. W. 1988. *Computer Simulation Using Particles.* Adam Hilger, March. ISBN: 0852743920.

Lee, D.-W., and Ko, H.-S. 2001. Natural hairstyle modeling and animation. *Graphical Models 63*, 2 (March), 67–85.

Lin, M., and Gottschalk, S. 1998. Collision detection between geometric models: A survey. In *Proceedings of IMA Conference on Mathematics of Surfaces.*

Magnenat-Thalmann, N., Hadap, S., and Kalra, P. 2000. State of the art in hair simulation. In *Proceedings of International Workshop on Human Modeling and Animation*, Korea Computer Graphics Society, Seoul, Korea, 3–9.

Mirtich, B. 1996. *Impulse-based Dynamic Simulation of Rigid Body Systems.* PhD thesis, University of California, Berkeley.

Monaghan, J. J. 1992. Smoothed particle hydrodynamics. *Annual Review of Astronomy and Astrophysics 30*, 543–574.

Morris, J. P. 1995. An overview of the method of smoothed particle hydrodynamics. AGTM Preprints, University of Kaiserslautern.

Panton, R. L. 1995. *Incompressible Flow*, 2nd edition ed. John Wiley, December.

Perlin, K. 1985. An image synthesizer. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques (SIGGRAPH)*, ACM SIGGRAPH, 287–296.

Plante, E., Cani, M.-P., and Poulin, P. 2001. A layered wisp model for simulating interactions inside long hair. In *Proceedings of Eurographics Workshop, Computer Animation and Simulation*, EUROGRAPHICS, Manchester, UK.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. 1993. *Numerical Recipes in C: The Art of Scientific Computing*, 2nd edition ed. Cambridge Univ Press.

Rosenblum, R., Carlson, W., and Tripp, E. 1991. Simulating the structure and dynamics of human hair: Modeling, rendering and animation. *Journal of Visualzation and Computer Animation 2* (June), 141–148. John Wiley.

Stam, J. 1997. Stochastic dynamics: Simulating the effects of turbulence on flexible structures. *Computer Graphics Forum 16*, 3 (August), 159–164.

Vemuri, B. C., Chen, L., Vu-Quoc, L., Zhang, X., and Walton, O. 1998. Efficient and accurate collision detection for granular flow simulation. *Graphical Models and Image Processing 60*, 5 (November), 403–422.

Efficient and Realistic Hair Animation Using Sparse Guide Hairs

Yizhou Yu   Johnny T. Chang   Jingyi Jin
Department of Computer Science, University of Illinois at Urbana-Champaign
1304 West Springfield Avenue, Urbana, IL 61801
E-mail: {yyz,jtchang,jingjin}@cs.uiuc.edu

Abstract

This paper presents a framework for animating curly hairs using a sparse set of guide hairs. Before animation, an initial static sparse hair model is extracted first. It can be effectively interpolated to produce a dense hair model. Random natural or artificial curliness can then be added to the dense model through a parametric hair offset function with a randomized distribution of parameters over the scalp.

Hair exhibits strong anisotropic dynamic properties which demand distinct animation techniques for single strands and hair-hair interactions. While a single strand can be modeled as a multibody open chain expressed in generalized coordinates, modeling hair-hair interactions is a more difficult problem. A dynamic model for this purpose is proposed based on a sparse set of guide strands. Long range connections among the strands are modeled as breakable static links formulated as nonreversible positional springs. Dynamic hair-to-hair collision is solved with the help of auxiliary triangle strips among nearby strands. Adaptive guide strands can be generated and removed on the fly to dynamically control the accuracy of a simulation.

Fine imagery of the final dense model is rendered by considering both primary scattering and self-shadowing inside the hair volume which is modeled as being partially translucent.

Index Terms: Hair Modeling, Guide Hairs, Hair Interpolation, Offset Functions, Hair Animation, Hair-Hair Interaction, Static Links, Collision Detection, Open Chain, Hair Rendering

1 Introduction

Hair is a crucial element of appearance. One of the many challenges in simulating believable virtual humans and animals has been to produce realistic looking hair. Creating realistic hair presents problems in all aspects of computer graphics technologies, i.e. shape modeling, dynamics and rendering [14, 29, 19, 36, 1, 6, 10, 38, 16, 11, 21, 12, 9, 20, 18, 28, 40, 4]. Hair rendering and shape modeling of fur like short hair is becoming increasingly available to animators. However, shape modeling and dynamics of long hair has been difficult. The difficulties arise from the number of hair strands, their geometric intricacies and associated complex physical interactions such as collisions, shadowing and static charges. These interactions contribute to both static and dynamic appearances of hair.

The most important thing in a hairstyle lies in the way hair strands curve and deform. The causes for hair deformation can be summarized as natural curliness, artificial hairstyling processes and deformation under external forces such as gravity, collision and static charges. According to [13], the degree of natural curliness is indicated by the curvature of a hair strand when it is not under any external forces. In an actual hairstyling process, many artificial procedures are provided by a hairdresser, such as perming, combing, shearing and the application of cosmetics. The effects of perming and cosmetics are most influential to cause long-term artificial hair deformation. They overcome natural curliness and make hairs hard to model. Lastly, hairs bend under gravity and collision. These factors should be considered in hair dynamics. Because of artificial hairstyling and external forces, hair strands do not grow and curve completely randomly in all directions, but follow certain global as well as local flow patterns. For example, long hair is usually draped down, and nearby strands usually form a cluster and deform in the same way. However, there is still small amount of randomness that distinguishes strands from each other in the way they curve.

In addition to static modeling, hair also has highly anisotropic dynamic properties, i.e. hair strands are extremely hard to stretch but free to move laterally and interact with each other irregularly. Strands cannot penetrate each other when they intersect; yet, each strand does not have a fixed set of neighboring strands. These unique properties inform us that a custom designed dynamic model is necessary to achieve realistic results. The dynamics of long hair involve three aspects. First, an individual hair strand can deform and interact with the scalp, cloth and other objects. Second, an initial hairstyle can usually be recovered after subsequent head movement and the application of external force fields. This means a hairstyle can memorize its original configuration. Slight movement does not erase this memory. However, radical movement may permanently damage this memory and no complete recovery is possible. Third, there are dynamic collisions among different strands. A real person can have more than 50,000 hairs. Each hair can be modeled as dozens of hair segments. Directly detecting pairwise collisions among hair segments is neither necessary nor computationally practical. Therefore, we should model hair collisions at a higher abstraction level.

In this paper, we present a framework for hair animation. Both modeling and animation start from a sparse set of guide hairs. The modeling approach can generate realistic wavy hairstyles by editing a basic sheared hair model with a generic offset function for curliness. The hair dynamics model has the following features: i) an initial hair connection model that allows hairstyle recovery after minor movement, ii) a hair mutual collision model that considers the hair volume as a collection of continuous strips, iii) an adaptive hair generation scheme to complement our sparse hair model. Since our framework is based on sparse guide hairs, designing hairstyles and solving physical interactions among hairs are computationally efficient without losing much of the quality from a method based on a dense model.

1.1 Related Work

The work we present in this paper has been made possible by previous work on hairs and other related topics. We limit the overview to the previous work on hair modeling and dynamics, focusing on explicit hair models. In these models, each hair strand is considered for shape and dynamics. They are more realistic and especially suitable for long hair.

There are a few methods for hair modeling. Due to effects of adhesive forces, hairs tend to form clumps. Watanabe introduced the wisp model in [36]. Yan *et al* [38] modeled the wisps as generalized cylinders. The wisp model is also used in [5]. Daldegan *et al* proposed to define a few characteristic hair strands in 3D and then populate the hairstyle based on them. Hadap and Magnenat-Thalmann [11] model hairs as streamlines of an ideal fluid flow. The user can set up a few flow elements around an object to design a hairstyle. Other researchers also tried to model and constrain hair using a single thin shell or multiple head hull layers [16, 20].

There has also been a number of proposed methods for hair animation. Rosenblum *et al* [29] and Daldegan *et al* [6] used a mass-spring-hinge model to control the position and orientation of hair strands. Anjyo *et al* [1] modeled hair with a simplified cantilever beam and used one-dimensional projective differential equation of angular momentum to animate hair strand. None of these previous attempts considered hair-hair interactions and hairstyle recovery after minor movement. Recently, Hadap and Magnenat-Thalmann [12] proposed a novel approach to model dense dynamic hair as continuum by using a fluid model for lateral hair movement. Hair-hair collision is approximated by the pressure term in fluid mechanics while friction is approximated by viscosity. This work presented an elegant and promising model for hair interactions using a dense set of strands. Plante *et al* proposed a wisps model for simulating interactions inside long hair [28]. Hair strands are clustered into wisps consisting of a skeleton and a deformable envelope. Collision forces among wisps follow an anisotropic formulation. This model is suitable for hairstyles with independently clustered wisps. Koh and Huang presented an approach by explicitly modeling hair as a set of 2D strips [18]. Collisions between hair strips are handled to create more realistic motion. Nonetheless, the volumetric aspect of the hair is not captured.

Recently, there has been a few feature films, such as Final Fantasy and Monsters Incorporated, with realistic hair simulations as well as some commercial software packages, such as Shave[33] and Shag[32]. Shave is considered as the best commercial hair modeling and simulation software in the industry. However, its single strand dynamics does not look realistic, and it does not have hair-hair collision. Final Fantasy is the film with the best simulations for long human hair. From the press releases, Aki's hair was modeled as a whole deformable exterior surface and some of the simulations were done using the Maya cloth plugin. That means hairs are constrained around the surface to enable very good hairstyle recovery, but much of the lateral freedom has been lost. In some situations, the hair flows like a piece of cloth instead of a set of individual stiff strands. On the other hand, Monsters Incorporated has long fur simulation [9]. Each hair is considered as particles linked in a chain by a set of stiff springs. A builder or a small snippet of code is used to generate the inbetween hairs. Hair-hair collision has not been considered.

1.2 Overview

Let us first clarify the types of hair models used in our system. We use both sparse and dense hair models. A sparse hair model has from dozens to hundreds of guide hairs. A dense hair model has around 50,000 strands which is close to the number of hairs human beings have. A hair model can also be static or dynamic. A static model does not change its geometric configuration over time. Hairs in a dynamic model change their positions and velocities over time when forces are present. Thus, we have four combinations of hair models as shown in Fig. 1(a).

Given a geometric model of a synthetic head, our system first generate a static sparse hair model H_{ss} using interactively defined vector fields and splines. A H_{ss} can be interpolated to generate a static dense hair model H_{sd} which can be further edited to have curliness. The dense curly hair model thus obtained can be rendered to produce final synthetic images (Fig. 1(b)).

A H_{ss} can also go through a different path in the system to produce hair animations. It can be the input to our simulation subsystem. Each guide hair from the H_{ss} is initially represented as a polyline with multiple vertices. The H_{ss} is then equipped with structural elements needed for dynamic simulation. For example, each vertex is considered as a rotational joint with a hinge. Connections and triangular meshes among guide hairs are then built for simulating hair-hair interactions. Such an

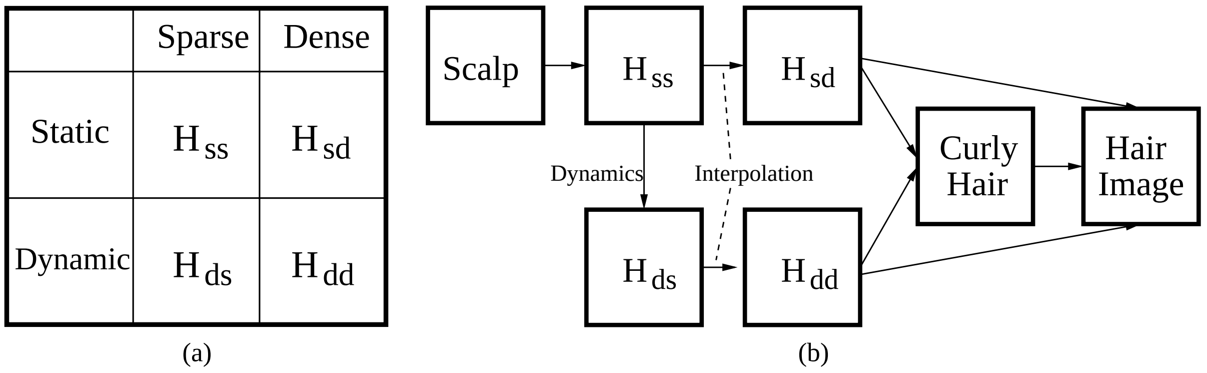


Figure 1: (a) Four types of hair models used in this paper; (b) a diagram of our hair modeling and animation system with various data flow paths.

enhanced model is then ready for dynamic simulation. Note that these enhanced structures are ”invisible”, which means they are never visualized during hair rendering although the effects they produce are incorporated into hair motion. Once an animation sequence of the sparse model is generated, we obtain a dynamic sparse hair model H_{ds} at each frame. A H_{ds} can then be interpolated to produce a dynamic dense model H_{dd} which can be rendered to produce a synthetic image corresponding to a specific frame in an animation sequence. Curliness can certainly be added before rendering as in the static case.

In the rendering stage, we consider both diffuse and specular reflection as well as partial translucency of each strand by integrating volume density rendering with a modified version of the opacity shadow buffer algorithm [17].

Fig. 1(b) shows the various paths in our system. Note that for curly hair, we have two levels of details. The sparse or interpolated hair model only has large-scale deformations without fine curly details. Each strand in these models serves as the spine of its corresponding curly strand. Curliness is added onto the interpolated dense hair model before rendering.

The organization of the rest of the paper is as follows. In the next section, we describe various aspects of hair modeling. Section 3 presents techniques for hair animation. Hair rendering is briefly discussed in Section 4. Section 5 presents our results and comparisons. And Section 6 provides conclusions.

2 Hair Modeling

Previous approaches either model individual hairs explicitly [1, 36, 29, 6, 5] or model all hairs collectively as a volume with a density at each point [14, 27]. We take the explicit approach here, considering the enormous variations of hairstyles. The input to our method is a polygonal model of a head including the scalp and face. Our overall virtual hairstyling process has the following multiple steps:

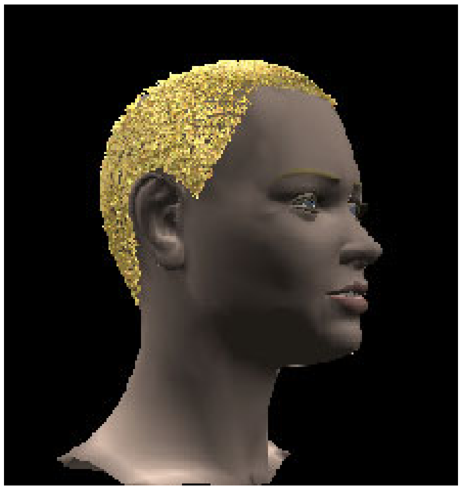
1. select a region on the scalp for hair growth, and specify a length distribution for hairs growing from the region (shearing);
2. model the polylines for a sparse set of guide hairs using a set of interactively defined vector fields, transform the polylines for the guide hairs into spline curves, and further adjust their deformations using their control points;
3. generate a dense hair model from the guide hairs through interpolation;
4. edit each strand in the dense model using an offset function to produce random curliness;
5. pull together nearby hairs to form clusters;

2.1 Shearing

We first need to define the region on the scalp where hair should be grown (Fig. 2(a)). The contour of this region is generated by linearly interpolating the 2D polar coordinates of a few points, which have been interactively selected in a flattened map of the scalp. The center of this map aligns with the top center of the scalp (Fig. 3(c)). Once specified, the same region can be repeatedly used for multiple hairstyles.

Shearing is the starting point of hairdressing. Every hairstyle needs a corresponding hair length distribution. The length of a particular strand is determined by the location of its root on the scalp. We use the vector between the root of the strand and the centroid of the scalp to define this location in a 3D polar coordinate system. Hairdressers sometimes cut hair with respect to some reference planes[39, 31]. For example, they would cut lower part of the hair to a horizontal plane at the bottom of the neck, and pull upper part of the hair straight up and cut it to a horizontal plane on top (Fig. 2(b)). At other times, the hair would be cut to have a smoothly varying length[39, 31].

According to these two shearing styles. We designed two representations for hair length distributions. The first one is based on a BSP tree (Fig. 2(b)). The plane defined for an intermediate node of the BSP divides a region on the scalp into two smaller ones. The plane at a leaf node of the BSP actually defines a reference plane for hair-cutting. All hairs rooted in the region corresponding to that leaf node should be cut to the reference plane given at the same leaf. This is done by setting the length of a strand to be the length of the shortest path outside the scalp between its root and that reference plane. So part of the shortest path may be curved and lying on the scalp. We actually use a sphere to approximate the shape of the scalp to accelerate this calculation. Based on the second shearing style, the second representation is a linear interpolation model based on the length specified at a few key locations on the scalp.


(a)

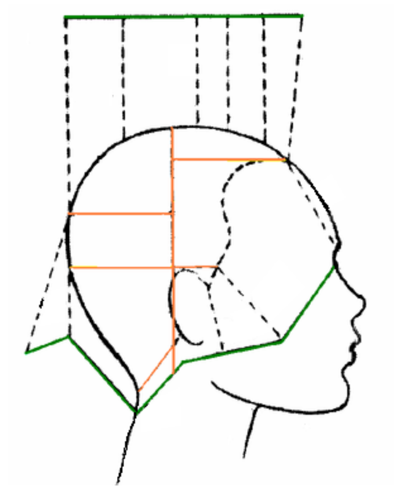
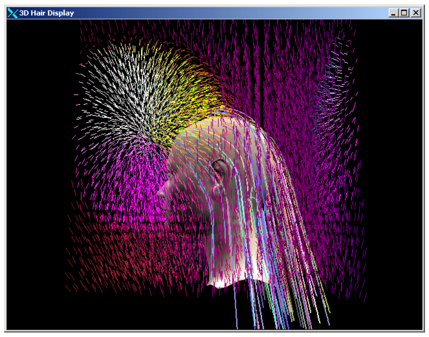
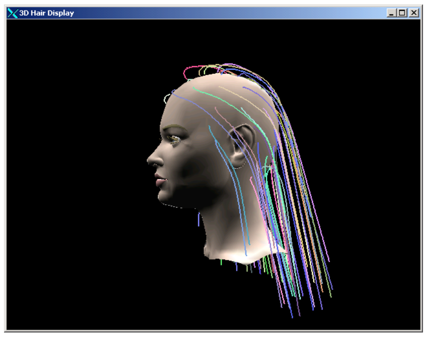

(b)

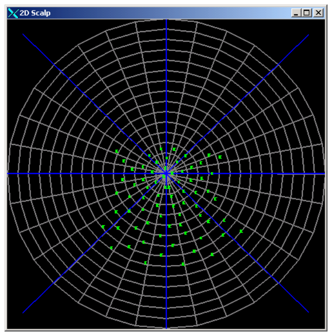
Figure 2: (a) A polygonal head model and the selected region on scalp for hair growth; (b) a length distribution represented with a BSP tree. Orange lines represent intermediate dividing planes; dark green lines represent reference planes at the leaf nodes.



(a)



(b)



(c)

Figure 3: (a) A visualization of a vector field generated from two field primitives. One of the primitives is a vortex in front of the forehead. The orientation at each point is pseudo-colored with brightness indicating field strength; (b) the initial static sparse hair model extracted from the vector field in (a); (c) a user interface for the 2D flattened map of the scalp. The green dots represent the roots of the sparse guide hairs shown in (b).

2.2 Modeling Sparse Guide Hairs

To generate a detailed hair model with around 50,000 strands, we start with a sparse set of guide hairs uniformly distributed over the scalp. Each guide hair is initially represented as a polyline which is obtained by tracing a trajectory through an interactively defined global vector field [34]. To perform the tracing, we start from the root of the guide hair on the scalp. At every step, we generate a new segment of the polyline along the current direction of the vector field. This is repeated until the guide hair has reached its predefined length.

The global vector field is actually a superposition of multiple vector field primitives with local influence. Since this approach is similar to the scheme in [11] which makes use of ideal fluid flow fields, we briefly introduce the concept and our adaptations

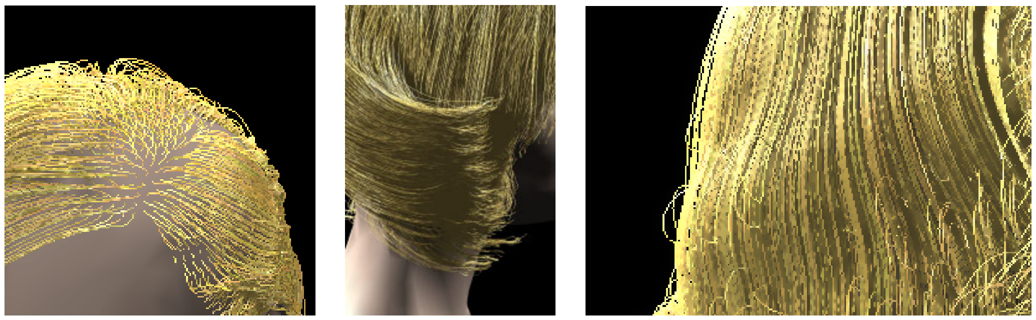

(a) (b) (c)

Figure 4: (a) As shown here, a thin ellipsoidal vector field primitive can be used for dividing hairs; (b) the magnetic field induced by a linear or arc current can be used as a perming roller to wrap hair around; (c) a field primitive that can generate a volumetric wave in the hair.

below. A vector field primitive is a simple vector field with a rigid transformation between the world coordinate system and its local coordinate system. The rigid transformation can be interactively edited to change the position and orientation of the local coordinate system. Each field primitive is responsible for one large feature in the hair flows. For example, we can use one field primitive to bend hairs towards the back of the head; or make the hairs on the forehead curve differently from those on the back of the head.

The orientation and strength of a vector field primitive at a certain point are provided by two separate (procedurally defined) functions. To achieve local influence, the strength of a field is actually a product of the inherent field strength and a spatial term that diminishes with increasing distance from a reference point in the field. In practice, we have used three common functions for the spatial term, namely, uniform, inverse power and a smoothly decreasing function connecting two constants. It is straightforward to define the first two functions. The smoothly decreasing function is defined as

$$SD(r) = \begin{cases} s_0, & r \le r_0; \\ 0.5(s_0+s_1) + 0.5(s_0-s_1)\cos(\frac{r-r_0}{r_1-r_0}\pi), & r_0 < r < r_1; \\ s_1, & r \ge r_1 \end{cases} \tag{1}$$

where a cosine function connects two constant parameters s_0 and s_1. Either s_0 or s_1 can be set to zero. Local influence can be achieved by setting s_1 to zero. Fig. 3 shows a sparse hair model extracted from a superposed vector field. Some of the field primitives we have used in practice are shown in Fig. 4.

The locality of a vector field primitive can also be specified on the scalp. We can associate a list of local regions on the scalp with a field primitive so that only hair strands growing from these regions can be affected by this field primitive. The union of these local regions is called the *domain* of the field primitive.

The guide hairs are represented as polylines so far. The smoothness of the polylines can be improved by Hermite spline interpolation. The tangent at each vertex required by a Hermite spline can be obtained by averaging the directions of the two line segments sharing the same vertex. We can also further edit the guide hairs by transforming Hermite splines into B-splines and interactively adjusting the positions of their control points.

Note that during this guide hair modeling stage, we can interactively make changes to the vector fields and B-splines in real-time since the rendering time for a small number of splines in addition to the synthetic face model is minimal. This is the reason why we choose to model sparse guide hairs first.

2.3 Hair Interpolation

Since we eventually need a dense hair model for rendering, the remaining hair strands in the dense set are interpolated from the guide hairs. Intuitively, one could imagine a simple procedure by averaging the position of the neighboring strands. However, this approach tends to group strands together into unnatural clusters. We adopt a more sophisticated method that produces better interpolation results. It requires an approach for defining a local coordinate system at each potential hair root. A typical scheme for this uses a global UP vector, such as the vertical direction, and the local normal orientation. Our interpolation procedure works as follows:

- Find the nearest root of a guide hair and transform the segments of that guide hair from the world coordinates to its local coordinates. Name this transformation M_1.
- Take these segments in the local coordinates and transform them back to the world coordinate using the local-to-world coordinate transformation defined at the root of the interpolated strand. Name this transformation M_2^{-1}.

$$M_2^{-1}M_1p \tag{2}$$

The procedure is summarized as equation (2), where p is the location of the guide hair in the world coordinate, M_1 and M_2 are the two transformations described previously. More than one nearby guide hairs can be used together to achieve smoother results by merging the multiple transformed guide hairs with some averaging scheme. Local clustering effects can be removed by interpolation from multiple guide hairs. However, when discontinuities caused by vector fields like the one in Fig. 4(a) are present, the guide hairs for interpolation should be chosen carefully so that they fall on the same side of any divider.

In summary, our procedure generates better results by taking into account the round shape of the scalp and considering both rotation and translation between local coordinate systems.

2.4 Generating Random Curliness with an Offset Function

Given the interpolated dense model from the previous section, we still need to add natural curliness of hair strands and even some local randomness in the sweep of each strand to create an overall natural appearance. To achieve this, we first define a specific offset function for hair strands in a canonical coordinate system, then reconstruct curly hairs by modulating the hair sweeps from vector fields with this function.

2.4.1 Offset Function

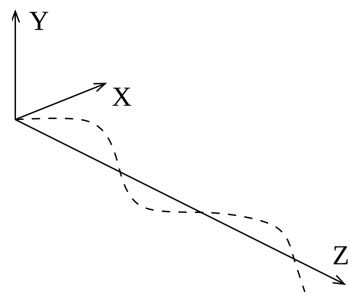


Figure 5: The canonical coordinate system for the hair offset function.

We assume that the underlying hair strand is a straight half line coincident with the positive z-axis of the canonical coordinate system for the offset function, and the origin is at the root of this strand. The offset function is a two-component function that returns the offsets along both $x-$ and $y-$ axes given a z value(Fig. 5).

According to [23], there are three primary types of natural hair waves, namely, a uniplanar wave, a dished wave and a helix. A uniplanar wave looks very much like a planar sinusoidal wave, while a dished wave looks like a sinusoidal wave which has been mapped onto half of a cylindrical surface with the axis of the wave lengthwise along the cylinder. [23] further presented the author's research results on the main reason for the formation of these waves, which is the sequence of periodical contraction and relaxation of small-scale muscles that are in control of the various follicle configurations. Please note that an artificial helix can also be shaped with a hair roller during perming[39] by first wrapping hairs around the roller and then removing the roller along its symmetric axis.

We can see that all three types of hair waves can be easily represented with our offset function although we do not have to be restricted to these types when synthesizing wavy hairs. The uniplanar wave can be represented with a sinusoidal offset for the x-axis and a zero offset for the y-axis. A helix can be represented with a sinusoidal offset for the x-axis and a cosinoidal offset for the y-axis. The dished wave can be represented with a sinusoidal offset for the x-axis, but the wave for the y-axis is a more complicated function which always returns a positive offset value and whose period is at most half of the period of the wave for the x-axis.

Although a variety of parametric or procedurally defined functions can be used, we designed a specific class of offset functions with a fixed number of parameters as follows.

$$\text{Wave}(t) = \text{Mag}(t)\sin(2\pi(Rt + P_0)t + \phi_0) + Bias \tag{3}$$

where t is the function variable; R, P_0, ϕ_0, and $Bias$ are constant parameters; and

$$\text{Mag}(t) \quad = \quad A + Bt\exp(-\alpha t) + C(1 - \exp(-\beta t) + D\exp(\gamma(t - t_0)) \tag{4}$$

where A, B, C, D, α, β, γ, and t_0 are all constant parameters. We can see this is basically a sinusoidal wave with variable period whose initial value is P_0, and with variable magnitude whose initial value is A. If $R > 0$, the period of the wave becomes smaller as we move closer to the tip of the hair, which is a common phenomenon exhibited by real wavy hairs. The last three terms in Eq.(4) allow us to vary offset magnitude at different parts of a hair strand. The second term reaches a maximum at $t = 1/\alpha$; the third term becomes close to C with sufficiently large t; and the last term increases exponentially especially when $t > t_0$ which can be used to model curly features at the end of a strand. With two sets of different parameters, this function can model offsets along both x- and y-axes. All the examples shown in this paper use this parameterization of the offset function.

2.4.2 Editing Hair Sweeps with the Offset Function

To modulate a hair sweep from vector fields with the above offset function, we need to consider the hair sweep as the parametric axis for the variable t in the offset function. For a certain point p on the sweep, its corresponding t value is the accumulated arc length between p and the root of the sweep. Thus, we can obtain a pair of offset values for each point on the sweep.

We still need to define a local coordinate system at each point on the sweep to impose the offsets. The z-axis of the local system at a point p is always the tangential direction of the sweep at p. We use the vector between the point p and the centroid of the scalp as an UP vector for the local system. Then both x- and y- axes can be derived from the UP vector and the z-axis. So p is originally at the origin of this local system. Its new location is always on the local xy-plane and is determined by the pair of offsets returned from the offset function. [30] presents a similar idea to edit details of a 2D curve.

The same editing operation can still be applied even when the hair sweep has dynamics which makes it deform from frame to frame. However, every local coordinate system, once initialized as above, should follow the same transforms that its corresponding point on the sweep undergoes.

2.4.3 Random Curliness

To achieve natural appearance, obviously we should not edit all hair sweeps with the same set of offset parameters. On the other hand, random hair curliness is far from white noise. The author of [23] observed smoothly spatially varying initial phase, ϕ_0 in Eq.(3), in hair waves. During perming, usually a local cluster of hair is wrapped around a roller, which also suggests that hair waves from that cluster share similar shapes. In practice, we set up a multi-scale grid over the scalp in a polar coordinate system, and generate smoothly varying random parameters for the offset function using Perlin's noise[26] given the mean and standard deviation of each parameter for each scale.

2.5 Hair Clustering

Because of cosmetics and static charges, nearby hairs tend to form clusters. This effect is partially dealt with in Section 2.4.3 when multiscale noise is synthesized. Hairs growing from the same cell in the highest resolution grid have the same parameters for the offset function, therefore tend to be close to each other. Here we introduce an additional offset to further reinforce this effect when the previous treatment is not sufficient. One hair is chosen to be the representative of the hairs growing from the same cell in the grid. Each vertex on other hairs from the same cell is forced to move by a certain distance towards its corresponding vertex on the representative hair. Hair clusters become obvious after this step.

Fig. 6(a) shows the interpolation result. Fig. 6(b)-(c) shows the difference the offset function can make on hair appearance. Fig. 6(d) demonstrates the effectiveness of the hair clustering step.

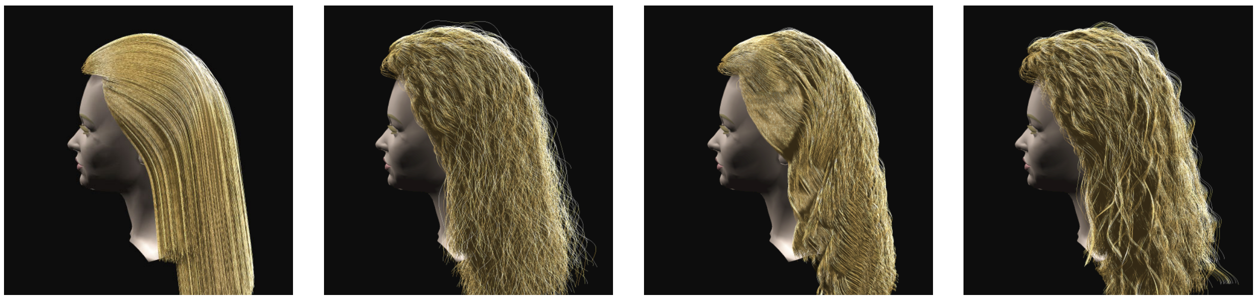


Figure 6: (a) A static dense hair model interpolated from the sparse model shown in Fig. 3(b); (b) an improved version after modulating the intial model in (a) with random offset functions at a fine scale; (c) another version by modulating the model in (a) with random offset functions at a coarse scale. (b) and (c) look quite different because of different offset functions; (d) the appearance of the model in (b) is improved by further clustering nearby hairs;

3 Hair Animation

The modeling approach in the previous section only generates static hair models. To elaborate hair animation techniques, let us begin with single hair dynamics. Then, a sparse model for hair-hair interactions will be introduced.

3.1 Single Hair Strand Dynamics

There are few techniques developed on modeling single hair strand dynamics [29, 1, 6, 12]. Some of the previous work [29, 6] models a single strand as particles connected with rigid springs. Each particle has 3 degrees of freedom, namely one translation and two angular rotations. This method is simple and easy to implement. However, individual hair strand has very large tensile strength, and hardly stretches by its own weight and body forces. This property leads to stiff equations which tend to cause numerical instability unless very small time steps are used. We model each hair strand as a serial rigid multibody chain. There is a rotational joint between two adjacent segments, and translational motion is prohibited. A single chain can be considered as a simple articulated body with joint constraints. Dynamic formulations of articulated bodies are addressed in robotics [7, 25] as well as graphics literature [37]. Both constrained dynamics with Lagrange Multipliers [2] and generalized(or reduced) coordinate formulation [7] can be used equally efficiently. The dynamics of a serial multibody chain and its generalized coordinate formulation have recently been applied to single hair simulation in [12]. Since our main contributions in hair animation concern hair-hair interactions, we describe the formulation of the serial multibody chain and our adaptations briefly in this section.

3.1.1 Kinematic Equations

In our model, we assume that the twisting of a hair strand along its axis is prohibited. This reduces each rotational joint in a strand to have two degrees of freedom. A rotational joint can be decomposed into two cascading one-dimensional revolute joints each of which has a fixed rotation axis [25]. The rotation angles at the 1D revolute joints represent the set of generalized coordinates in a multibody chain system. If a 1D revolute joint has a rotation axis ω along with a point q on the axis, the matrix transformation corresponding to a rotation around ω by an angle θ can be given by the exponential map $e^{\widehat{\xi}\theta}$ [25] where

$$\widehat{\xi} = \begin{bmatrix} \widehat{\omega} & v \\ 0 & 0 \end{bmatrix}, \widehat{\omega} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix},$$

and $v = -\omega \times q$. Suppose a hair segment has n preceding 1D revolute joints in the chain and a local frame is defined at the segment. Assume the local-to-world transformation for this frame when all preceding joint angles are zero is $g_{st}(0)$. The updated local-to-world transformation after a series of rotations at the n joints becomes

$$g_{st}(\Theta) = e^{\widehat{\xi}_1\theta_1} e^{\widehat{\xi}_2\theta_2} \cdots e^{\widehat{\xi}_n\theta_n} g_{st}(0) \tag{5}$$

Thus, given an arbitrary series of joint angles, the position of every vertex in the chain can be obtained using this product of exponentials of its preceding joints. The exponential map actually is just another way of formulating a 4×4 homogeneous matrix. It can be calculated in constant time [25]. Therefore, the whole chain can be evaluated in linear time.

3.1.2 Dynamics of Hair Strand

Given the mapping in Eq. 5 which is from the set of generalized coordinates (joint angles) to real 3D world coordinates, hair strand simulation can be solved by integrating joint angular velocities and accelerations. Forward dynamics of a single strand in terms of joint angular velocities and accelerations can be solved using Featherstone's algorithm [7] or Lagrange's equations for generalized coordinates [25]. The former method is more efficient with a linear time complexity. Detailed formulations of these methods can be found in [7, 25]. In practice, we adopt an implementation of Featherstone's algorithm by Kuffner and Mirtich [24] for the stable simulation of cascading 1D revolute joints.

Both external and internal forces are indispensable for single hair dynamics. In this paper, hair-hair interactions are formulated as external forces in addition to gravity. The actual form of these external forces will be discussed in Section 3.2. At each joint of the hair chain, there is also an internal actuator force to account for the bending and torsional rigidity of the strand. We model the actuator force as a hinge with a damping term as in [29]. Since the hairs from our modeling stage may have deformations even when there are no external forces, we define a nonzero resting position for each hinge. Any deviation from the resting position results in a nonzero actuator force trying to reduce the amount of deviation. This setup helps a strand to recover its original shape after subsequent movement.

3.1.3 Strand-Body Collision

In order to simulate inelastic collision between the hair and human body, there is no repelling forces introduced by the human body. Once a hair vertex becomes sufficiently close to the scalp or torso, it is simply stopped by setting its own velocity to be the same as the velocity of the human body while all the following vertices in the multibody chain are still allowed to move freely.

Any acceleration towards the human body is also prohibited at the stopped vertices which, however, are allowed to move away from or slide over the human body. Frictional forces are added as well to those vertices touching the human body. Collision detection is handled explicitly by checking penetration of hair strand particles with the triangle mesh of the body parts.

This scheme cannot guarantee that the hair vertices do not penetrate other colliding surfaces in the middle of a time step. If penetration does occur, we need to move the part of the penetrating strand outside the surface in the same time step so that no penetration can be actually observed. It is desirable that the tip of the hair, if outside the surface, remains unchanged during this adjustment in order to introduce minimal visual artifacts. To achieve this goal, inverse kinematics [25] can be applied to adjust the positions of the intermediate vertices between the tip and the adjusted locations of the penetrating vertices. In our implementation we opt for a simpler method using iterative local displacements. Starting from the root, we move the first penetrated vertex p_1 to its nearest valid location p'_1, and then propagate this displacement by moving the subsequent vertices. More specifically, assume the following vertex of p_1 is p_2, we compute the vector $v' = ||p_2 - p_1|| \frac{v}{|v|}$, where $v = p_2 - p'_1$. The new location for p_2 after the adjustment is $p'_2 = p'_1 + v'$. We repeat this for all the vertices following p_1 until reaching the tip.

3.2 A Sparse Model for Hair-Hair Interaction

We devise a novel scheme to simulate only a sparse set of guide hairs for complex hair-hair interactions. A guide hair essentially represents a local hair wisp. Therefore, it should have the mass of a whole wisp instead of a single hair. We first introduce an elastic model to preserve the relative positions of the hair strands. The static links model the interaction of the hair due to interweaving, static charges and hairstyling. Second, the hair-hair collision and friction is simulated using the guide hairs and a collection of auxiliary triangle strips. Last, we provide an adaptive hair generation technique to complement our sparse hair model. The proposed method models the hair dynamics efficiently with good visual realism.

3.2.1 Static Links

It is evident that the hair strands tend to bond together with other strands in their vicinity because of cosmetics, static charges and the interweaving of curly hairs. As a result, the movement of each strand is on most part depended on the motion of other strands. These interactions can have relatively long range effects besides clustering in a small neighborhood. While hair local clustering is modeled by default using our sparse model, longer range interaction is not. Furthermore, slight head movements or external forces exerted on the hair do not change a hairstyle radically. This is partly because each hair strand has its internal joint forces and resting configuration. However, an individual hair's recovery capability is quite limited especially for long hairs. The bonding effect among hairs plays an important role. Dramatic movements can break the bonds created by hairstyling, static charges or interweaving.

To effectively model the bonding effect, we may view the hair as one elastically deformable volume. Traditional models for deformable bodies include 3D mass-spring lattice, finite difference, and finite element method [35, 41]. These models approximate the deviation of a continuum body from its resting shape in terms of displacements at a finite number of points called nodal points. Although the vertices of hair strands may serve as the nodal points inside this hair volume, directly applying traditional models is not appropriate for the following reasons. We are only interested in an elastic model for hair's lateral motion. Under strong external forces, the continuum hair volume may break into pieces which may have global transformations among them. Therefore, using one body coordinate system for the whole hair volume is inadequate.

We propose to build breakable connections, called static links, among hair strands to simulate their elastic lateral motion and enable hairstyle recovery. These connections are selected initially to represent bonds specific to a hairstyle since different hairstyles have different hair adjacency configuration. The static links enforce these adjacency constraints by exerting external forces onto the hair strands. Intuitively, one can use tensile, bending and torsional springs as bonds to preserve the relative positions of the hair strands. In practice, we opt for a simpler and more efficient method using local coordinates.

We introduce a local coordinate system to each segment of the hair strands. For each segment, we find a number of closest points on nearby strands as its reference points. To improve the performance, an octree can be used to store the hair segments for faster searching. We transform these points, which are in the world coordinates, to the segment's local coordinates (Fig. 7a). The initial local coordinates of these reference points are stored as part of the initialization process. Once strands have relative motion, the local coordinates of the reference points change and external forces are exerted onto these strands to recover their original relative positions (Fig. 7b). We model these external forces as spring forces with zero resting length. One advantage of using the local coordinates is that it eliminates the need for bending and torsional springs.

Let us consider a single hair segment h with m reference points. The initial local coordinates of these reference points are represented as $p^o_{h,i}, i = 1, ..., m$, while their new local coordinates are represented as $p^n_{h,i}, i = 1, ..., m$. The accumulated force this segment receives due to static links can be formulated as

$$\mathbf{f}_h = \sum_i \left[k^s_{h,i} |\mathbf{l}_i| - k^d \frac{\mathbf{v}_i \cdot \mathbf{l}_i}{|\mathbf{l}_i|} \right] \frac{\mathbf{l}_i}{|\mathbf{l}_i|} \qquad (6)$$

We compute the spring force using the Hook's law in (6), where $k^s_{h,i}$ is the spring constant for the i-th reference point of segment h, and k^d is the universal damping constant. Since the resting length in our case is zero, $|\mathbf{l}_i = p^n_{h,i} - p^o_{h,i}|$ is multiplied by $k^s_{h,i}$ directly. $\mathbf{v}_i$ is the time derivative of $\mathbf{l}_i$.

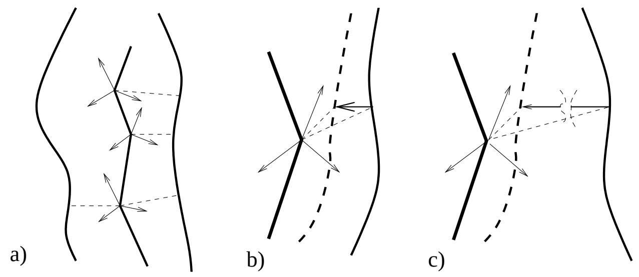


Figure 7: Each hair segment has its own local coordinate system where the forces from all static links (dashed lines) are calculated.

Similar to the bonds of stylized hair, static links can be broken upon excessive forces. We set a threshold for each static link. If the length change of a static link is greater than the threshold, the static link breaks (Fig. 7c). Once a link is broken, the damage is permanent; the link will remain broken until the end of the simulation. To be more precise, we model the spring constant $k^s_{h,i}$ as shown in Fig. 8. As $|\mathbf{l}_i|$ increases beyond δ_1, the spring constant begins to decrease gradually and eventually becomes zero at δ_2 as the spring snaps. The spring constant will not recover even when $|\mathbf{l}_i|$ shrinks below δ_1 again. This nonreversible spring model would make the motion of the hair look less like a collection of rigid springs.

When external forces recede, the original hairstyle may not be completely recovered if some of the static links have been broken. New static links may form for the new hairstyle with updated neighborhood structures.

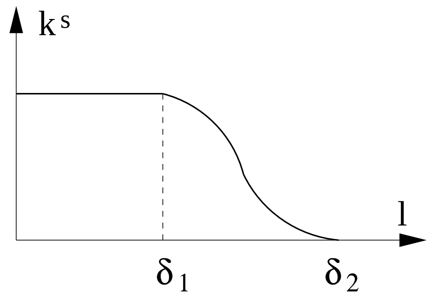


Figure 8: Spring constant $k^s_{h,i}$ vs displacement graph.

3.2.2 Dynamic Interactions

Elastic deformation only introduces one type of hair-hair interactions. Hairs also interact with each other in the form of collision. To effectively simulate hair-hair collision and friction using a sparse hair model, we need to have a dynamic model that imagines the space in between the set of sparse hairs as being filled with dense hairs. Collision detection among the guide hairs only is much less accurate. Let us consider a pair of nearby guide hairs. The space between them may be filled with some hairs in a dense model so another strand cannot pass through there without receiving any resistance. To model this effect, we can either consider the guide hairs as two generalized cylinders with sufficiently large radii to fill up the gap between them, or build an auxiliary triangle strip as a layer of dense hair between them by connecting corresponding vertices. The triangular mesh can automatically resize as the guide hairs move, but it is trickier to resize the generalized cylinders. Therefore, we propose to construct auxiliary triangle strips between pairs of guide hairs to approximate a dense hair distribution. If we consider the set of dense hairs collectively as a volume, a triangle strip represents a narrow cross section of the volume. A number of such cross sections can reasonably approximate the density distribution of the original hair volume.

Since the distance between a pair of vertices from two hairs may change all the time during simulation, we decide to use the distance among hair roots. A triangle strip is allowed as long as two guide hairs have nearby hair roots. Each triangle only connects vertices from two guide hairs, therefore is almost parallel to them. Note that the triangle strips may intersect with each other. This does not complicate things because each triangle is treated as an independent patch of hair during collision detection. The triangles are only used for helping collision detection, not considered as part of the real hair geometry during final rendering. They do not have any other dynamic elements to influence hair movement. However, some triangles may have nearby static links which can help them resist deformation. The triangle edges are not directly constructed as static links because static links only connect nearby hair segments while not all the segments connected by triangles are close to each other.

As in standard surface collision detection, two different kinds of collision are considered, namely, the collision between two hair segments and the collision between a hair vertex and a triangular face. Since each guide hair represents a local hair cluster with a certain thickness, a collision is detected as long as the distance between two hair elements falls below a nonzero threshold. Once a collision is detected, a strongly damped spring force is dynamically generated to push the pair of elements away from each other [3]. Meanwhile, a frictional force is also generated to resist tangential motion. A triangle redistributes the forces it receives to its vertices as their additional external forces. Both the spring and frictional forces disappear when the distance between the two colliding elements becomes larger than the threshold. The spring force in effect keeps other hairs from penetrating a layer corresponding to a triangle strip. An octree is used for fast collision detection. All the moving hair segments and triangles are dynamically deposited into the octree at each time step. An octree node has a list of segments and triangles it intersects with.

Hair also exhibits strong anisotropic dynamical properties. Depending on the orientation of the penetrating hair vertex and the triangular face, the repelling spring force might vary. For example, hair segments of similar orientation with the triangle strip should experience weaker forces. We scale the repelling spring force according to the following formula.

$$\mathbf{f}_r = \lambda(1 - |\mathbf{a} \cdot \mathbf{b}|)\mathbf{f}_s \tag{7}$$

The original spring force $\mathbf{f}_s$ is scaled in Eq. (7), where $\mathbf{a}$ is the normalized tangential vector of the hair at the penetrating vertex, $\mathbf{b}$ is the interpolated hair orientation on the triangular face from its guide hair segments, and λ is a scale factor. When $\mathbf{a}$ and $\mathbf{b}$ are perfectly aligned, the scaled force $\mathbf{f}_s$ becomes zero. On the contrary, when they are perpendicular, the spring force is maximized. The collision force between two hair segments can be defined likewise.

The hair density on each hair strip is also modeled as a continuum. It can be dynamically adjusted during a simulation. If there is insufficient hair on a strip, the strip can be broken. This would allow other hair strands to go through broken pieces of a hair strip more easily. This is reasonable because sometimes there is no hair between two hair clusters while at the other times, there may be a dense hair distribution. In our current implementation, the length of the triangle edges serves as the indicator for when the hair density on a strip should be adjusted. If a triangle becomes too elongated, it is labeled as broken. If a triangle is not broken, the magnitude of the collision force in Eq. (7) is made adaptive by adjusting the scale factor λ according to the local width of the triangle strip to account for the change of hair density on the triangle. Unlike the static links, this process is reversible. Once the two guide hairs of a strip move closer to each other again, indicating the hair density between them is increasing, the generated collision force should also be increased, and the triangle strip should be recovered if it has been broken. If every triangle strip in our method is modeled as broken from the beginning, our collision model becomes similar to the wisp model in [28] since every guide hair in our method actually represents a wisp.

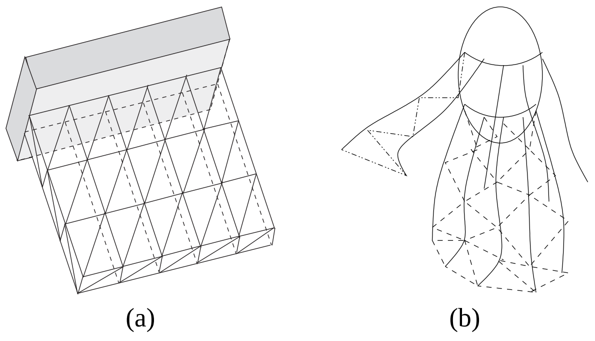


Figure 9: (a) For a brush, triangle strips can be inserted between horizontally and vertically adjacent guide hairs. (b) For a human scalp, triangle strips are inserted only between horizontally adjacent guide hairs

It may not be necessary to build triangle strips among all pairs of nearby strands. For a simple brush in Fig. 9a, we can only insert triangle strips between horizontally and vertically adjacent guide hairs. For human hair, we sometimes find it practically good enough to build triangle strips between guide hairs with horizontally adjacent hair roots (Fig. 9b). This is because hairs drape down due to gravity, and the thickness of the hair volume is usually much smaller than the dimensions of the exterior surface of the hair volume. In such a situation, using triangles to fill the horizontal gaps among guide hairs becomes more important.

3.2.3 Adaptive Guide Hair Generation

Initially, we select the guide hairs uniformly on the scalp. However, it is not always ideal to pick the guide hairs uniformly. During a run of the simulation, some part of the hair may be more active than the other parts. For example, when the wind is blowing on one side of the hair, the other side of the hair appears to be less active. As a result, some computation is wasted for not so active regions. For not so active regions, fewer guide hairs combined with interpolation is sufficient. However, for more active regions, it is desirable to use more guide hairs and less interpolation for better results. We design an adaptive hair generation method to complement our sparse hair model.

We generate additional guide strands adaptively during the simulation to cover the over interpolated regions. The distribution and the initial number of guide strands are determined before the simulation. However, as the simulation proceeds, more hair strands can be added. The hair model may become more and more computationally intensive if hairs can only be inserted. We notice that the inserted hairs may become inactive again later in the same simulation. Therefore, we also allow them to be deleted if necessary. To keep our hair strands relatively sparse, we may also set a limit on how many adaptive hair strands can exist at the same time. Picking the right place to generate adaptive guide hair is important.

We use a simple technique to detect where to add and remove adaptive guide hairs. For each pair of guide strands, we compute the distance between all pairs of corresponding vertices of the strands. If any pair of vertices become farther away than a threshold, it indicates that the hair in between these two guide strands is relying too much on the interpolation. We then add an adaptive guide hair half-way between these two strands. At the same time, we exam the adaptive guide hairs from the last step of the simulation. If some of the guide strands are no longer needed (when the two neighboring strands are sufficiently close), we remove those strands and save them for future hair generation. When an adaptive hair is generated, its initial vertex positions and velocities are obtained by interpolating from those of the two initiating guide hairs. If there was a triangle strip between these two initiating hairs, it should be updated to two strips with the new hair in the middle. The new adaptive hair then follows its own dynamics from the next time step, colliding with nearby strands and triangle strips. To avoid discontinuous motion on the rest of the hairs, a new adaptive hair does not spawn static links with other strands.

3.2.4 Adaptations for Curly Hair

In our approach (Fig. 1(b)), curliness is modeled in the last step before rendering. Guide hairs of a curly hair model are actually quite straight except for large-scale deformations. They represent the spines of curly hairs. However, the mutual interactions among the guide hairs should reflect the features in the final dense hair model since the motion part of the guide hairs is still going to be used for rendering even after hair curliness has been changed. In the case of a dense curly hair model, there should be stronger static links because curly hairs have more interweaving effects, and weaker hair-hair collisions because curliness generates a soft cushion effect among hairs. In practice, we follow these intuitions to set up the strength of mutual interactions.

3.3 Dynamic Dense Hair Modeling

Hair animations are generated in two passes in our system. In the first pass, the set of guide hairs are simulated from frame to frame with all the external forces and hair-hair interactions described in Section 3.2. During the simulation of each individual guide hair as described in Section 3.1, forces due to hair-hair interactions are treated as additional external forces. Hair-object collisions are also detected during this simulation. By the end of the first pass, we have obtained the positions and velocities of each guide hair at each frame.

In the second pass, we generate a dense hair model for each frame according to the guide hair positions. Curliness modeled using our offset function in Section 2.4 can also be added if necessary. To maintain the consistency from frame to frame, each hair in the dense model is assigned a fixed offset function throughout the whole sequence. Therefore, at each frame, we merely modulate the same offset function onto the underlying deforming spine to produce the desired curly strand. However, the offset function may still vary from strand to strand.

Since small objects may miss all the guide hairs, but still hit some of the strands in the dense model, we decide to run hair-object collision detection for each (curly) hair in the dense model. Although this involves a certain amount of computation, the computing power available nowadays on a single processor workstation has already become sufficient to perform this task in a very short amount of time. If a hair penetrates an object, a scheme similar to the one described in Section 3.1.3 is used to adjust the hair.

4 Hair Rendering

Although the primary focus of this paper is on hair modeling and animation, we will discuss briefly our approach to rendering realistic hair. The kind of physical interaction considered here includes self-shadowing and light scattering. Hair strands are not completely opaque. Therefore, the interaction between light and hair leads to both reflection and transmission. When a dense set of hairs is present, light gets bounced off or transmitted through strands multiple times to create the final exquisite appearance. Basically, we can view a dense hair as a volume density function with distinct density and structures everywhere. The hair density is related to the local light attenuation coefficient while the structures, including the local hair orientation, are related to the phase function during scattering. In this section, we discuss how to efficiently render animated sequences of hair with high visual quality by considering the above factors.

While secondary scattering can improve the rendering quality, primary scattering and self-shadowing are considered much more important. Since the rendering performance is our serious concern when generating hair animations, we decide to simulate the latter two effects only. This is equivalent to solving the following volume rendering equation [15],

$$L(x,\vec{\omega}) = \int_{x_0}^{x} \tau(x',x)\sigma(x') \sum_{l} f(x',\vec{\omega}_l,\vec{\omega}) I_l(x') dx' \tag{8}$$

where $L(x,\vec{\omega})$ represents the final radiance at x along direction $\vec{\omega}$, $f(x',\vec{\omega}_l,\vec{\omega})$ is the normalized phase function for scattering, $I_l(x')$ is the attenuated light intensity from the l-th light source, and $\tau(x',x) = \exp(-\int_{x'}^{x}(\alpha(\xi)+\sigma(\xi))d\xi$ where $\alpha(x)$ is the absorption coefficient and $\sigma(x)$ is the scattering coefficient. As in volume rendering, the final color of a pixel can be approximated as the alpha-blending of the colors at a few sample points along the eye ray going through that pixel. To perform alpha-blending correctly, the sample points need to be depth-sorted. In terms of hair, the sample points can be the set of intersections between the eye ray and the hair segments. Note that the input to the rendering stage is a large number of hair segments resulted from the discretization of the spline interpolated dense hairs. In order to obtain the set of intersections at each pixel efficiently, scan conversion is applied to the segments and a segment is added into the depth-sorted list of intersections at a pixel once it passes that pixel. Antialiasing by supersampling each pixel can help produce smoother results.

To finish rendering, we still need a color for alpha-blending at each of the intersections. It should be the reflected color at the intersection. The reflectance model we use is from [10]. It is a modified version of the hair shading model in [14] by considering partial translucency of hair strands. Since other hairs between the light source and the considered hair segment can block part of the incident light, the amount of attenuation is calculated using the opacity shadow maps [17] which can be obtained more efficiently than the deep shadow maps [22]. Basically, the algorithm in [17] selects a discrete set of planar (opacity) maps perpendicular to the lighting direction. These maps are distributed uniformly across the volume being rendered. Each map contains an approximate transmittance function of the partial volume in front of the map. Thus, the approximate transmittance of the volume at any point can be obtained by interpolating the transmittance at corresponding points on the two nearest opacity maps. In our implementation, exponential interpolation has been used since the attenuation of light through a volume is exponential. The exponential interpolation can be written as

$$\exp(-\alpha_1 \frac{d_2}{d_1+d_2} - \alpha_2 \frac{d_1}{d_1+d_2})$$

where $\exp(-\alpha_1)$ and $\exp(-\alpha_2)$ are the attenuation at the two nearest maps, and d_1, d_2 are the distance from the point to the two maps, respectively.

When the hair is rendered together with other solid objects, such as the head and cloth, which we assume to be completely opaque, the color of the solid objects needs to be blended together with the hair's during volume rendering. The solid objects also have their separate shadow buffer for each light source. Anything in the shadow of the solids receives no light while those solids in the shadow of the hair may still receive attenuated light.

5 Results

5.1 Hair Modeling

We have applied our modeling technique to a variety of hairstyles with different degrees of curliness, and obtained satisfactory results. Some of the hairstyles are shown in Fig. 10. They look natural and realistic. Vector field primitives can either be interactively adjusted by the user or be randomly generated. Typically, we need around ten vector field primitives to interactively model a hairstyle. The time for user interaction is usually between one and two hours. On the other hand, Fig. 10(d) uses 273 field primitives randomly generated and distributed over the scalp.

For each local cluster of hairs, the parameters in Eq. (3) are randomly generated. The mean and standard deviation of each parameter at each scale are fixed for all hairs, and are specified by the user. For the results shown here, we use two levels of scales to generate the random parameters. The final hair waves in Fig. 10(a) and in Fig. 6(d) actually have the same average magnitude, 5mm. However, hair waves in Fig. 6(d) look more random because they have a much shorter period and a much larger standard deviation for the initial phases of the waves. The hairs in Fig. 10(c) have an average magnitude of 1mm to give a smoother look.

5.2 Hair Animation

We have also successfully tested our hair dynamic model in a few animations. In our experiments, we used around 200 initial guide hairs during the animation of the sparse model with 15 segments for each strand. During each time step, a strand is interpolated with a Hermite spline and discretized into around 50 smaller segments. Based on this set of resampled sparse hairs, a dense hair model with 50,000 strands is generated on the fly at each frame for the final rendering. The guide hair animation stage takes about one second per frame on a Pentium III 800MHz processor. Hair interpolation, hair-object collision detection and antialiased rendering takes another 20 seconds per frame on a Pentium 4 2GHz processor. Fig. 11(a)-(b) show a sparse hair model with static links and the dense interpolated model. Static links can prevent excessive changes to a hairstyle during motion (Fig. 11(c)-(d)) as well as provide hairstyle recovery after motion. Fig. 12 shows two synthetic renderings of animated hair.

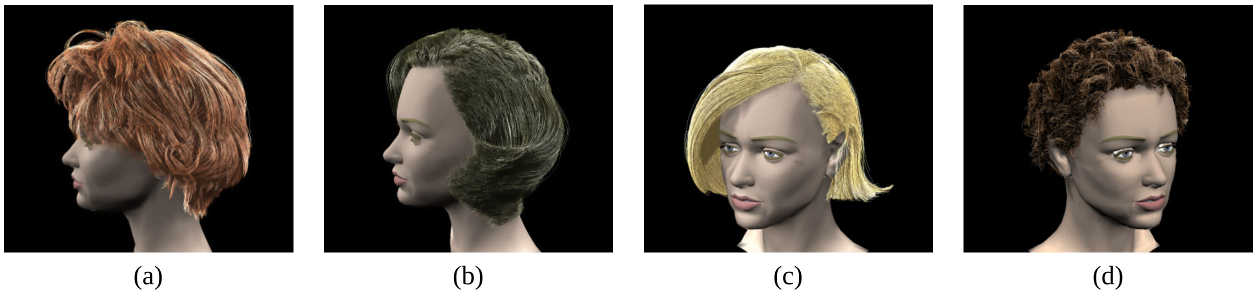


Figure 10: Four hairstyles produced from our techniques. These hairstyles have different length distributions and different levels of curliness.

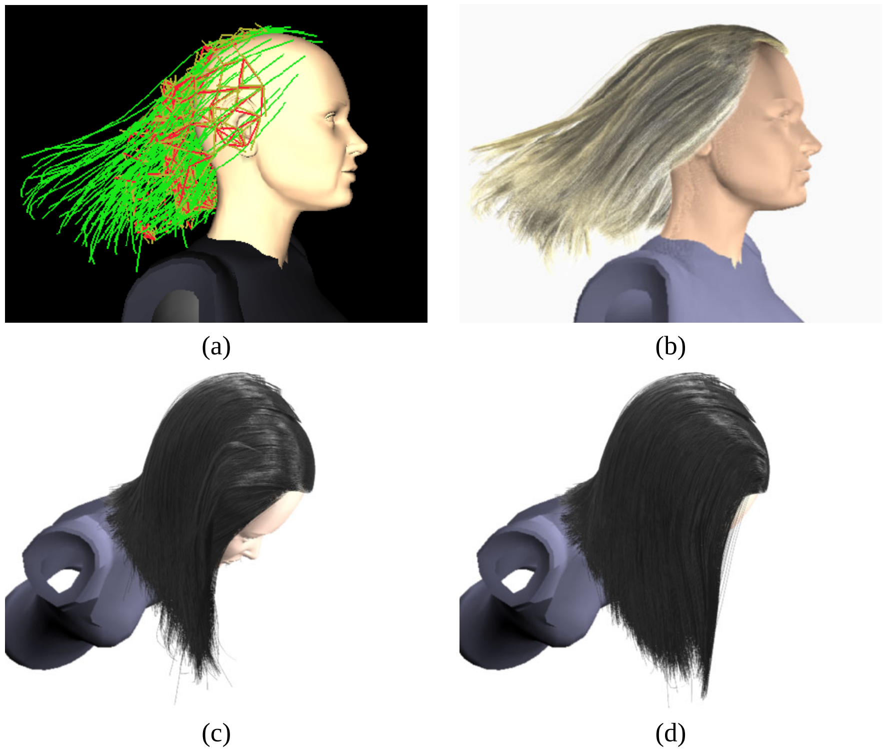


Figure 11: (a) A sparse hair model displayed with static links, (b) a rendered image of the interpolated dense model, (c)-(d) a comparison between hairs with and without static links. The image in (c) has static links, while the one in (d) does not.

5.2.1 Comparison with Ground Truth

A synthetic head shaking sequence is compared with a real reference sequence in Fig. 13. The hair strands in the real sequence obviously have mutual connections since they move together. We use relatively strong static links to simulate this effect. The head motion in the synthetic sequence was manually produced to approximate the real motion. Nonetheless, the synthetic hair motion reasonably matches the real one.

5.2.2 Dynamic Collision

To demonstrate the effectiveness of our hair collision strategy, we built a simple braided hair model and let it unfold under gravity. There are basically two sets of guide hairs in the model, and static links and triangle strips are only built among hairs from the same set. Therefore, the two sets can move away from each other. A comparison is given between images from two synthetic sequences in Fig. 14, one with collision detection and the other without. In the simulation without collision detection, hairs go through each other. But in the sequence with collision detection, hairs unfold correctly in a spiral motion.

5.2.3 Hair-Air Interaction

Hair-air interaction is traditionally modeled as air drag which only considers the force exerted on the hair from the air. However, the velocity field of the air is also influenced by the hair. The method in [12] can be adapted to our model for hair-air interaction. That is, the air is simulated as a fluid and it generates a velocity field. Each hair vertex receives an additional external force from the air. This force can be modeled as a damping force using the difference between the velocity of the air at the vertex and the velocity of the hair vertex itself. The force exerted from the hair back to the air can be modeled similarly. If the air is simulated using a voxel grid [8], the velocity of the hair at each grid point can be approximated using the velocities of the nearby hair vertices and auxiliary triangles.

Fig. 15(top) shows images from a hair animation with a wind. The wind velocity field is driven by an artificial force field with a changing magnitude and direction. The head and torso are considered as hard boundaries in the wind field while the wind can go through hairs with a certain amount of attenuation.

5.2.4 Brush Simulation

In addition to human hair interactions, we simulate the dynamics of brushes. Fig. 16 shows images from a sequence with a sphere colliding with a synthetic brush. The mutual interactions are weak when only a small number of hairs drape down behind the sphere. However, when more and more hairs drape down, they stabilize much faster because of the collisions.

5.3 Hair Rendering

An artistic flavor can also be added to the images by rendering the hair with increased translucency and specularity. Fig. 15(bottom) shows some re-rendered images from one of the wind blowing sequences with higher specularity. Fig. 17 shows a comparison between normally rendered hair and hair with high translucency.

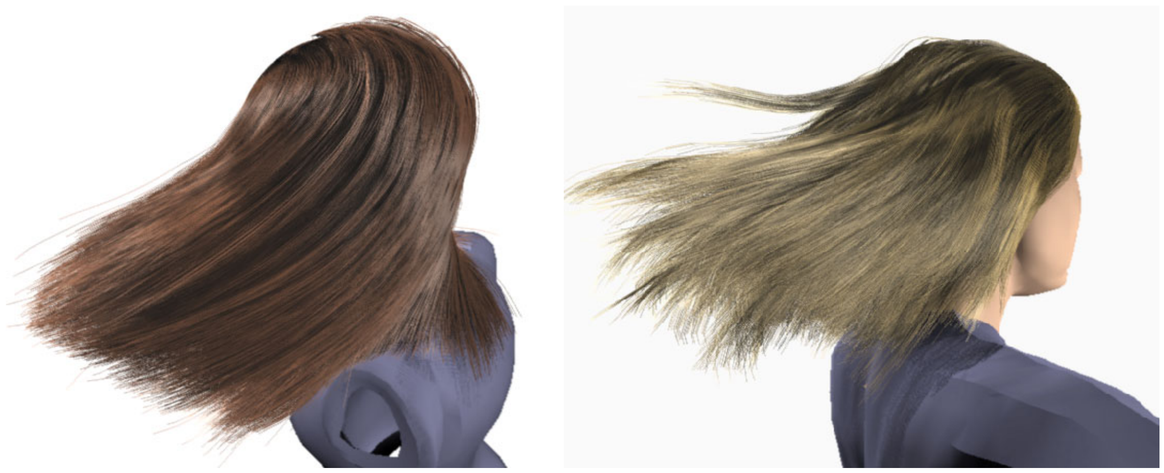


Figure 12: Two synthetic renderings of animated hair.

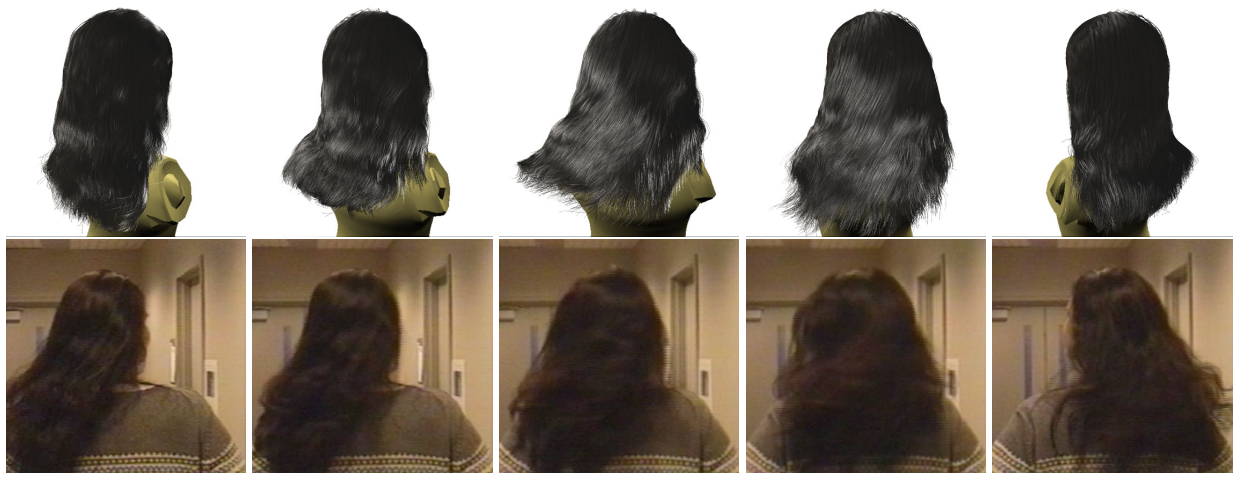


Figure 13: A comparison between a simulated hair animation and a real video. The hair motion is caused by the underlying head motion. Top row: images from the simulated hair motion sequence. Bottom row: images from the real video. The simulated hair motion approximately matches the real hair motion in the video.

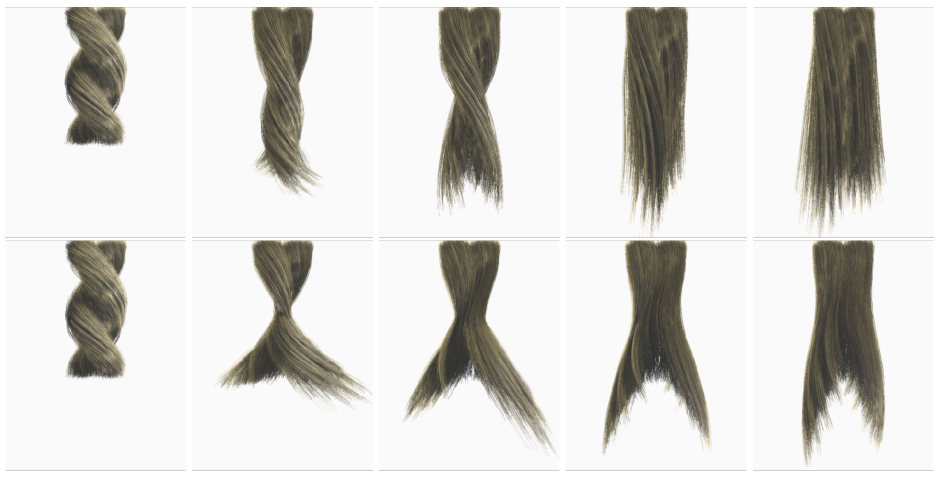


Figure 14: A comparison between two hair animations with and without collision detection. Top row: braided hair unfolds correctly in a spiral motion because of the collision detection. Bottom row: hairs penetrate each other when there is no collision detection.

6 Discussions and Conclusions

In this paper, we presented a unified framework for hair modeling and animation. Both modeling and animation start from a sparse set of guide hairs. The proposed modeling approach can generate realistic wavy hairstyles by editing a basic sheared hair model with a generic offset function for curliness. we also presented an integrated model for hair animation. Specifically, the dynamic model can perform the following functions: the static links and the joint actuator forces enable hairstyle recovery; hair-hair collision becomes more accurate by inserting auxiliary triangle strips and performing collision detection among strands as well as between strands and triangle strips; stable simulation of individual strands is provided by the formulation for multibody open chains. Although our model is not originally designed for hairs without obvious clustering effects, with our multiple hair interpolation scheme, visual results for this kind of hair turned out quite reasonable.

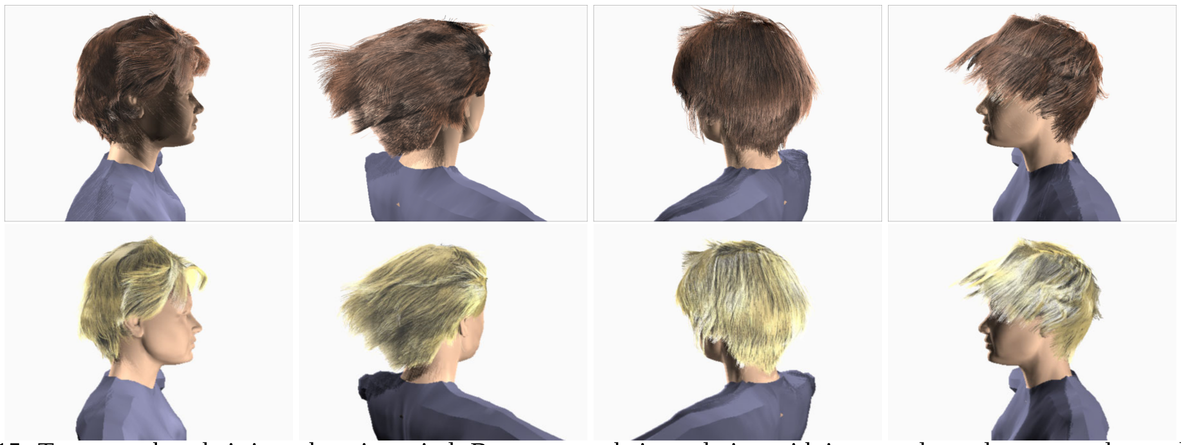


Figure 15: Top row: short hair in a changing wind. Bottom row: hair rendering with increased translucency and specularity to convey an artistic flavor.

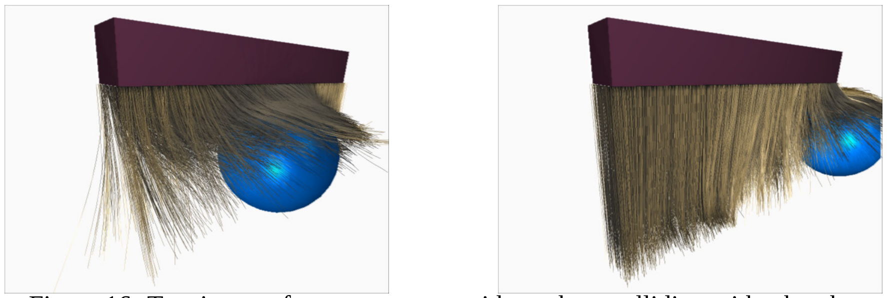


Figure 16: Two images from a sequence with a sphere colliding with a brush.

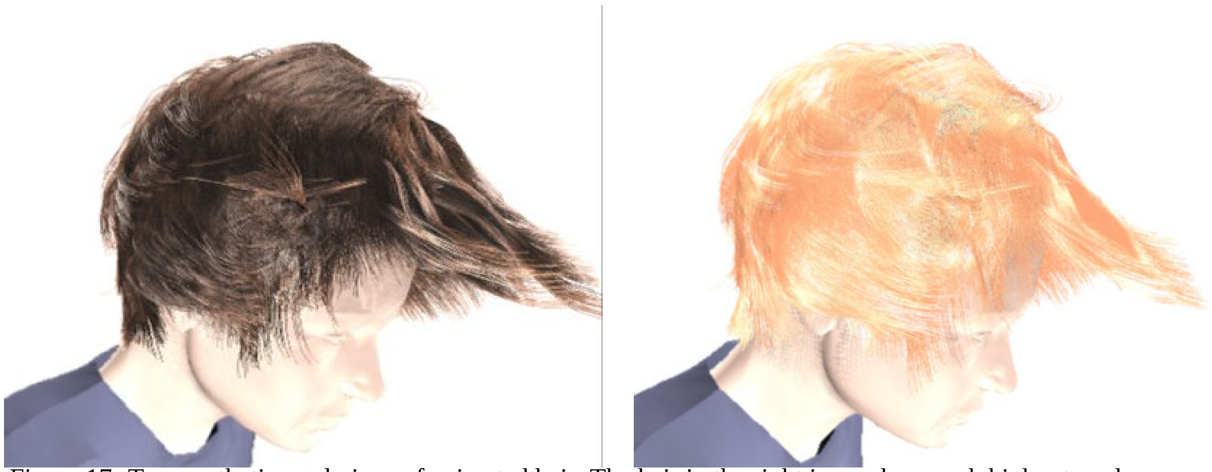


Figure 17: Two synthetic renderings of animated hair. The hair in the right image has much higher translucency.

Acknowledgments

This work was supported by National Science Foundation CAREER Award CCR-0132970 and start-up funds from the University of Illinois at Urbana Champaign.

References

[1] K. Anjyo, Y. Usami, and T. Kurihara. A simple method for extracting the natural beauty of hair. In *Proc. of SIGGRAPH'92*, pages 111–120, 1992.

[2] D. Baraff. Linear-time dynamics using larange multipliers. In *Proc. of SIGGRAPH'96*, pages 137–146, 1996.

[3] D. Baraff and A. Witkin. Large steps in cloth simulation. In *Proc. of SIGGRAPH'98*, pages 43–54, 1998.

[4] J.T. Chang, J. Jin, and Y. Yu. A practical model for hair mutual interactions. In *Proceedings of ACM SIGGRAPH Symposium on Computer Animation*, pages 73–80, 2002.

[5] L.-H. Chen, S. Saeyor, H. Dohi, and M. Ishizuka. A system of 3d hair sytle synthesis based on the wisp model. *The Visual Computer*, 15(4):159–170, 1999.

[6] A. Daldegan, N.M. Thalmann, T. Kurihara, and D. Thalmann. An integrated system for modeling, animating and rendering hair. *Computer Graphics Forum(Eurographics'93)*, 12(3):211–221, 1993.

[7] R. Featherstone. *Robot Dynamics Algorithms*. Kluwer Academic Publishers, 1987.

[8] R. Fedkiw, J. Stam, and H.W. Jensen. Visual simulation of smoke. In *SIGGRAPH 01 Conference Proceedings*, pages 15–22, 2001.

[9] M. Fong. Animating monster fur. In *SIGGRAPH course 36 notes*, 2001.

[10] D. Goldman. Fake fur rendering. In *Proc. of SIGGRAPH'97*, pages 127–134, 1997.

[11] S. Hadap and N. Magnenat-Thalmann. Interactive hair styler based on fluid flow. In *Computer Animation and Simulation 2000. Proceedings of the Eleventh Eurographics Workshop*, 2000.

[12] S. Hadap and N. Magnenat-Thalmann. Modeling dynamic hair as continuum. In *Eurographics Proceedings. Computer Graphics Forum, Vol.20,No.3*, 2001.

[13] R.R. Ogle Jr. and M. J. Fox. *Atlas of Human Hair(Microscopic Characteristics)*. CRC Press, 1999.

[14] J. Kajiya and T. Kay. Rendering fur with three dimensional textures. In *Proc. of SIGGRAPH'89*, pages 271–280, 1989.

[15] J.T. Kajiya and B.P. von Herzen. Ray tracing volume densities. *Computer Graphics (SIGGRAPH 84 Proceedings)*, 18(3), 1984.

[16] T.-Y. Kim and U. Neumann. A thin shell volume for modeling human hair. In *Proc. of IEEE Computer Animation*, 2000.

[17] T.-Y. Kim and U. Neumann. Opacity shadow maps. In *Proc. of Eurographics Workshop on Rendering*, pages 177–182, 2001.

[18] C. K. Koh and Z. Huang. A simple physics model to animate human hair modeled in 2d strips in real time. In *Proceedings of Eurographics Computer Animation and Simulation*, 2001.

[19] A.M. LeBlanc, R. Turner, and D. Thalmann. Rendering hair using pixel blending and shadow buffers. *Journal of Visualization and Computer Animation*, pages 92–97, 1991.

[20] D.-W. Lee and H.-S. Ko. Natural hairstyle modeling and animation. *Graphics Models and Image Processing*, 63:67–85, 2001.

[21] J. Lengyel. Real-time hair. In *Proc. of Eurographics Workshop on Rendering*, pages 243–256, 2000.

[22] T. Lokovic and E. Veach. Deep shadow maps. In *Proc. of SIGGRAPH'00*, pages 385–392, 2000.

[23] A.G. Lyne and B.F. Short. Biology of the skin and hair growth. In *Proc. of a Symposium held at Canberra, Australia*, 1964.

[24] Multibody dynamics package. http://robotics.stanford.edu/ kuffner/software. Developed by J. Kuffner and B. Mirtich.

[25] R.M. Murray, Z. Li, and S.S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[26] K. Perlin. An image synthesizer. In *Proc. of SIGGRAPH'85*, pages 287–296, 1985.

[27] K. Perlin and E.M. Hoffert. Hypertexture. In *Proc. of SIGGRAPH'89*, pages 253–262, 1989.

[28] E. Plante, M.-P. Cani, and P. Poulin. A layered wisps model for simulating interactions inside long hair. In *Proceedings of Eurographics Computer Animation and Simulation*, 2001.

[29] R.E. Rosenblum, W.E. Carlson, and E. Tripp. Simulating the structure and dynamics of human hair: Modeling, rendering and animation. *The Journal of Visualization and Computer Animation*, 2:141–148, 1991.

[30] M.P. Salisbury, S.E. Anderson, R. Barzel, and D.H. Salesin. Interactive pen and ink illustration. In *Proc. of SIGGRAPH'94*, pages 101–108, 1994.

[31] M.T. Scali-Sheahan. *18 Men's Styles*. Milady Publishing Company, Albany, NY, 1994.

[32] Shag (plugin for 3d studio max). home.abac.com/ddag/hair.html.

[33] Shave (plugin for lightwave). www.joealter.com/software.html.

[34] J. Stam. *Multi-Scale Stochastic Modeling of Complex Natural Phenomena*. PhD thesis, University of Toronto, 1995.

[35] D. Terzopoulos, J.C. Platt, and A.H. Barr. Elastically deformable models. In *Proceedings of SIGGRAPH'87*, pages 205–214, 1987.

[36] Y. Watanabe and Y. Suenaga. A trigonal prism-based method for hair image generation. *IEEE Computer Graphics and Applications*, 12(1):47–53, 1992.

[37] J. Wilhelms. Using dynamic analysis for realistic animation of articulated bodies. *IEEE CG&A*, 7(2):12–27, 1987.

[38] X.D. Yan, Z. Xu, J. Yang, and T. Wang. The cluster hair model. *Graphics Models and Image Processing*, 1999.

[39] K. Young. *28 Styles for Student Practice with Basic Cutting and Styling Guides*. Milady Publishing Company, Albany, NY, 1992.

[40] Y. Yu. Modeling realistic virtual hairstyles. In *Proceedings of Pacific Graphics*, pages 295–304, 2001.

[41] O.C. Zienkiewicz and R.L. Taylor. *The Finite Element Method: Solid and Fluid Mechanics Dynamics and Non-Linearity*. McGraw-Hill Book Company, 1989.

Algorithms for Hardware Accelertaed Hair Rendering

Tae-Yong Kim*
tae@rhythm.com

Rhythm & Hues Studio

*formerly at the University of Southern California

In this talk, I will discuss issues related to hair rendering and introduce practical algorithms for hardware accelerated hair rendering. More specifically, I will introduce a simple antialiasing algorithm amenable for hardware accelerated hair drawing, the opacity shadow maps algorithm for hair self-shadow computation, and a programmable shader implementation of Kajyia-Kay hair shading model. All the examples are shown in an OpenGL-fashion, but it should be straightforward to adapt these algorithms to other standard APIs such as Direct3D.

Topics covered are

- Issues in rendering hair with graphics hardware
- A brief overview of self-shadow computation algorithms (shadow buffer, deep shadow maps, and opacity shadow maps)
- Self-shadows generation with graphics hardware (opacity shadow maps)
- Bin sort based visibility ordering for antialiased hair drawing
- Local shading computation with programmable graphics hardware

Additional Materials

1. Tae-Yong Kim and Ulrich Neumann, Opacity Shadow Maps, Eurographics Rendering Workshop 2001 (reprinted in the course note).
2. Tae-Yong Kim, Modeling, Rendering, and Animating Human Hair, Ph. D. Dissertation, University of Southern California, 2002 (available at http://graphics.usc.edu/~taeyong)

1. Introduction

Hair is considered one of the most time-consuming objects to render. There are many reasons why hair rendering becomes such a time-consuming task.

First of all, when rendering hair, we deal with a very complex geometry. The number of hair strands often ranges from 100,000 (for the case of human hair) to some millions (animal fur). Moreover, each hair strand can have geometrically non-trivial shape. For example, let's assume that each hair strand is drawn with 20 triangles. A simple multiplication says that we'd be dealing with a large geometry consisting of several or tens of million triangles! This geometric intricacy complicates any task related to hair rendering.

Second issue is the unique nature of the hair geometry. A hair strand is extremely thin in diameter (~0.1 mm), but can be as long as it grows. This property causes a severe undersampling problem, aliasing. The sampling theorem dictates that the number of samples to reconstruct a signal (in our case, hair geometry) should be higher than the maximum frequency of the signal. Assume that we draw a hair strand as thin triangle strips. According to the sampling theory, the size of a pixel[1] should be smaller than half the thickness of the thinnest hair. In practice, this is equivalent to having an image resolution of 10,000 by 10,000 pixels when the entire screen is approximately covered by somebody's hair. Moreover, when hairs are far away, the required sampling rate should increase! The current display devices hardly reach this limit, and are not likely to reach this limit in the near future. Thus, correct sampling becomes a fundamental issue for any practical hair rendering algorithms.

Third issue is the optical property of hair fibers. A hair fiber not only blocks, but also transmits and scatters the incoming light. As an aggregated form, hairs affect the amount of lighting onto each other. For example, a hair fiber can cast shadows onto other hairs as well as receive lights transmitted through other hairs. Due to the unique geometric shape of hair, the amount of light a hair fiber reflects and scatters varies depending on the relationship between hair growth direction, light direction, and eye position. This effect is known as anistropic reflectance, and defines one of the most prominent characteristics of a hair image (you can easily notice that the direction of the highlight is always perpendicular to the direction of hair growth).

All these issues (number of hairs, sampling issues, and complexity of lighting) make hair rendering a computationally demanding task. In a naïve form, a software renderer[2] (that is not parallelized, and does not utilize any graphics hardware capability) will demand significant computation time. Fortunately, recent progresses in graphics hardware shed some lights. The fastest GPUs at the time of this writing (march, 2003) can now render up to 80 million triangles per second, or 2 ~ 3 million triangles per frame (30 fps). More promisingly, the raw performance of current GPUs increases at a faster rate than that of the general purpose CPUs. So, it seems natural to consider hardware acceleration methods for hair rendering. However, one should

[1] A pixel is essentially a point sample. The extent of a sampling region and the pixel sample (color, depth…) should be differentiated. For convenience, we let the size of a pixel denote that of the sampling region.

[2] Here a 'software renderer' refers to a rendering program that is solely dependent on general purpose CPUs. In contrast, a 'hardware renderer' refers to a rendering program that utilizes specialized graphics hardware (such as OpenGL API). In the note, the term 'hardware' will not really mean a dedicated hardware for hair rendering although there is no reason why there can't be such hardware!

keep in mind that most existing graphics cards are not designed for small objects such as hairs. These create a number of difficulties when we use graphics hardware for hair rendering.

2. Tiny, tiny triangles

A hair fiber is naturally represented with a curved cylinder. Thus, it is tempting to draw a hair as some tessellated version of a cylinder (Figure 1).

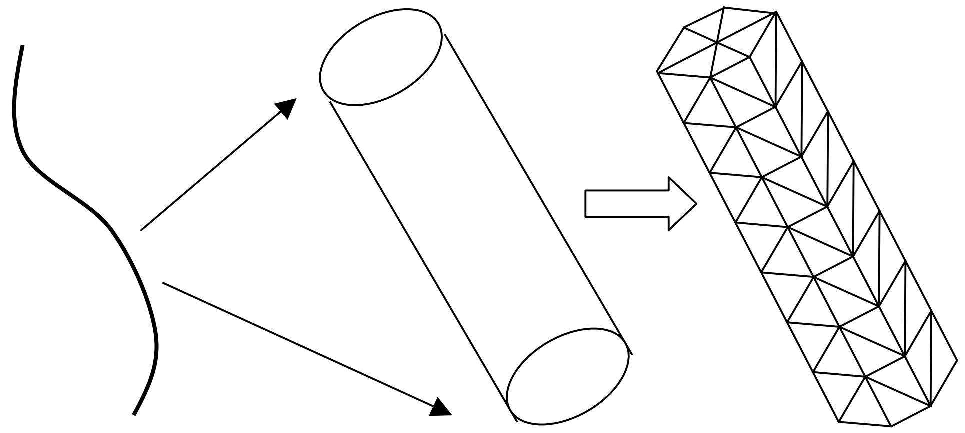


Figure 1. A hair as tessellated cylinder.

This model is totally valid if we were living in a microscopic world where we see just a few hairs in our view. In practice, we deal with so many hairs that this naïve method would generate too many triangles. Moreover, a hair is so thin that the curved shape of the cylinder will be rarely noticeable. Alternatively, we can approximate hair as a flat ribbon that always faces towards the camera (Figure2). In practice, this model approximates hair very well since variation of color along hair's thickness is often ignorable.

We can further simplify the geometry and draw hair as a connected line strips (Figure

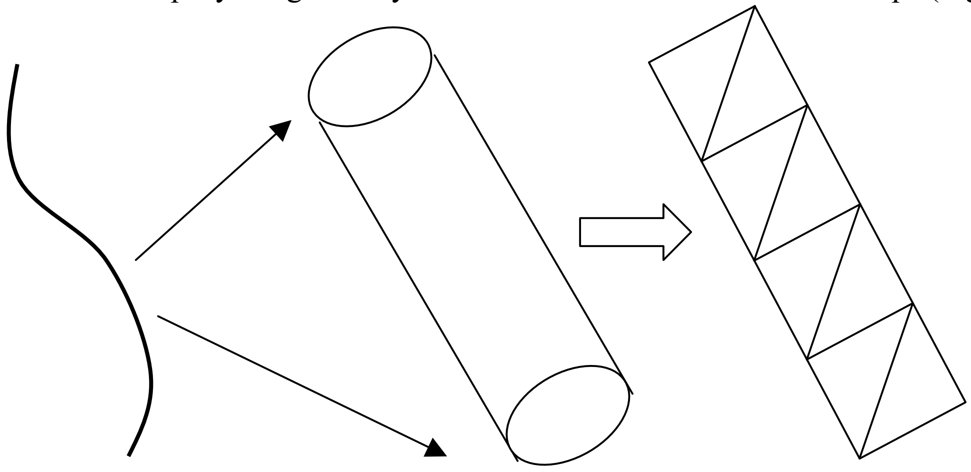


Figure 2. Hair as a flat ribbon.

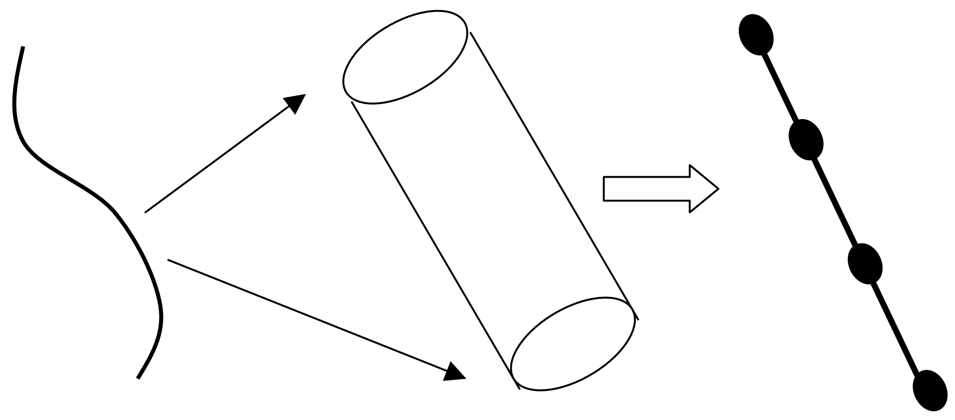


Figure 3. Hair as a line strip.

3). Although mathematically a line should be infinitesimally thin, a line in this case is associated with some artificial thickness value (often a pixel's width).

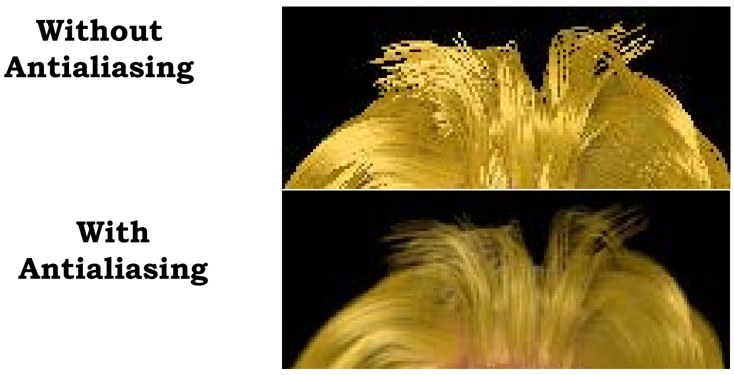
For the convenience of discussion, I will use the line strip as our hair representation, but discussions and algorithms here equally apply to the polygonal ribbon representation. Let's assume that a hair strand is approximated with a number of points p0, p1,… pn-1 and its associated colors c0,c1,…cn-1. The following code will draw a hair as a connected line strip.

```
DrawHair(p0,p1,..,pn-1,c0,c1,….cn-1)
{
        glBegin(GL_LINE_STRIP)
        glColor3fv(c0);
        glVertex3fv(p0);
        glColor3fv(c1);
        glVertex3fv(p1);
        …
        glColor3fv(cn-1);
        glVertex(pn-1);
        glEnd()
}
```

Routine1. DrawHair

Optimistically, by calling this function repeatedly, you might think that we will able to draw as many hairs as we want. Unfortunately, it is not that simple…

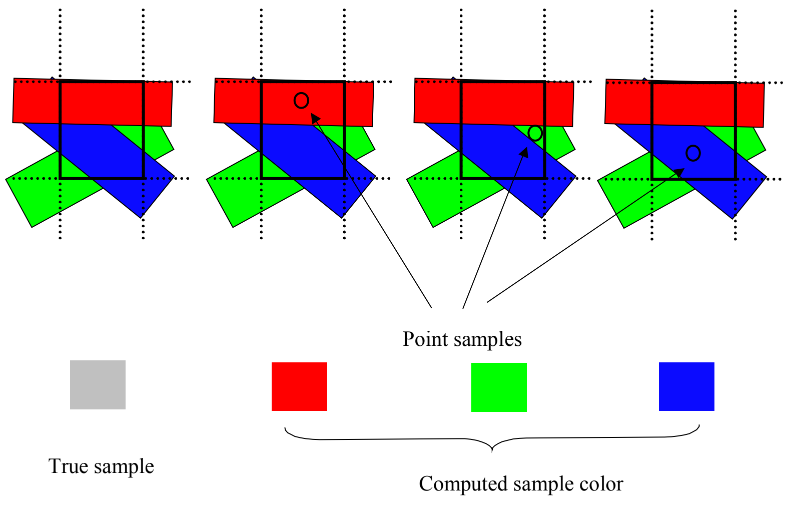
Figure 4. Importance of antialiasing in hair rendering



When many lines are drawn, the approach will suffer from severe aliasing artifacts as shown in the image above (without antialiasing). Current graphics hardware almost always relies on the Z-buffer algorithm to determine whether a pixel's color should be overwritten. The z-buffer algorithm is a point sampling algorithm. A pixel's color (or depth) is determined entirely by a limited number of point samples (the default setting being just one sample per pixel).

See Figure 5. Assume that three lines cover a pixel and each line's color is red, green, and blue, respectively. If each line covers exactly one third of the pixel's extent, the correct color sample of the pixel should be an averaged color of the three colors - gray. Unfortunately, a single point sample will cause the pixel to change in color to either of the three. So, instead of gray (a true sample), the pixel's color will alternate in red, blue, and green, depending on the point sample's position.

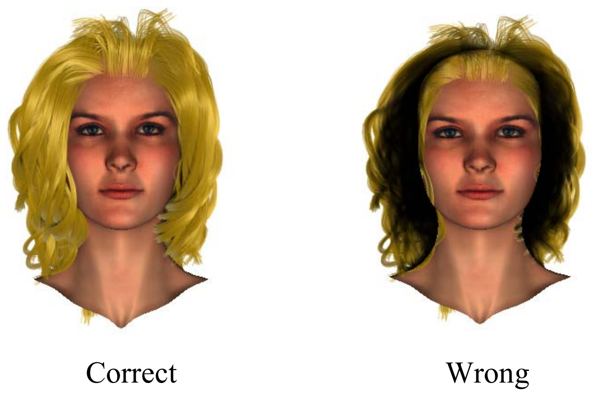
Figure 5. Consequence of point sampling



Now we are aware that the Z-buffer, the most common sampling algorithm in many graphics hardware, is not designed for small objects such as hair. In a point sample-based algorithm such as Z-buffering, the number of point samples determines the quality of the final image. The required number of samples is closely related to the complexity of the scene. The rule of thumb is that there should be at least as many samples as the number of objects that fit in a pixel. That's why we often don't see much aliasing when we draw relatively large triangles, but in hair. There are ways to increase the number of samples. The most common method is the accumulation buffer. In this method, the number of samples per pixel corresponds to the number of accumulation steps performed. However, accumulation buffers tend to be slow in many OpenGL implementations and the accumulation steps must be performed at every frame.

The thickness of a hair is often much smaller than the size of a pixel. So, it seems natural to draw a line with small alpha value and the attempt will prove fine for one line. However, as more lines (hairs) are drawn, we will encounter a similar problem we had before. The alpha blending in OpenGL requires that the scene should be sorted by the distance from the camera. Otherwise, the image will not look right – you will *see through* the pixels (Figure 6).

Figure 6. Alpha blending needs correct visibility ordering



In short, we need to 1) sample each hair correctly, 2) draw each hair with the correct thickness, and 3) blend the colors of all the hairs correctly. Many current graphics hardware offer decent, if not perfect, hardware accelerated antialiased line drawing features. To draw each hair with correct sampling, we can exploit the feature. To draw each hair with the correct thickness, we set the alpha value of each line to a small (<1.0) value. To blend the colors correctly, we use the alpha blending with the correct visibility order. Both hardware line antialiasing and alpha blending require a correct visibility order. So, we will do the visibility ordering by ourselves, departing from the troublesome Z-buffering.

2.1 Further Readings

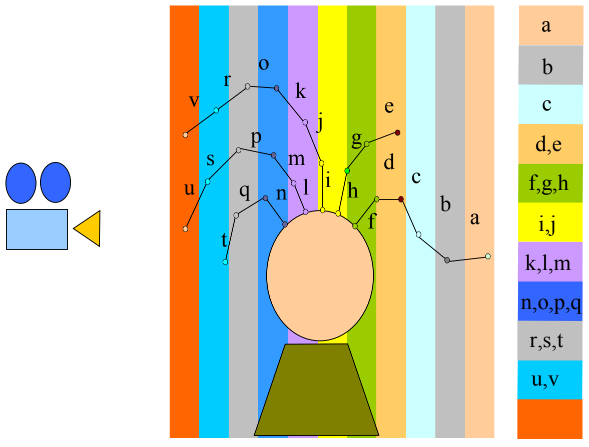
For rigorous discussions on issues with point sampling, refer to the following papers.

- D. Mitchell, Consequences of stratified sampling in graphics, SIGGRAPH Proceedings, 1996, 277-280
- T. Lokovic and E. Veach. Deep shadow maps, SIGGRAPH Proceedings, 2000, 385-392
- A. R. Smith. A Pixel is Not A Little Square, A Pixel is Not A Little Square, A Pixel is Not A Little Square! (And a Voxel is Not a Little Cube), Tech Memo 6, Microsoft, Jul 1995

3. Visibility ordering for antialiased hair drawing

For correct visibility computation, we need to draw hair far to near (or near-to-far as long as it is consistent). Assume that each hair is broken into line segments, and we draw a large number of such line segments for the entire hair model. Antialiasing can be performed in two steps. First, the visibility order of a given hair model is determined based on the distance to the camera (Figure 7). 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 colored bar in Figure 7), 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.

Figure 7. Bin-based Visibility Ordering



The end points of line segments are grouped into a set of bins that slice the entire volume (Figure 7). The bin enclosing each point is found by

$$i = \left\lfloor N \frac{D - D_{\min}}{D_{\max} - D_{\min} + \varepsilon} \right\rfloor, 0 \leq i < N$$

, where i is the index for the bin and N is the number of bins. D is the distance from the point to the image plane. D_{min} and D_{max} are minimum and maximum of such distances, respectively. ε is a constant such that $\varepsilon < \frac{D_{\max} - D_{\min}}{N}$.

Given a point $\vec{p}$ and the camera positioned at $\vec{c}$ looking in direction of $\vec{d}$, the distance to the image plane is computed by

$$D = (\vec{p} - \vec{c}) \cdot \vec{d}$$

Precise visibility ordering of lines is difficult to obtain by depth sorting alone. When lines are nearly parallel to the image plane, the errors are small, provided the spacing between bins is dense enough. However, when line segments extend over many bins, the visibility order cannot be determined either by maximum depth or minimum depth. Such lines could be further subdivided. However, on the other hand, the pixel coverage of such a line tends to be small. For example, when a line is perpendicular to the image plane, the pixel coverage of the line is at most a single pixel. In practice, using maximum depth for every line produces good results.

In the drawing pass, each bin is accessed from the farthest to the closest. For each bin, the line segments whose farther points belong to the bin are drawn. Each line segment is drawn with hardware antialiasing. The color of each line segment is accumulated using

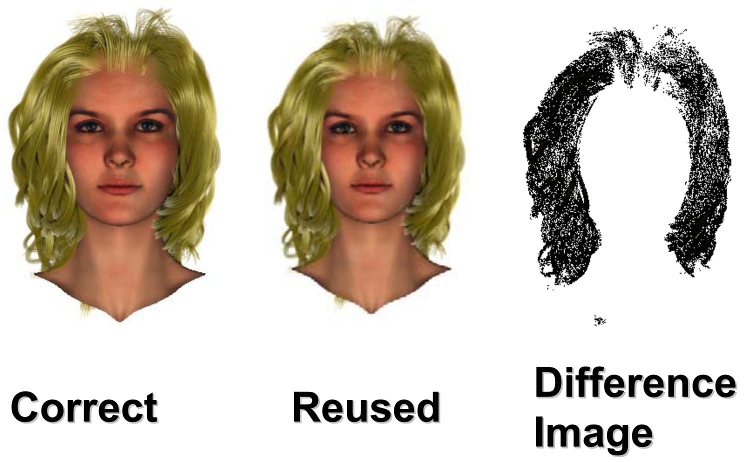
glBlendFunc(GL_SRC_ALPHA,GL_ONE_MINUS_SRC_ALPHA)

The bin-based visibility ordering algorithm can be summarized as follows.

```
Visibility ordering pass:
For every line segment,
        1. compute the depth of the end points.
        2. using the larger depth, compute the bin location
        3. add the index of the line to the bin.
Drawing pass:
1. Enable the depth buffer
2. Draw all the other scene objects
3. Disable the depth buffer
4. For each bin, from the farthest bin to the nearest,
        Draw all the line segments whose indices are stored in the bin
```

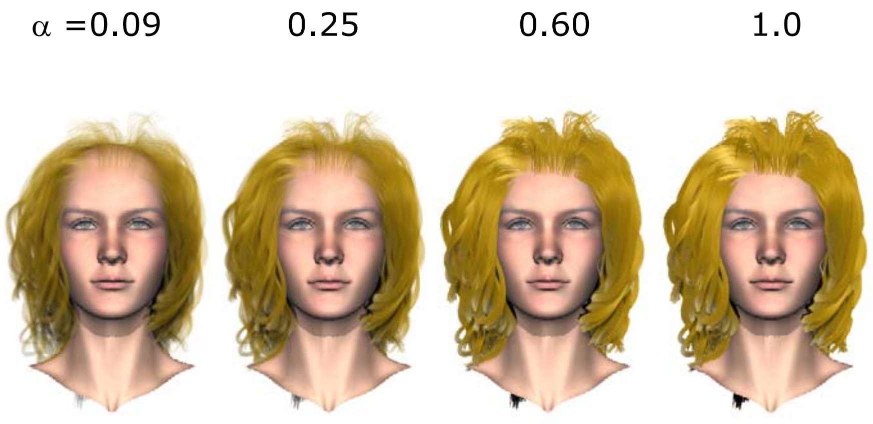
Note that since the line segments are drawn in a correct order, all the hardware features can be now happily exploited (antialiased line drawing, alpha blending). Although simple, the method is fast and converges to exact ordering as hair strands are drawn with more segments. The visibility-ordering algorithm runs at about 700,000 lines per second on a Pentium III 700Mhz CPU. Another benefit is that we can separate the visibility ordering pass from the actual drawing pass. For example, during interactive modeling, the viewpoint does not change much from frame to frame. This coherence enables performing the visibility ordering periodically and reusing the computed order for subsequent frames. In Figure 8, the image in the middle was drawn with a previously computed visibility order where the head model was rotated by 30 degree along the y-axis. Although the difference image shows that there's some discrepancy against the correct order (left image), visual degradation is not so objectionable. In contrast, depth buffer based super-sampling methods (e.g., accumulation buffer) must compute visibility at every frame.

Figure 8. Coherence of Visibility Order



In addition, the alpha values of line segments can control the perceived thickness of hair strands. As hair strands become thinner, super-sampling methods would require more samples while alpha value changes suffice in the visibility-ordered hair model (Figure 9).

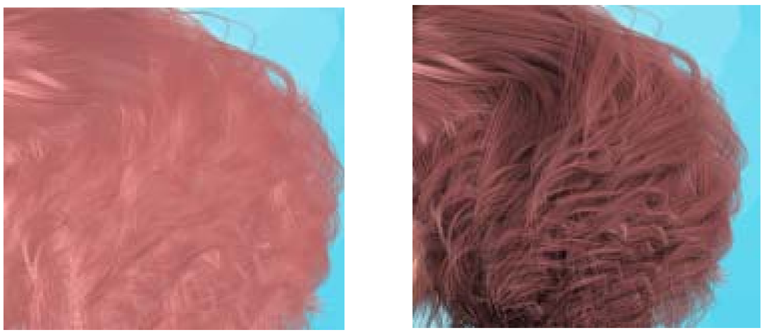
Figure 9. Thickness Change



4. Self-shadows

Hairs cast shadows onto each other, as well as receive and cast shadows from/to other objects in the scene. Especially, self-shadows create crucial patterns that distinguish one hairstyle from others. Without self-shadows, the underlying structure of a hair model cannot be correctly visualized (Figure10). This section introduces an efficient shadow generation algorithm that makes full use of graphics hardware accelerator.

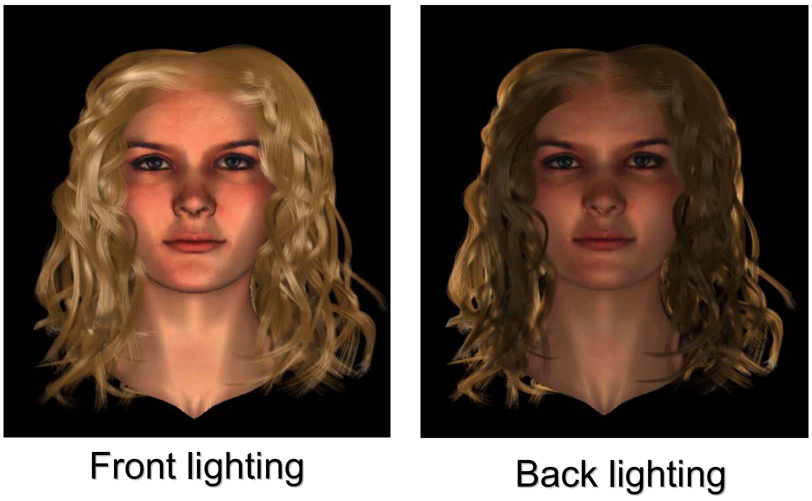
Figure 10. Self-shadows are crucial for volumetric hair



There are mainly two issues with self-shadows in hair rendering – the thin geometry of the hair fiber and the translucency. The thin geometry of hair causes serious aliasing artifacts in shadow computation in a very similar way as in the previous section. This is not surprising if we note that the shadow computation is just one instance of the more general visibility computation problem. For hair drawing, we computed the visibility of each hair from the camera. For shadow computation, we need to compute the visibility of each hair from the *light source*. If the hair *sees* more lights, it will receive more illumination from the source, and if the hair can't see the light, it will be left dark (or shadowed).

Another important aspect in hair self-shadowing is that a hair fiber often does not completely block the incoming light. It not only reflects and scatters the incoming illumination, but often lets the light pass through. This unique property of the hair fiber is the most clearly observable in the 'back lighting' situation where the silhouette shines brightly when the light is put behind (Figure 11).

Figure 11. Translucency of hair



In the previous section, we discussed the problems in using Z-buffer for hair rendering. When the Z-buffer is used for shadow computation, it is called 'shadow map'. In this *depth-based shadow map* (DBSM), the scene is rendered from the light's point of view and the depth values are stored. Each point to be shadowed is projected to the light's camera and the point's depth is checked against the depth in the shadow map.

One attractive feature of the traditional shadow map is that the shadow map can be generated with hardware by rendering the scene from the light's point of view and storing the resulting depth buffer. However, severe aliasing artifacts can occur with small semi-transparent objects. As discussed in the previous section, in a dense volume made of small primitives, depths can vary radically over small changes in image space. The discrete nature of depth sampling limits DBSM in handling such objects. The binary decision in depth testing inherently precludes any translucency. Thus, DBSM is unsuited for volumetric objects such as hairs.

The transmittance $\tau(p)$ of a light to a point p can be written as

$$\tau(p) = \exp(-\Omega)\text{, where } \Omega = \int_0^l \sigma_t(l')dl' \quad (1)$$

In (1), l is the length of a path from the light to the point, σ_t is the extinction (or a density) function along the path. Ω is the *opacity* value at the point.

The deep shadow maps (DSM) algorithm originally presented at SIGGRAPH 2000 proposed that each pixel stores a piecewise linear approximation of the transmittance function instead of a single depth, yielding more precise shadow computation than DBSM. Deep shadow maps account for the two important properties of hair shadows.

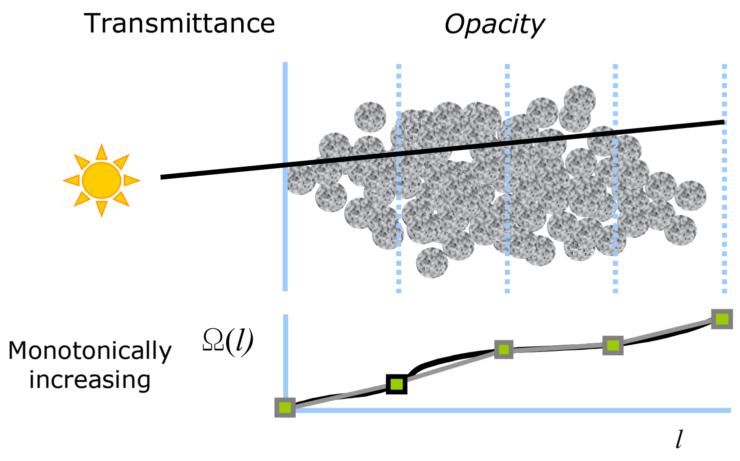
Partial visibility: In the context of shadow maps, the transmittance function can be viewed as a partial visibility function from the light's point of view. If more hairs are seen along the path from the light, the light will be more attenuated (occluded), resulting in less illumination (shadow). As noted earlier (recall Figure 5), visibility can change drastically over the pixel's extent. The transmittance function handles this partial visibility problem by correctly integrating and filtering all the contributions from the underlying geometry.

Translucency: a hair fiber not only reflects, but also scatters and transmits the incoming light. Assuming that the hair fiber transmits the incoming light only in a forward direction, the translucency is also handled by the transmittance function.

Despite the compactness and quality, however, due to the underlying data structure (linked list), a hardware implementation becomes tricky with deep shadow maps.

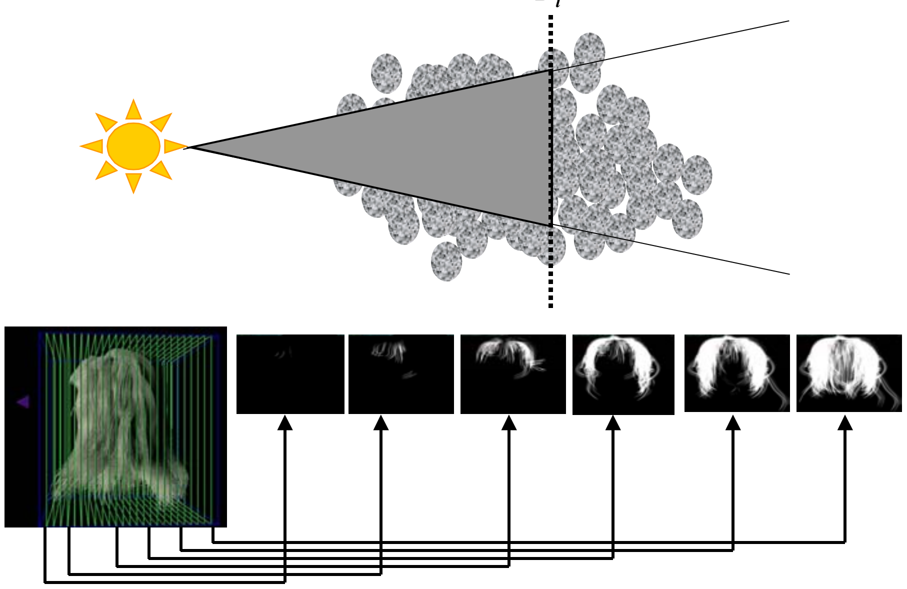
The Opacity Shadow Maps (or OSM) was originally designed as a fast alternative to DSM for computing the transmittance function. Opacity shadow maps (or OSM) algorithm uses a set of parallel opacity maps oriented perpendicular to the light's direction (Figure 12).

Figure 12. Opacity Shadow Maps



By approximating the transmittance function with discrete planar maps, opacity maps can be efficiently generated with graphics hardware. On each opacity map, the hair model is rendered from the light's point of view, clipped by the map's depth (Figure 13). Instead of storing depth values, each pixel stores Ω, the line integral of densities along the path from the light to the pixel. The opacity values from adjacent maps are then sampled and interpolated during rendering.

Figure 13. Opacity Maps



4.1 Basic Algorithm

Opacity shadow maps heavily rely on graphics hardware and operate on any bounded volumes represented by standard primitives such as points, lines and polygons. (In our context, hairs are represented as a cluster of lines.) The hair volume is sliced with a set of opacity map planes perpendicular to the light's direction. The scene is rendered to the alpha buffer, clipped by each map's depth. Each primitive contributes its associated alpha value. The alpha value is a user-controllable parameter that depends on the size (thickness) and the optical property of hair. It also depends on the resolution of the opacity maps. Each pixel in the map stores an alpha value that approximates the opacity relative to the light at the pixel's position. The opacity values of adjacent maps are sampled and linearly interpolated at the position of each shadow computation point, to be used in a shadowed shading calculation.

The pseudo code in Routine 2 uses the following notation. P is the set of all the shadow computation sample points (or simply *shadow samples*). N is the number of maps and M is the number of shadow samples. D_i is the distance from the opacity map plane to the light $(1 \le i \le N)$. P_i is a set of shadow samples that reside between D_i and D_{i-1}. p_j is j_{th} shadow sample $(1 \le j \le M)$. *Depth*(p) returns a distance from p to the light. $\Omega(p_j)$ stores the opacity at p_j. $\tau(p_j)$ is the transmittance at p_j. B_{prev} and $B_{current}$ are the previous and current opacity map buffers.

Routine 2. Opacity Shadow Maps

```
1.  $D_0$ = Min ($Depth$($p_j$)) for all $p_j$ in P ($1 \le j \le M$)
2.  for ($1 \le i \le N$)                                          (Loop 1)
3.      Determine the opacity map's depth $D_i$ from the light
4.  for each shadow sample point $p_j$ in P ($1 \le j \le M$)      (Loop 2)
5.      Find $i$ such that $D_{i-1} \le Depth(p_j) < D_i$
6.      Add the point $p_j$ to $P_i$.
7.  Clear the alpha buffer and the opacity maps $B_{prev}$, $B_{current}$.
8.  for ($1 \le i \le N$)                                          (Loop 3)
9.      Swap $B_{prev}$ and $B_{current}$.
10.     Render the scene clipping it with $D_{i-1}$ and $D_i$.
11.     Read back the alpha buffer to $B_{current}$.
12.     for each shadow sample point $p_k$ in $P_i$                (Loop 4)
13.         $\Omega_{prev} = sample(B_{prev}, p_k)$
14.         $\Omega_{current} = sample(B_{current}, p_k)$
15.         $\Omega$ = interpolate (Depth($p_k$), $D_{i-1}$, $D_i$, $\Omega_{prev}$, $\Omega_{current}$)
16.         $\tau(p_k) = e^{-\kappa\Omega}$
```

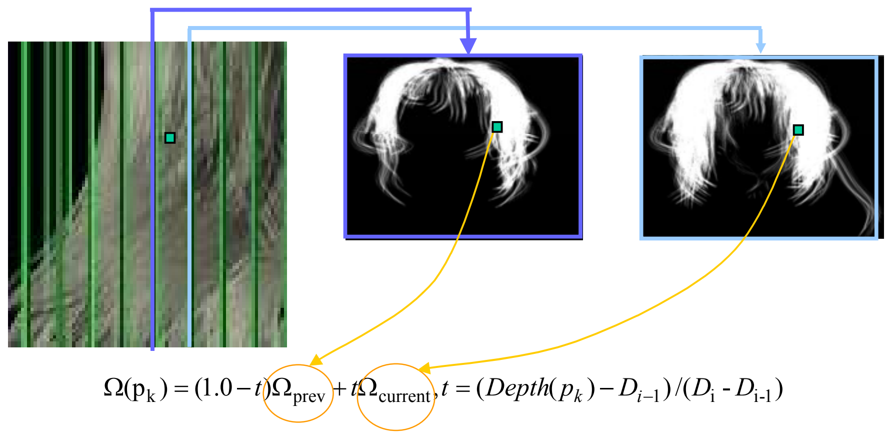
In loop 1, the depth of each map is determined. Uniform slice spacing is reasonable for evenly distributed volumes. Prior to shadow computation, shadow samples are prepared. In our hair representation, the primitives (line segments) tend to be very small and thus the end points of lines often suffice. Thus, for each hair strand, we choose the endpoints of line segments as shadow samples. More samples can be taken if needed. When many samples are required for each primitive, it may be useful to pre-compute the visibility and use only the visible points as shadow samples. Loop 2 prepares a list of shadow samples that belong to each buffer. Note that this procedure is exactly the same as the bin-based visibility sorting method in section 3. Thus, the same code can be reused here.

Each pixel in the map stores the opacity value, which is a summation that produces the integral term Ω in equation (1). Thus each line segment can be rendered antialiased with full hardware support in any order (the order can be arbitrary due to the commutative nature of integration). The alpha buffer is accumulated each time the volume is drawn with the OpenGL blend mode *glBlendFunc(GL_ONE,GL_ONE).* The depth buffer is disabled. Clipping in line 10 ensures correct contribution of alpha values from the primitives and culls most primitives, speeding up the algorithm.

As loop 3 and 4 use only two opacity map buffers at a time, the memory requirement is independent of the total number of opacity maps computed. In loop 4, the shadow is computed only once for each sample. So, the amortized cost of the algorithm is linear in the number of samples. The overall complexity is *O(NM)* since the scene is rendered for each map, but the rendering cost is low with hardware acceleration.

The sample function in loop 4 can be any standard pixel sampling function such as a box filter, or higher-order filters such as Bartlett filter and Gaussian filter. For the examples shown here, a 3x3 averaging kernel is used. Such filtering is possible because alpha values are stored instead of depths. The sampled opacity values Ω_{prev}, $\Omega_{current}$ are linearly interpolated for each point p_k (Figure 14).

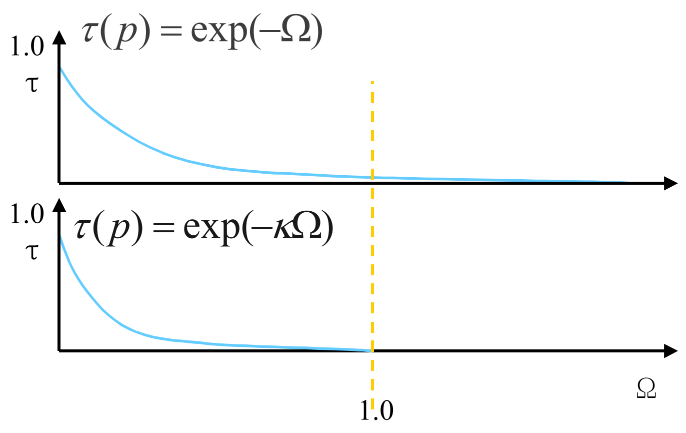
Figure 14. Interpolating the opacity values


$\Omega(\mathrm{p_k}) = (1.0 - t)\Omega_{\mathrm{prev}} + t\Omega_{\mathrm{current}}, t = (Depth(p_k) - D_{i-1})/(D_{\mathrm{i}} - D_{\mathrm{i\text{-}1}})$

A higher order interpolation may be used. For example, four buffers can be used for a cubic-spline interpolation.

A volume turns opaque as the opacity Ω reaches infinity. The quantization in the alpha channel limits the maximum amount of opacity that a pixel can represent. A constant κ in line 15 controls the scaling of opacity values such that $e^{-\kappa} = 2^{-d}$, where d is the number of bits per pixel (for example, κ is about 5.56 for 8 bit alpha buffer). Thus, an opacity value of 1.0 represents a complete opaqueness.

Figure 15. Scaling the opacity



4.2 Additional Clipping

A simple modification can yield a significant speedup in the basic algorithm. When shadow sample points coincide with the end points of the line segments, we can exploit the fact that the line segments are depth-sorted. At each opacity map generation step, we can check if the line segments reside in the current depth range (D_{i-1} and D_i). The line 10 in Routine 2 (shown below)

```
10.      Render the scene clipping it with $D_{i-1}$ and $D_i$.
```

can be augmented as

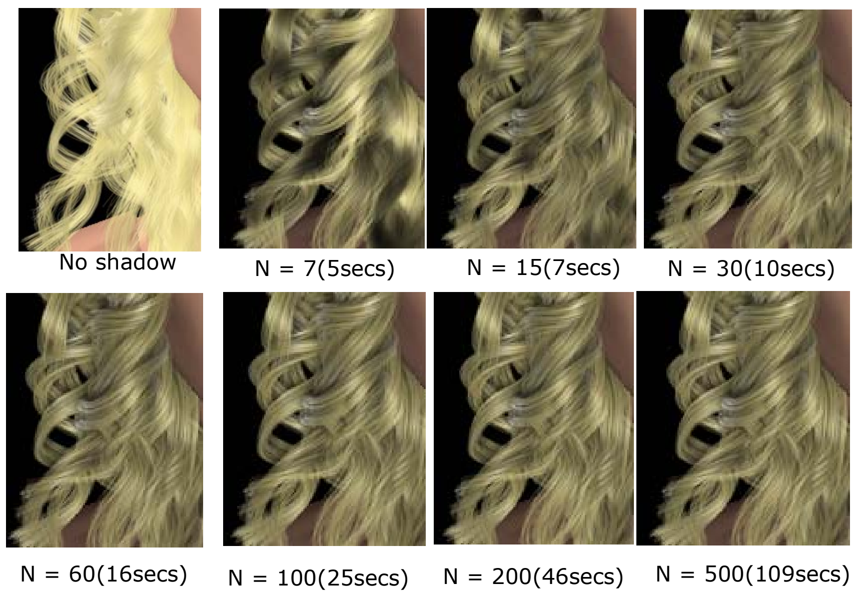
```
10.a     For each line segment (p1,p2) , compute Depth(p1) and Depth (p2).
10.b     Draw the line only if $D_{i-1}$ <.Depth(p1) < $D_i$ or $D_{i-1}$ <.Depth(p2) < $D_i$
```

With this additional scene culling scheme, the observed time complexity becomes close to *O(N).*

4.3 Examples

Figure 16 illustrates a test scene where the number of maps (N) was varied. Note that the tradeoff between speed and image quality can be achieved by varying the number of maps. The rendering time is linear in the number of maps when the basic algorithm was used. With the modification in section 4.2, the rendering time becomes sub-linear in the number of maps (about 12 secs for N = 500) since the number of primitives drawn to each map decreases as the number of maps increases.

Figure 16. Results



4.4 Further Readings

More details of the original deep shadow maps algorithm as well as an excellent discussion on aliasing in hair shadows can be found in

- T. Lokovic and E. Veach. Deep shadow maps, SIGGRAPH Proceedings, 2000, 385-392

For more discussions and results for the Opacity Shadow Maps algorithm, refer to

- Tae-Yong Kim and Ulrich Neumann, Opacity Shadow Maps, Eurographics Rendering Workshop 2001 (reprinted in the course note).

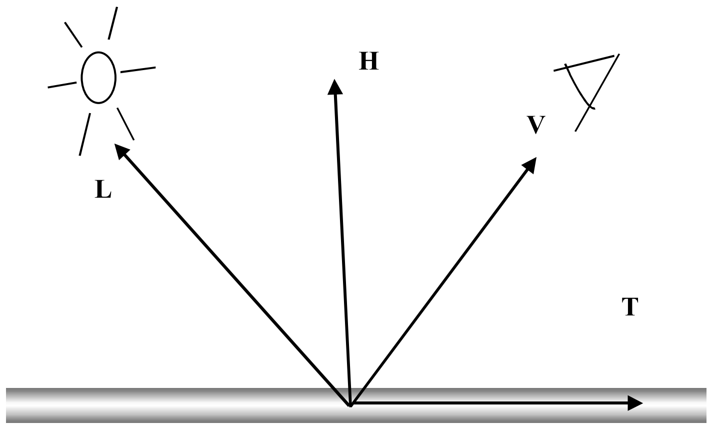
A modification on the transmittance function computation in OSM is suggested in

- Johnny Chang, Jingyi Jin, Yizhou Yu, A Practical Model for Mutual Hair Interactions, ACM SIGGRAPH Symposium on Computer Animation, July 2002 (reprinted in the course note).

5. Shading Model

A shading model describes how much a hair fiber reflects or transmits in a given direction when the hair fiber is fully lit. Since global aspects such as shadows are not accounted for, the term *local shading model* is often used. It is often assumed that the shape of a hair strand on its local neighborhood is a straight cylinder (Figure 17). A hair shading model is often constructed the function of three vectors, L, the light's direction, V, the viewing direction, and T, the tangent vector.

Figure 17. Hair shading geometry



The most commonly used shading model is the one originally developed by Kajiya and Kay. The model is composed of a Lambertian diffuse component and an anistropic specular component. In the Kajiya-Kay model, a Lambertian cosine falloff function is used for diffuse lighting. The closer the light is to the normal, the more illumination is received.

$$\Psi_{Diffuse} = K_d \sin(T, L)$$

, where K_d is a scaling parameter for the diffuse illumination and (V_1,V_2) denotes the angle between two vectors V_1,V_2.

A non-diffuse (specular) illumination is computed using the viewing vector V**.** The specular illumination becomes the biggest when the half vector H = (L + T) / 2 becomes perpendicular to the tangent vector. The Kajiya-Kay model computes

$$\Psi_{Specular} = K_s[(T \cdot L)(T \cdot V) + \sin(T, L)\sin(T, V)]^p$$

Thus, the amount of light a hair fiber scatters in the direction of V can be written as

$$\Psi_{Hair} = \Psi_{Diffuse} + \Psi_{Specular} = K_d \sin(T, L) + K_s[(T \cdot L)(T \cdot V) + \sin(T, L)\sin(T, V)]^p$$

Multiplying the transmittance function τ computed from section 4, the (shadowed) color of a point on the hair fiber can be expressed as

$$\Psi_{Hair} = \tau(\Psi_{Diffuse} + \Psi_{Specular})$$

To be accurate, the transmittance function and shading model should be computed at every pixel sample. However, the colors tend to smoothly vary along hair's length. In practice, computing the shaded color only at the end points of line segments often yield good results (analogous to the Gouraud shading for polygons).

5.1 Further Readings

Improvements on the original Kajiya-Kay model were introduced by Banks [1994] and Goldmann [1997]. Read the following papers for more details.

- J. Kajiya and T. Kay, Rendering fur with three-dimensional textures, SIGGRAPH Proceedings, Vol. 23, pp. 271-280, 1989.
- D. C. Banks, Illumination in diverse codimensions, SIGGRAPH Proceedings, pp. 327-334, 1994
- D. Goldman, Fake Fur Rendering, SIGGRAPH Proceedings, pp. 127-134, 1997.

6. Data structure

Since shadows are view-independent, it is often convenient to precompute the shadow values at the end points of each line segment and reuse the shadow values during the viewpoint change. For each line segment, we use the following data structure.

Pos1	Tangent1	Color1	Shadow1
Pos2	Tangent2	Color2	Shadow2

The position of each end point comes from the hair model. The tangent vectors are derived from the position. With these position and tangent vectors, the (unshadowed) colors are computed with the shading model. The shadow values are computed with the opacity shadow maps algorithm.

Then, the entire procedure of hair rendering using graphics hardware can be summarized as

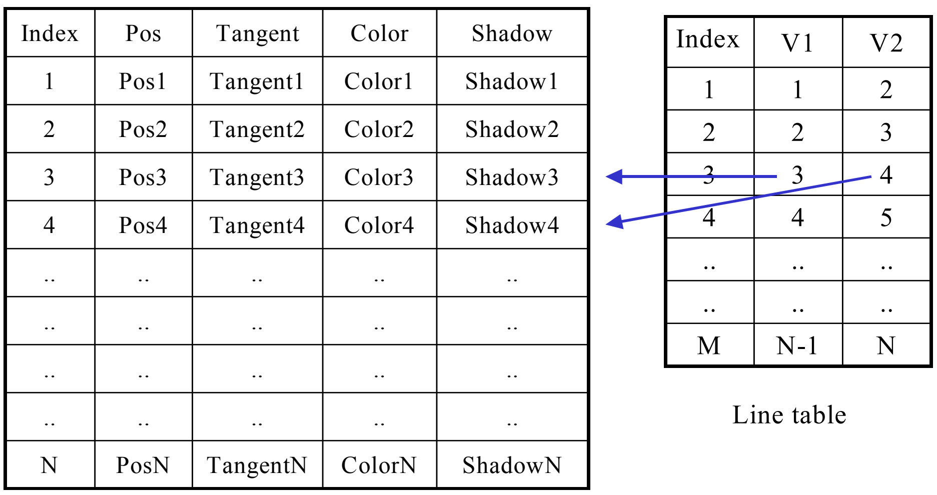
Routine 3

```
Setup pass:
    Compute the visibility order
    Compute shadow values
Drawing pass
    For each line segment $L_i$ ordered due to the visibility order
```

```
Set thickness (alpha value)
Compute the shaded color ($L_i$.color1)
Compute the shaded color ($L_i$.color2)
C1 = $L_i$.color1 * $L_i$.shadow1
C2 = $L_i$.color2 * $L_i$.shadow2
glBegin(GL_LINES)
glColor3fv(C1)
glVertex3fv($L_i$.Pos1)
glColor3fv(C2)
glVertex3fv($L_i$.Pos2)
glEnd()
```

Since line segments are connected, all the information shown above is duplicated. A more efficient implementation is thus to store all the position, color, tangent information in a vertex table and let each line be represented by indices to the table, as shown in Figure 18.

Figure 18. Data structures



Index	Pos	Tangent	Color	Shadow
1	Pos1	Tangent1	Color1	Shadow1
2	Pos2	Tangent2	Color2	Shadow2
3	Pos3	Tangent3	Color3	Shadow3
4	Pos4	Tangent4	Color4	Shadow4
..	..	..	..	..
..	..	..	..	..
..	..	..	..	..
..	..	..	..	..
N	PosN	TangentN	ColorN	ShadowN

Vertex table

Index	V1	V2
1	1	2
2	2	3
3	3	4
4	4	5
..	..	..
..	..	..
M	N-1	N

Line table

7. Hair shading with Programmable Vertex Shader

The computation of shaded color can be further accelerated with the use of programmable vertex shader. In this case, the color field will be computed on the fly inside the vertex shader. Routine 3 will change as follows.

Routine 4

Setup pass:

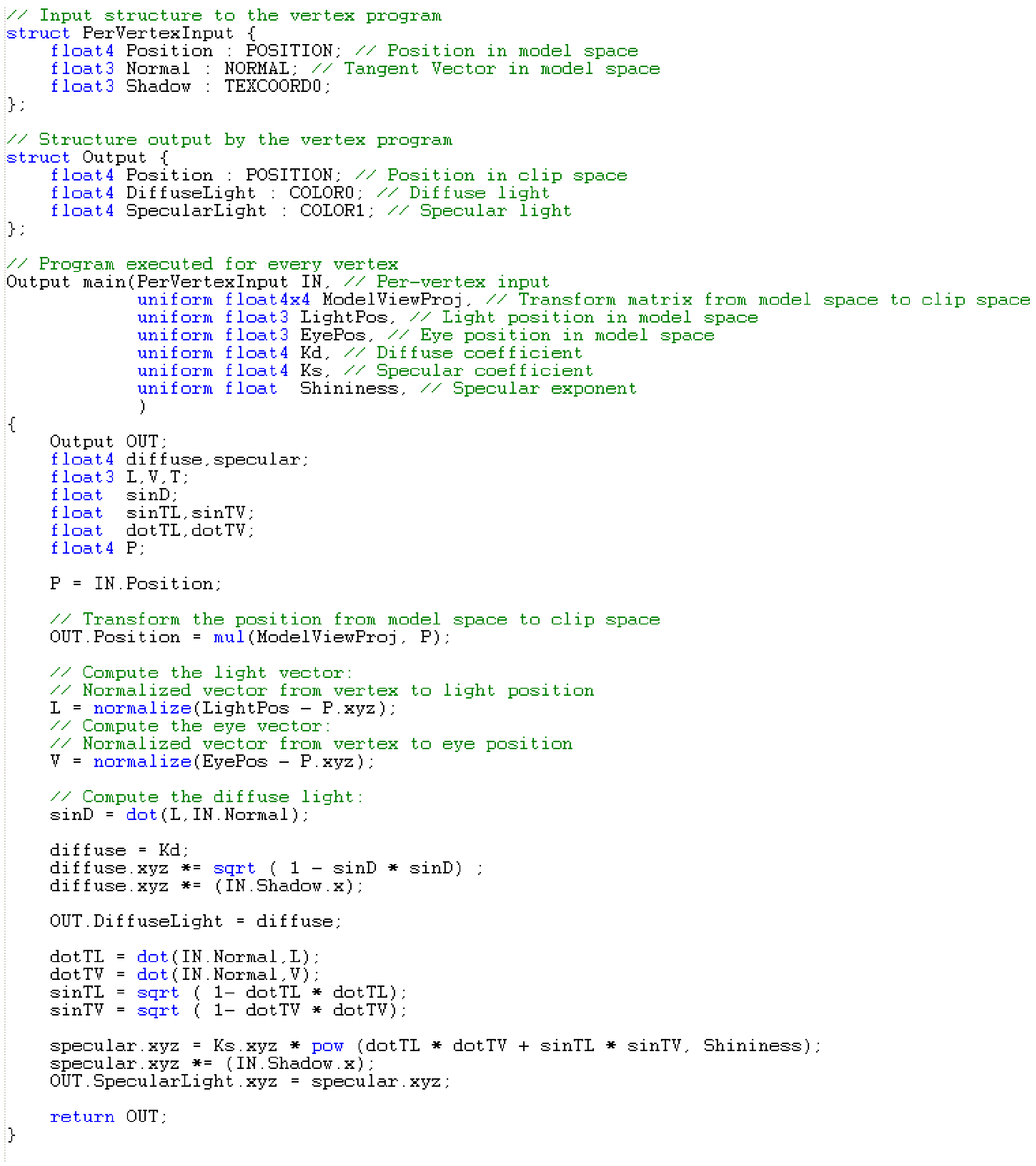
```
    Compute the visibility order
    Compute shadow values
```

Drawing pass:

```
For each line segment L_i ordered due to the visibility order
    Set thickness (alpha value)
    Draw L_i with programmable shader
```

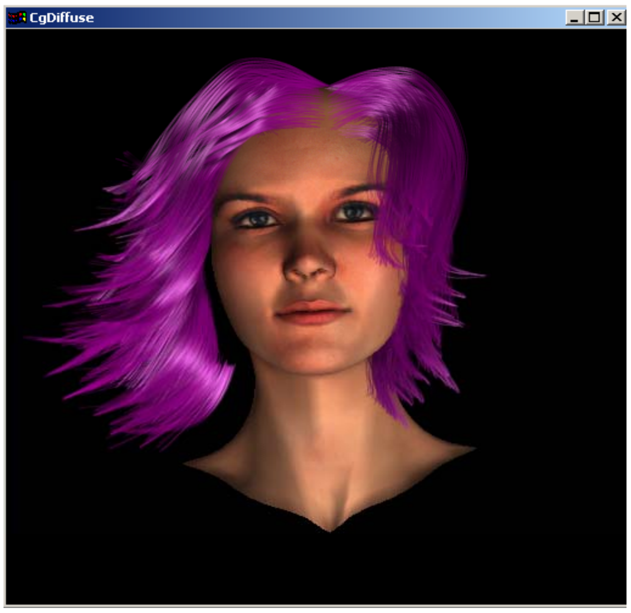
The inputs to the vertex shader are the camera position, the light position, shading parameters, position, tangent vector, and shadow values for each vertex. An example implementation of a hair shader is given in Figure 19 as nVidia CG program.

Figure 19. cg program for Kajiya-Kay shading model



```
// Input structure to the vertex program
struct PerVertexInput {
    float4 Position : POSITION; // Position in model space
    float3 Normal : NORMAL; // Tangent Vector in model space
    float3 Shadow : TEXCOORD0;
};

// Structure output by the vertex program
struct Output {
    float4 Position : POSITION; // Position in clip space
    float4 DiffuseLight : COLOR0; // Diffuse light
    float4 SpecularLight : COLOR1; // Specular light
};

// Program executed for every vertex
Output main(PerVertexInput IN, // Per-vertex input
            uniform float4x4 ModelViewProj, // Transform matrix from model space to clip space
            uniform float3 LightPos, // Light position in model space
            uniform float3 EyePos, // Eye position in model space
            uniform float4 Kd, // Diffuse coefficient
            uniform float4 Ks, // Specular coefficient
            uniform float  Shininess, // Specular exponent
            )
{
    Output OUT;
    float4 diffuse,specular;
    float3 L,V,T;
    float  sinD;
    float  sinTL,sinTV;
    float  dotTL,dotTV;
    float4 P;

    P = IN.Position;

    // Transform the position from model space to clip space
    OUT.Position = mul(ModelViewProj, P);

    // Compute the light vector:
    // Normalized vector from vertex to light position
    L = normalize(LightPos - P.xyz);
    // Compute the eye vector:
    // Normalized vector from vertex to eye position
    V = normalize(EyePos - P.xyz);

    // Compute the diffuse light:
    sinD = dot(L,IN.Normal);

    diffuse = Kd;
    diffuse.xyz *= sqrt ( 1 - sinD * sinD) ;
    diffuse.xyz *= (IN.Shadow.x);

    OUT.DiffuseLight = diffuse;

    dotTL = dot(IN.Normal,L);
    dotTV = dot(IN.Normal,V);
    sinTL = sqrt ( 1- dotTL * dotTL);
    sinTV = sqrt ( 1- dotTV * dotTV);

    specular.xyz = Ks.xyz * pow (dotTL * dotTV + sinTL * sinTV, Shininess);
    specular.xyz *= (IN.Shadow.x);
    OUT.SpecularLight.xyz = specular.xyz;

    return OUT;
}
```

Figure 20. Hair shaded in realtime



7.1 Further Readings

For more details on the CG compiler, refer to CG manual that can be downloaded from http://www.nvidia.com

A Basic Hair/Fur Pipeline

Armin Bruderlin*
Sony Pictures Imageworks
armin@imageworks.com

Abstract

We have implemented a practical pipeline for the digital creation of hair and fur which has been used in productions like “Stuart Little”, “Stuart Little 2”, “Hollowman”, “Harry Potter”, and “The ChubbChubbs”. Our approach combines geometrical, animation and rendering techniques in order to provide a flexible, robust and efficient method to generate realistic looking animal fur or human hair. Rather than modeling tens of thousands of individual hairs, we define a smaller number of “control” hairs from which the final dense fur coat or hair is generated. The main calculations are divided up into determining the static features (instancing) and the animated features (interpolating) of each final hair. We have been able to achieve convincing looks of dry and wet fur coats as well as different styles of human hair. The approach also produces seamless hair/fur across the underlying NURBS skin patches.

This paper describes the original fur pipeline implemented for the first “Stuart Little” movie. We have since optimized, extended and adjusted our approach in several ways, mainly to facilitate the creation of human hair. Some of these extension are briefly addressed at the end of this paper and in the presentation.

Keywords: Computer animation, hair, fur, natural phenomena, animals.

1 Introduction

One of the many challenges in modeling, animating and rendering believable mammals in computer graphics has been to produce realistic-looking fur. A real fur coat is made up of hundreds of thousand of individual, cylindrical hairs covering the skin, and fulfills vital functions such as protection against cold and predators. Between animals as well as across the body of individual animals, the look and structure of these hairs vary greatly with respect to length, thickness, shape, color, orientation and under/overcoat composition. Fur is closely related to human hair, except that the latter is often longer and restricted to certain regions of the body.

Generating convincing hair/fur in computer graphics requires dedicated solutions to a number of problems. First, it is infeasible to individually model and animate all of the huge number of hairs of a fur coat. Real hair also interacts with light in many intricate ways, and due to simplified models of hairs, special shading methods are necessary to account for effects like reflection, opacity, self-shadowing or radiosity. Other problems which can arise are aliasing of hairs and collision of hairs, both between neighboring hairs and between hairs and the underlying surface. Finally, hair is not static, but moves and breaks up as a result of the motion of the underlying skin and muscles, as well as due to external influences, such as wind and water.

We have developed a practical hair/fur pipeline to address some of the above problems. Our goals have been to provide a tool which generates *realistic*, life-like fur, which is *flexible* (easy to modify and add new effects), *robust* (reliable to push lots of frames through), *efficient* (as inexpensive as possible, in time and storage), and *easy to use* by animators.

With these goals in mind, we take a brief look at other approaches to this problem in section 2. Our basic hair/fur pipeline is discussed in section 3 along with hair modeling, instancing and interpolating. Section 4 addresses some special effects we implemented into the current pipeline, such as clumping and special shaders we have developed for hair. In section 5, we discuss some improvements we have made since the first implementation. Finally, section 6 concludes our approach.

2 Related Work

Approaches to generating hair and fur can traditionally be divided into two basic categories: modeling [2, 3, 5, 7, 12, 13, 14, 15, 20] and rendering techniques [8, 11, 16, 17]. In the former case, individual hair primitives are geometrically defined, which can produce good close-up hair looks, but requires enormous computational and storage expenses. In the latter case, the geometry of hairs is “faked” through special rendering techniques to create the illusion of fur. These approaches address the anisotropic surface characteristics of a fur coat, and can result in convincing distant hair or fur shots, but lack close-up hair detail, and don’t provide a means for animating hairs. Because of the particular geometry of hairs, most modeling approaches also address some of the special shading requirements indicated in the previous section. The most recent approach for human hair [12] solves some of these issues, and introduces some effective multi-level hair styling tools.

There are also a number of proprietary solutions to generating hair and fur used for effects in motion pictures, but little detail has been published. DreamQuest Images is said to have developed fur software for “Mighty Joe Young”, Pixar has produced techniques for Hannah in “Toy Story” and Geri in the animation short “Geri’s Game”. Most recently, they have developed a convincing system used in “Monsters, Inc.” [9]. As described in the next section, we have incorporated some of Pixar’s available technology into our fur pipeline, namely RenderMan’s[1] 3.7 Curve and Procedural primitives. Rhythm & Hues has developed proprietary software for “Mouse Hunt”, the Polar Bear commercials and “Babe” to generate fur. Industrial Light & Magic (ILM) has applied a published “fake fur” method [8] for “101 Dalmations”. Their probabilistic rendering algorithm with special illumination and opacity functions generates believable short-fur distance shots. ILM also developed software for the movie “Jumanji”. Santa Barbara Studios implemented their own hair pipeline for “American Werwolf in Paris” and was able to produce striking dry and wet hair effects. Finally, Digital Domain developed special hair shaders for the mice men in “The Island of Dr. Moreau”.

Most 3D animation systems today also include some hair/fur module, such as Alias|Wavefront’s Maya[2] and 3D Studio MAX from Discreet. Finally, Joe Alter [1] has been implementing a comprehensive and powerful system for hair modeling, animation and rendering.

[1]RenderMan is a registered trademark of Pixar.
[2] Based on the system first developed at Santa Barbara Studios.

Despite all these successful hair/fur generation techniques, we decided to implement our own method for several reasons. First, back in 1997 when we started production on the first Stuart Little movie, there were no real off-the-shelf fur packages available. Second, by taking our own approach, we have been able to more easily integrate the software into our production pipeline to generate the required hair/fur effects, and to achieve the goals mentioned in the previous section. Also, some of the existing research techniques and commercially available packages didn't seem to be flexible, powerful and robust enough for our purposes. For example, the Maya fur tool lacks the necessary controls for long hair, hair clumping, and is tied into the Maya renderer (while we have been using RenderMan in our production pipelines). Of course, we have incorporated several ideas from existing techniques, such as texture maps for static hair features, or using a combination of modeling and rendering procedures.

3 Basic Approach

Our hair/fur implementation is characterized by the following key-features, which are an integral part of the pipeline (Figure 1):

- all the hairs are "grown" on NURBS surface patches, since our characters are modeled this way. For example, for the mouse in our examples, we have approximately 120 patches for head, chest, back, arms, hands and tail with fur on them. Our fur algorithm therefore needs to incorporate continuity across NURBS patch boundaries (see section 3.2).
- we don't model each individual hair, but only some "control" hairs, from which the final fur coat is determined (on the order of 1 control hair for every few hundred final hairs; see section 3.2). This simplifies combing and animation of hairs, and reduced file sizes and computation times throughout the pipeline.
- the fur calculations are divided into a static (once per shot; see instancing, section 3.3) and an animated part (once per frame; see interpolating. section 3.4). This provides efficiency: static hair features are pre-calculated after modeling, and frame-changing features are determined at render time.
- delay fur calculations as much as possible throughout the modeling-animation-rendering pipeline. This way, we minimize the amount which needs to be specified and carried along at each step.
- use of texture maps to specify static hair features (e.g. length, density, width of hairs; see section 3.3). This improves flexibility and efficiency, since maps provide fine-control and can be swapped for a different look without complete recalculation.
- all of the hair files (control hairs, instanced hairs, texture maps) throughout the pipeline are NURBS patch-based, e.g. there is one hair length map for each patch which has hair on it. This provides consistency and ease of adding new features.
- use of the RenderMan [19] 3.7 RiCurves primitives and RiProcedural dynamic shared object (DSO) feature for rendering hairs (see section 3.4). Since each NURBS patch and all the hairs on it are procedurally generated at render time, the file sizes of the RenderMan *.rib* files are reduced, and we achieve greater flexibility in tweaking hair parameters. Also, the RiCurves procedure renders the hairs efficiently, and our custom hair shaders (see section 4.4) provide subtle hair-effects.
- effects like clumping (static and animated), breaking of hair (this feature is not discussed here; please refer to [4] for more detail), or wind effects are seamlessly integrated into the pipeline and can be turned on and off without recalculation of all of the hair files (see section 4.2 and 4.3).

3.1 Pipeline

Figure 1 illustrates the major components of our fur pipeline, which is activated once our models (e.g. the mouse) are fully defined, physiqued and animated in the 3D animation system. Please note that for clarity, this is a simplified picture of our complete pipeline, which doesn't show some other fur-specific software such as file translators and versioning, as well as our proprietary front-end system to RenderMan.

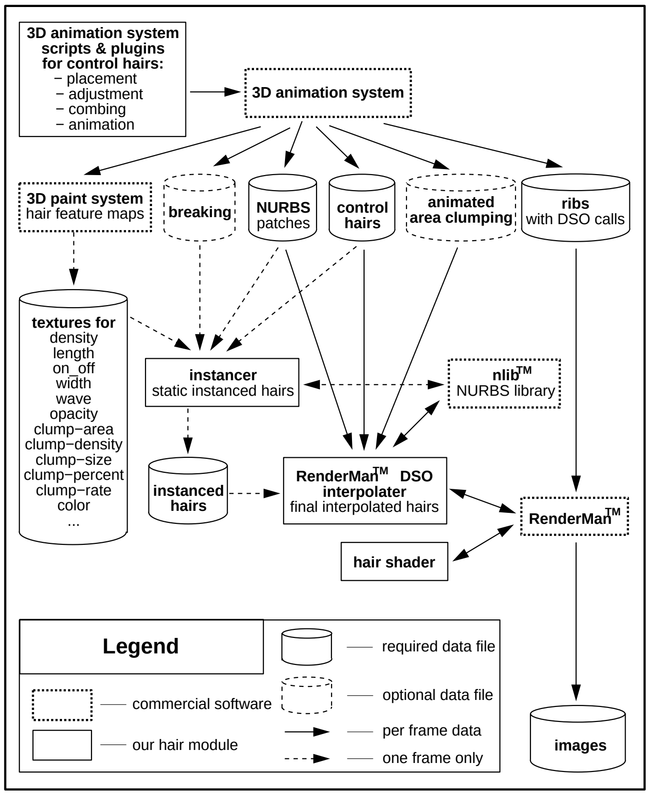


Figure 1: Hair/Fur pipeline.

The first step is the definition of the *control* hairs, which is done through scripts and plugins to the 3D animation system (see section3.2 below). At the same time, the model is exported into the 3D paint system, where the static feature maps for the hairs are painted directly onto the model. Some of these maps are described in more detail in section 3.3. Three primary sets of files are then output from the 3D animation system: the NURBS patches of the model, the control hairs and the RenderMan *.rib* files. The NURBS patch and control hair files are used by the *instancer*, which calculates the static (frame-independent) features of each final hair (see section 3.3), and produces a set of instanced hair files. These instanced hair files, together with the NURBS patch and control hair files, are read in by the *interpolater*, which calculates the final hairs for each frame of an animation (see section 3.4). This interpolater is implemented as a DSO to RenderMan and called at render-time for each frame. The generation and processing of the optional files and feature maps for hair *clumping* is described in sections 4.2. Special hair shaders have also been implemented to create a realistic hair look (section 4.4).

3.2 Control Hairs

Control hairs are the modeled and animated hairs which determine the geometry of the final fur, i.e. all the rendered hairs. They are created as NURBS curves in the 3D animation system, attached to follicles (points-on-surfaces) of the underlying NURBS patches. The ratio of the number of control hairs to “real” finals hairs is usually 1 to a few hundred or even thousands (e.g. about 1 control hair per 800 hairs on the head of the mouse). If the control hairs vary little in their orientation, much fewer are necessary. As a minimum, we need 4 control hairs in each corner of a NURBS patch. We have developed scripts and plugins for conveniently *placing*, *adjusting*, *combing* and *animating* control hairs, which is described in the following paragraphs.

Placement of control hairs is possible in several different ways. The simplest algorithm places x hairs in the u and y hairs in the v-direction of the selected NURBS patches (x and y are specified by the user), equally spaced in parametric space. A slightly more expensive option is to place x and y hairs equally by arc-length, which results in a more uniform distribution across a patch. This algorithm, however, does not achieve a balanced distribution of control hairs across patches of different size: x and y hairs are placed on all selected patches, large or small. We therefore developed a second method to place control hairs, which takes the area of a NURBS patch into account to determine x and y for each patch (the user now specifies z hairs per square area). This is illustrated in Figure 2, where z was set to 10, which resulted, for instance, in 4 by 3 hairs for the left-mid-face patch (dark-gray) to achieve a balanced placement of hairs across all of the 34 face patches. The hairs were generated with the equal arc-length option. Each control hair, by default, is defined by 4 control points (see insert in Figure 2), and points along the surface normal. Control hairs can also be placed individually or along curves-on-surfaces for more fine control. For instance, we usually place extra control hairs along the sharp rim of the ears to ensure proper alignment of the final fur in this area.

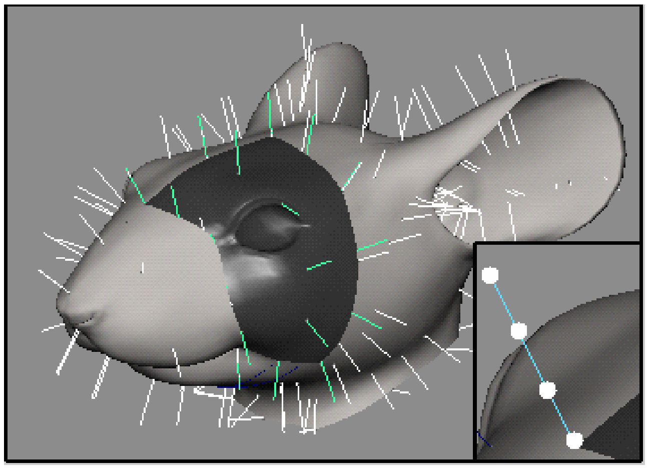


Figure 2: Control hairs on NURBS patches.

Since control hairs are placed per NURBS patch, there is a problem: boundary control hairs (hairs which lie on either U_MIN, U_MAX, V_MIN or V_MAX) on one patch might not line up with the ones on a neighboring patch, which can lead to discontinuities of the final hairs along patch boundaries. To solve this, we implemented a control hair adjustment routine, which consists of two steps: a seam generator, and an alignment/duplication for boundary control hairs. The seam generator[3] takes all the NURBS patches and constructs all seams between these patches (either whole-edge, T-junctions, or corners). The alignment/duplication module then traverses all the boundary control hairs of a patch, and if there is a corresponding control hair on the neighboring patch (within a small delta), the neighboring hair is aligned (snapped together) with the first hair. If the neighboring patch does not have a corresponding control hair, then one gets inserted (actually instanced from the existing hair) first before alignment. For example, the top-left corner control hair for the left-mid-face patch in Figure 2 is aligned with its three neighboring corner control hairs (four patches share this corner). No insertion is necessary here since the neighboring patches already have a control hair in the common corner (as mentioned above, there is always a control hair placed at each corner of a patch). The control hair to the right of that corner hair on patch left-mid-face would first be duplicated at the same location on the neighboring patch left-nose and then aligned, since there is no corresponding control hair yet on the latter patch.

The result of combing the control hairs in Figure 2 is illustrated in Figure 3. Our combing algorithm works on selected control hairs, and allows for specifications such as global combing direction curves (white curve above the head), degree of bending and curvature of the hair, and region of influence with a falloff for each direction curve. Combing also provides a simple hair/surface collision mode, in which control hairs which intersect with the underlying NURBS surface are pushed back up. This iterative algorithm uses a intersection check, where the line segments are defined by two successive control vertices of the NURBS curve control hair; if a control vertex *c* goes below the surface, the hair is rotated back towards the surface normal from the previous non-intersecting vertex just enough for *c* to clear the surface.

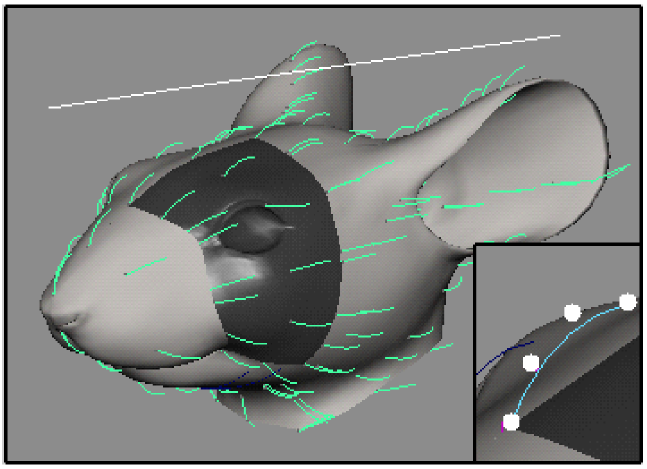


Figure 3: Control hairs after combing.

Placement and adjustment of control hairs are usually only performed once per model and are reused in each shot, whereas some shots might require different combing to produce, for instance, a groomed look versus a slightly messy look of the fur. The combing module can also be used to animate control hairs by key-framing the combing parameters, and executing the combing node at each frame during playback. We have also experimented with a different technique for animating control hairs in the 3D animation system, by turning each control vertex of a hair into a particle, dynamic effects like gravity and external forces can be applied. However, this approach is computationally expensive,

[3] This module is also utilized in our routine for stitching NURBS surfaces.

and more difficult to control. Also, additional code is required to enforce the length of each hair.

After placing, adjusting, combing and possibly animating the control hairs, they are written out into control hair files. We write out the (u,v) position of each control hair on a patch, plus the position of each control vertex in [normal,du,cross(normal,du)] coordinates as shown in Figure 4; this way, we only need one control hair file per patch for non-animated control hairs (even if the underlying NURBS patch is animated/deformed). For animated control hairs, one file per patch per frame is written out.

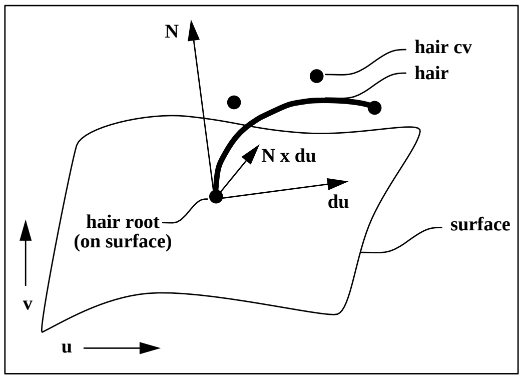


Figure 4: Hair coordinate system.

3.3 Instancer

The instancer calculates the static, frame-independent features of the final fur, based on the control hairs and the hair feature maps. It reads in one frame of the NURBS patch and the control hair file, and writes out a set of instanced hair files, one per NURBS patch. Although we use all the hair feature maps listed in Figure 1 in production, they are optional to provide fine control. For instance, one required input to the instancer is the length of the final hairs[4], which is taken absolute if no hair length feature map for the current patch is found, or it is multiplied for a particular final hair at (u,v) on the patch with the corresponding normalized (s,t) value in the feature map.

The three most important calculations of the instancer are: the (u,v) *positions* of each final hair from the density value and optional density map; the *enclosing three control hairs* for each final hair; and the *weights* of each of these three control hairs with respect to the final hair. How we compute these values -- which are used later at render-time by the interpolater (see section 3.4) to determine the orientation of each final hair -- is described next.

The (u,v) positions of each final hair are computed from an overall (global to all patches) density input value (number of hairs per square unit area) and an optional density feature map per patch. The two main problems which needed to be resolved were to make the number of final hairs independent of the feature map resolution and independent of the NURBS patch size to provide seamless density across patches of different scale. As a solution, we implemented the following algorithm: assume that the specified density value (*dmax*) is 10 hairs/unit square area, and the density feature map resolution is 128x128 pixels and 8-bit/pixels (0-255 values).The first step now is to calculate 128x128 equally-spaced (in arc-length) points in (u,v) on the patch[5], and assign a density value to each of these points, by multiplying *dmax* with the corresponding normalized (s,t) value in the density map. We then traverse these equally spaced points as illustrated in Figure 5: assume that the looked-up density values of the current four neighboring points are 4, 5, 6 and 6 hairs, respectively, per unit square area. We approximate the area in (u,v) between these neighboring points by the area of the two triangles a1 and a2, and average the number of hairs/square unit area for each triangle from the values at its vertices, which results in (4+5+6)/3=5 and (4+6+6)/3=5.333 hairs/square unit area for the top left and bottom right triangle, respectively. If we assume a value of 0.4 for area a1 and 0.3 for area a2, we now have 0.4*5+0.3*5.333=3.5999 total hairs to place in subpatch (ui,vi), (ui,vi+1), (ui+1,vi+1), (ui+1, vi). Since we can't place fractional hairs, we place either 3 or 4 hairs depending whether a uniformly generated random number in [0,1] is bigger or smaller than the fractional part (0.5999). We then place the 3 or 4 hairs randomly in u [ui,ui+1] and randomly in v [vi,vi+1], and proceed to the next four equally spaced points.

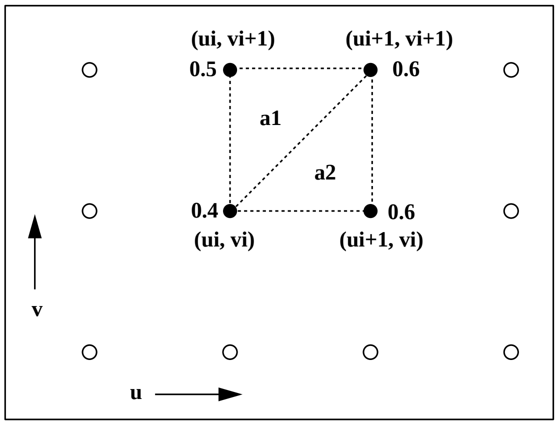


Figure 5: Equally-spaced points on NURBS patch.

Having calculated the (u,v) positions, we now determined the three enclosing control hairs for each final hair. For this purpose, a 2-dimensional Delaunay triangulation [10, 18] is constructed of the (u,v) positions of the control hairs for each patch. This triangulation was chosen because it creates "well-proportioned" triangles, by minimizing the circumcircle and maximizing the minimal angles of the triangles. Once the Delaunay triangulation is constructed, we look up the triangle which each final hair falls into, and assign the indices (in the control hair file) of the three control hairs which form this triangle to that hair.

Finally, we calculate the weights (w1,w2,w3) which each of the three control hairs (c1,c2,c3) has on the final hair (h). This is done using barycentrc coordinates [10], and is illustrated in Figure 6, where "A" is the area of triangle (c1,c2,c3). These weights are used for interpolating the final hair from the control hairs (next section).

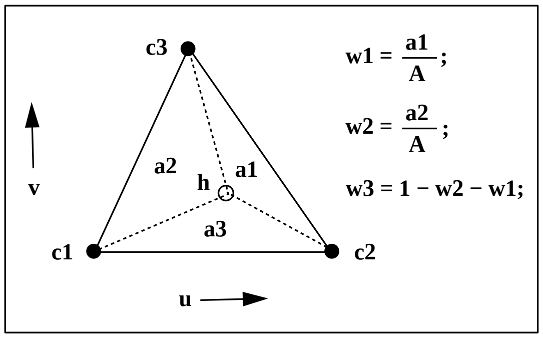


Figure 6: Calculation of control hair weights.

[4] There is also an optional length noise parameter to produce random variations on each hair length.

[5] We used nlib, a NURBS library from GeomWare, for all our NURBS calculations.

Figure 7 shows the result of applying a density and a length map (left) to a single patch after rendering (see next section) the hair (right). In this case, the same map was used both for density and length, which produces denser and longer hair in lighter areas of the map.The hair is also brighter where it is denser and longer because it reflects more light.

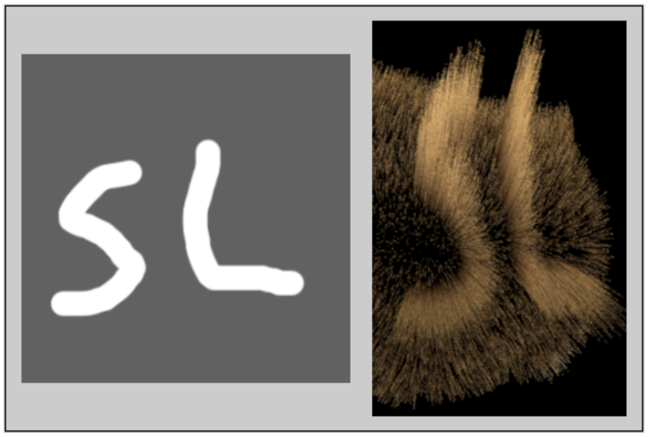


Figure 7: Density/length maps and furry patch.

Besides the density and length[6] feature maps, we also employ maps for the width of the hairs, the waviness, opacity, clumping (see section4.2), and hair color. We also have on/off maps which, if present, get combined in the instancer with the density maps to quickly turn off areas on patches which don't need hairs; for instance, areas of the body which are covered by clothes in certain shots. Values of all these maps are assigned to each final hair, and written out into the instanced hairs files together with the information calculated above. Again, the reason for pre-calculating all the static, frame-independent features before rendering is efficiency.

3.4 Interpolator

The interpolater reads in the instanced, control hair and NURBS patch files, and is executed at each frame during rendering to calculate the geometry of each final hair and add attributes to the geometry which are used in the shading process. We implemented this module as a DSO to RenderMan, which has several advantages: it reduced the size of the RenderMan *.rib* files, since there is only one call per NURBS patch with all its hair on it. Also, the final hair and patch geometry is procedurally generated at render time (RiCurves and RiNuPatch) on the fly, which makes the pipeline more efficient and allows for last-minute tweaking of hair parameters (e.g. add randomness, or change the waviness, width, etc., of the hairs[7]).
Recall that the control hair files contain the (u,v) position of each control hair plus its control vertices in [normal,du,cross(normal,du)] space. For each final hair (h), the interpolater now (see Figure 8):

1. converts the corresponding three control hairs (c1,c2,c3) into patch space;
2. calculates control hair vectors, e.g. (**v11,v12,v13**), between control vertices;
3. normalizes control hair vectors; e.g. (**nv11,nv12,nv13**);
4. interpolates corresponding control hair vectors of the three control hairs multiplied by the three weights;
   e.g. **iv1**= **nv11***w1+**nv21***w2+**nv31***w3;
5. scales resulting vectors to final hair length (**siv1,siv2,siv3**);
6. calculates control vertices of final hair from scaled resulting vectors;
7. issues an RiCurves call for each hair[8].

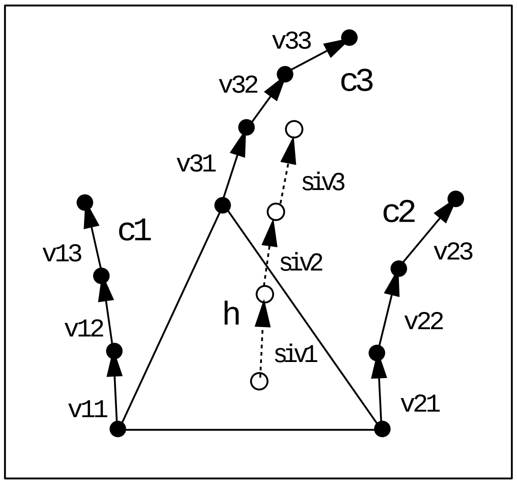


Figure 8: Interpolation of final hair (h) from control hairs (c1,c2,c3).

As an option, each hair can also be individually motion-blurred by issuing two RiCurves calls per final hair. Figure 9 shows the rendered fur, generated from the combed control hairs and model of Figure 3[9] and consisting of about 450,000 individual hairs[10].

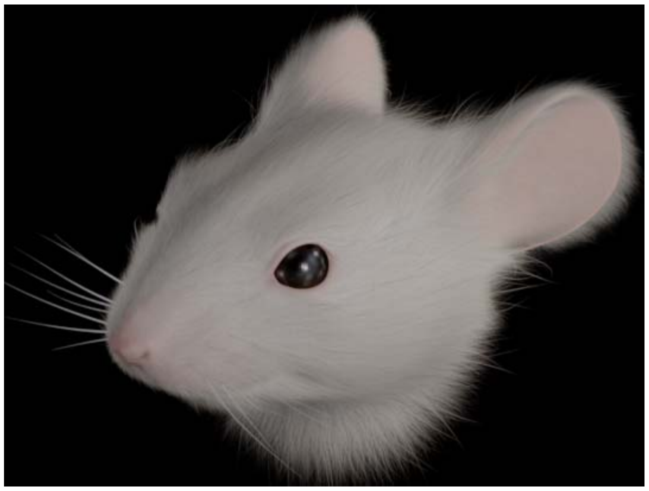


Figure 9: Snapshot of a rendered sequence.

[6] As not previously mentioned, we also import the length maps into the 3D animation system to draw the control hairs at their proper length.
[7] The interpolator also randomly turns a small percentage of hairs into "guard" hairs (longer and thicker).

[8] We have improved efficiency by packing several hairs (on the order of hundreds) into one RiCurves call.
[9] The whiskers are modeled, animated and rendered through a separate branch of the pipeline.
[10] In this model, about 30% of all the hairs are located on the ears, which have very a short, dense coat.

4 Special Effects

4.1 UnderCoat/Overcoat

The coat of most furred animals is composed of a fuzzier, thinner and shorter layer of hairs called undercoat, plus an overcoat of longer and thicker hairs. We can simulate this phenomenon in our pipeline as a two-(or multiple-)pass process, whereby the instancer is executed more than once with different feature maps, producing a different set of instanced hair files at each pass. The interpolater then processes all the layers (i.e. sets of instanced hair files) at render-time. A simple example of combining an undercoat with an overcoat this way is shown in Figure 10. We have also applied this multi-pass feature for human hair when both long and short hair are present and can be treated independently.

4.2 Hair Clumping

Clumping of hairs can occur when the fur gets wet due to the surface tension or cohesion of water. The effect is that the tips of neighboring hairs (a bunch of hairs) tend to gravitate towards the same point, creating a kind of cone-shaped "super-hair", or circular clump. We have implemented two kinds of clumping techniques: *static area clumping* and *animated area clumping*. The former generates hair clumps in fixed predefined areas on the model, whereas the latter allows for clumping areas to move on the model. In both cases, we provide parameters which can be animated to achieve various degrees of dry-to-wet fur looks. We have also used this approach for dry fur clumping by breaking up the combed look of dry fur, as well as for human hair.

4.2.1 Static Area Clumping

Static area clumping is processed by the instancer to define clumps, and the interpolater to differently interpolate hairs which are members of clumps. We are also developing a special shader for wet hair (see section 4.4).
We call the center hair of each clump the *clump-center hair*, and all the other member hairs of that clump, which are attracted to this clump-center hair, *clump hairs*. There are four required clumping input parameters to the instancer, which can be supplemented by corresponding feature maps to provide local control: *clump-density, clump-size, clump-percent* and *clump-rate*. Similar to the hair-density parameter, clump-density specifies how many clumps should be generated per square area. The instancer translates this density into an actual number of clumps, i.e. clump-center hairs, depending on the size of each NURBS patch. As a result, some of the instanced, final hairs are turned into clump-center hairs. Clump-size defines the area of a clump in world space. Clump-density takes priority over clump-size, in that if there are many clumps, such that most of them overlap, the size can not be maintained, since a clump hair can only be a member of one clump. If both clump-density and size are small, many final hairs between the clumps will not be clumped. To determine clump membership of each final hair, the instancer first converts the specified clump-size into a u-radius and v-radius component in parametric patch space at each clump-center hair location. It then checks for each final hair, whether it falls within the radii of a clump-center hair; if so, that clump-center hair's index is stored with the final hair; if not, the final hair is not part of any clump, i.e. is not a clump hair, and thus will be interpolated "normally" by the interpolater as described in section 3.4. Similar to hair length, the instancer also has an optional clump-size noise parameter to produce random variations in the size of the clumps. Also, a clump-area feature map can be provided to limit clumping to specified areas on NURBS patches rather than the whole model.

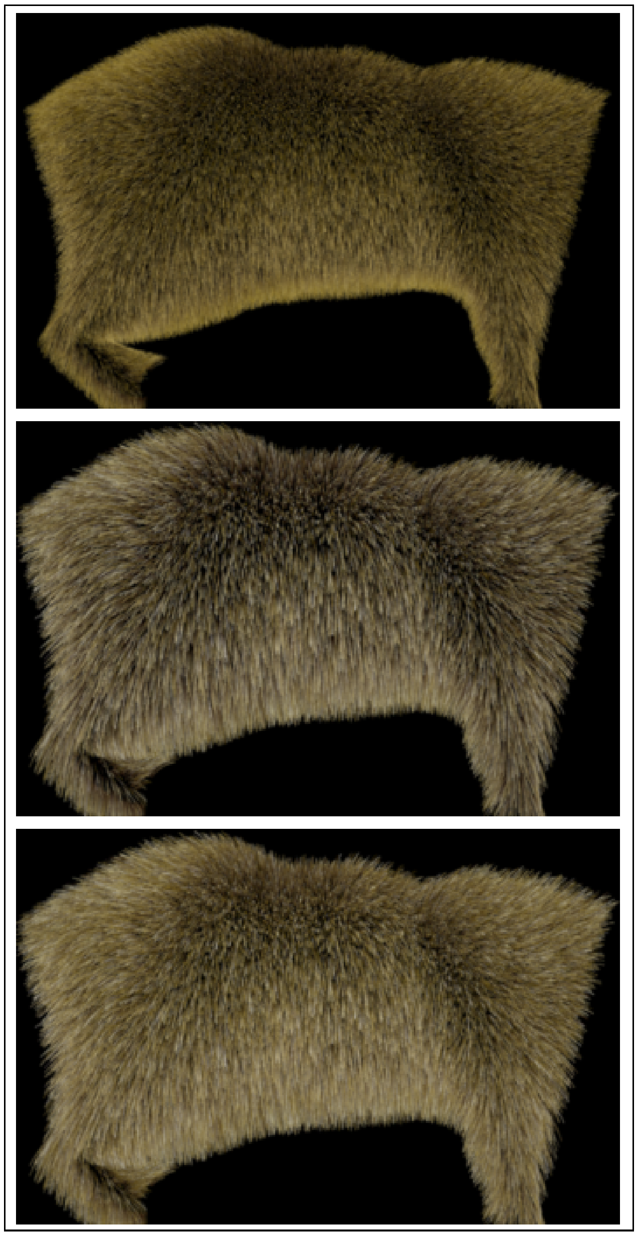


Figure 10: Undercoat (top) + overcoat (middle) = final coat (bottom).

Finally, the instancer also assigns a clump-percent and clump-rate value to each clump hair. The values for both range between [0,1], and are used by the interpolater as illustrated in Figure 11 and described below. Clump-percent specifies the degree of clumping for a clump hair: a value of zero means that the hair is not clumped at all, i.e. it is interpolated like a "dry" hair; a value of one means that the hair is fully attracted to its clump-center hair, i.e. the tip of this hair (its distant control vertex) is in the same location as the tip of the clump-center hair. Clump-rate defines how tightly a clump hair clumps with its clump-center hair. A value of zero means that the clump hair is linearly increasingly attracted to its clump-center hair, from the root to the tip; a value closer to one means that the hair's control vertices closer to the root are proportionally more attracted to their corresponding clump-center hair vertices than those closer to the tip, which results in tighter clumps.

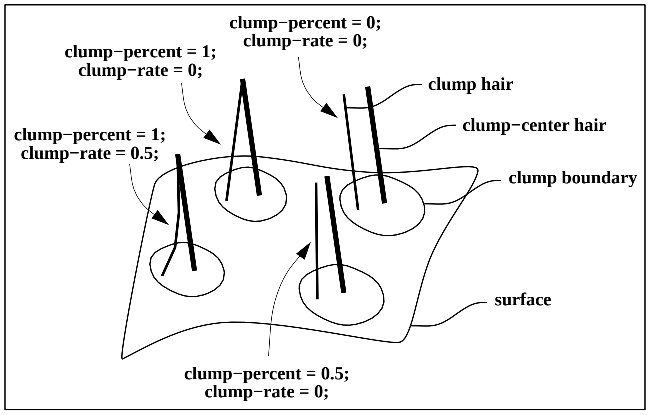


Figure 11: Clump percent and clump rate.

The clumping information is written out into the instanced hair files, and processed by the interpolater. In a first pass, all the final hairs are interpolated normally as described in section 3.4. In a second pass, the control vertices of each clump hair (except the root vertex) are displaced towards the their corresponding clump-center hair vertices from their position after the first pass as follows[11], where the default value for numberOfCVs is 3 (4 minus the root vertex), and the index for the current control vertex, i, ranges from 1-3:

fract = i / numberOfCVs;
delta = clumpPercent (fract + clumpRate (1 - fract));
clumpHairCV[i] = **clumpHairCV**[i] +
delta (**clumpCenterHairCV**[i] - **clumpHairCV**[i]);

The interpolater also accepts a global clump-off switch, which allows the generation of non-clumped fur even if the instanced hair files contain clumping information. Also, both clump-percent and clump-rate can be overwritten at render-time, and simple animation expressions for both can be specified. This is illustrated in Figure 12, which shows three frames from an animated clump-percent and clump-rate sequence for the same fur as in Figure 9 (where clump percent and rate are both zero). In the top image, clump percent is 0.7 and clump rate is 0, which results in a slightly wet look. For the middle, clump percent is 1.0 and clump rate is 0.3, which results in a wet look. In the bottom image, clump percent and rate are both 1.0, which produces a very wet look.

4.2.2 Animated Area Clumping

Animated clumping areas are desirable if we want to simulate spouts of water or raindrops hitting the fur and making it increasingly wet. We have developed tools within the 3D animation system to generate these animated clumping areas, and adjusted our interpolater to process this frame-to-frame information from the animated area clumping files.

[11] It is noted that both rate and percent operations change the lengths of clump hairs, but we have not observed any visual artifacts even if these values are animated.

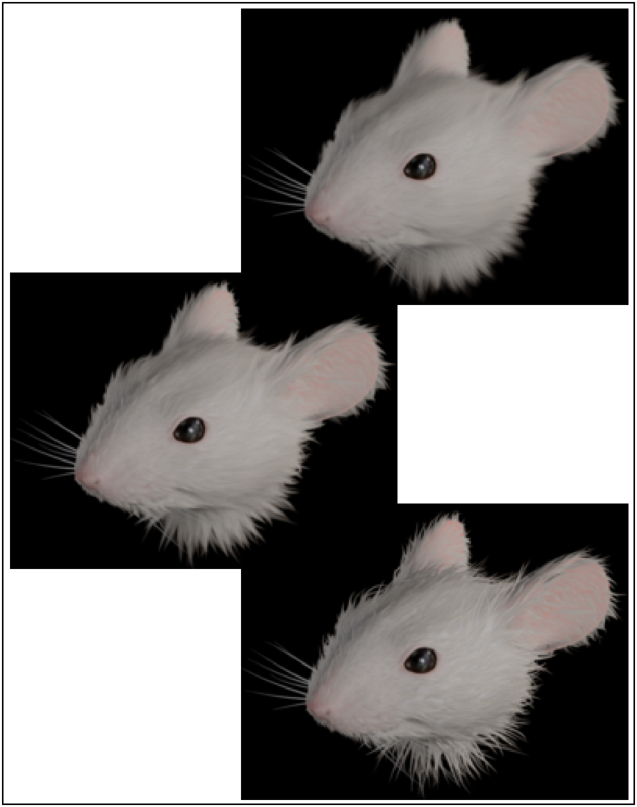


Figure 12: Three frames of a static clumping sequence.

Animated clumping areas in the 3D animation system are defined through particles hitting NURBS patches. These particles originate from one or more emitters, whose attributes determine, for instance, the rate and spread of the particles. Once a particle hits a NURBS patch, a circular clumping area is created on the patch at that (u,v) location, with clump-percent, clump-rate and radius determined by a creation expression. Similar to static clump-size, this radius is converted into a corresponding u-radius and v-radius. Runtime expressions then define clump percent and rate, to determine how quickly and how much the fur "gets" wet. Figure 13 shows a snapshot of animated clumping areas, where each area is visualized as a cone (the most recent cone is highlighted). In this case, one particle emitter is used and has been animated to follow the curve above the head. Once the animation and parameters are satisfactory, the areas are written out into animated area clumping files, one per patch per frame.

In order for animated area clumping to work, the instancer is executed just like in the static area clumping case, with clump-area turned off, so that the whole model is clumped. The interpolater then restricts clumping to the animated clumping areas at each frame, so that final hairs of clumps outside clumping areas are normally interpolated as "dry" hairs. Also, the values for clump-percent and clump-rate for each clump within a clumping area are replaced with the corresponding values from the animated area clumping files at each frame. Clumping areas are usually much bigger than a clump, thus containing several individual clumps. To determine whether a clump falls within a clumping area, the interpolater checks at each frame whether the (u,v) distance between the clump-center hair of the clump and the center of the clumping area is within the (u,v) radii of the clumping area. For clumps in overlapping clumping areas, the values for clump percent and rate are added to generate wetter fur. The same frame as in the animated clumping area specification sequence of Figure 13 is shown rendered in Figure 14.

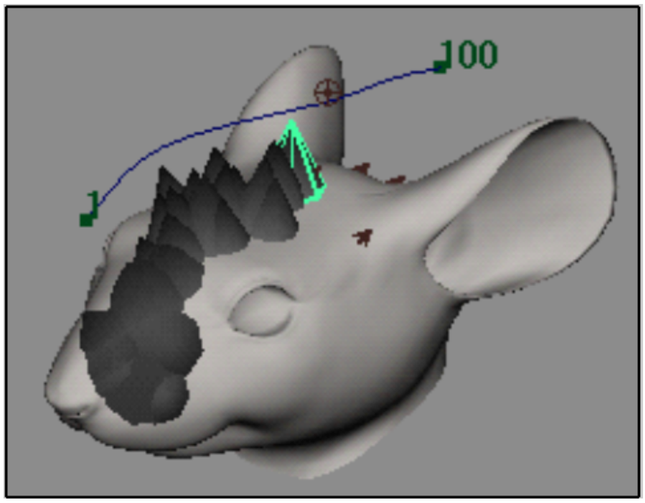


Figure 13: Animated clumping areas.

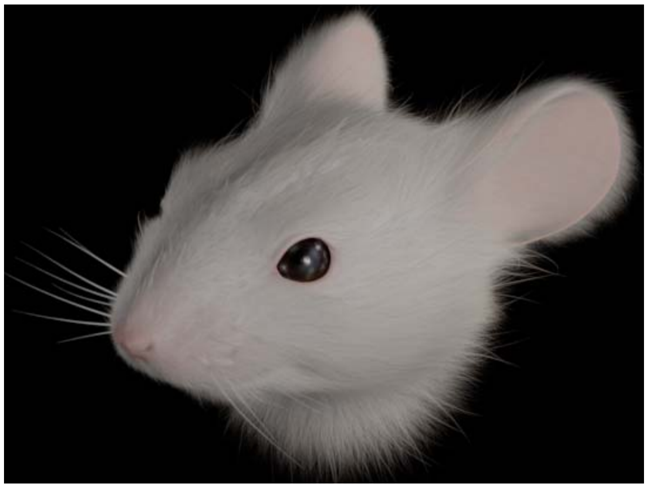


Figure 14: Frame of an animated area clumping sequence.

One problem with animated clumping areas has not been addressed yet: they can straddle NURBS patch boundaries as shown in Figure 15, which shows a top view of a clumping area (light-colored cone): the center of this clumping area (black dot) is on patch P1, but the area straddles onto patch P2 and P3 (across a T-junction; seams are shown as bold lines). Since our clumping areas are written out from the 3D animation system per patch (the patch which contains the (u.v) of the center, i.e. the position where the particle hit), portions of a clumping area straddling neighboring patches are ignored. This would lead to discontinuities in clumping of the final fur, as illustrated in Figure 16, top.

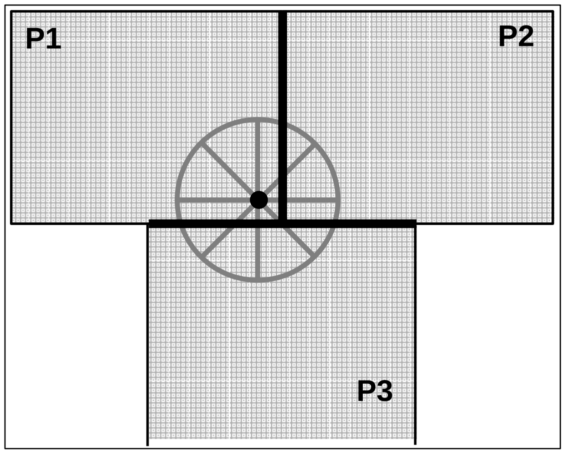


Figure 15: Clumping area straddling a T-junction.

The solution we implemented again utilizes our NURBS seam and topology generator module: whenever a new particle hits a surface, we check whether the (u,v) radii exceed the boundaries of the patch; if so, we calculate the (u,v) center and new (u,v) radii of this clumping area with respect to the affected neighboring patches[12], and write this information into animated area files for those patches. Figure 16, bottom, shows the result of this algorithm after the interpolater has automatically processed the two additional files of the affected patches.

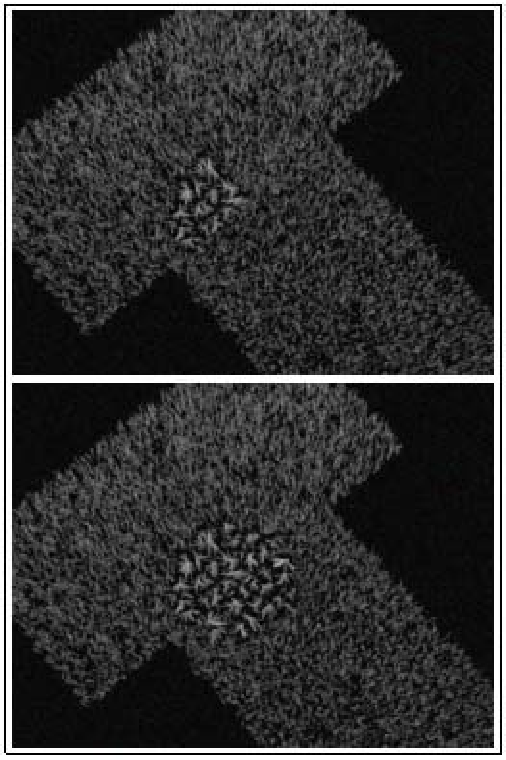


Figure 16: Straddling clumping area: fur without (top) and with (bottom) special treatment.

[12] The fact that the (u,v) clumping area centers of the affected patches actually lie outside their (u,v) bounds is irrelevant, since the interpolater does not evaluate the patches, but computes distances in (u,v) space.

4.3 Wind Effects

For wind effects, we apply a noise function with a user-specified wind amplitude, wind direction and frequency directly to the control points of a hair. These computations are applied at render time in the interpolator after step 6 in section 3.4.

4.4 Hair Shaders

In order to render large amounts of hair quickly and efficiently, the geometric model of each hair needs to be simplified. As previously mentioned, we use the RenderMan RiCurves primitive to render the final hairs. This primitive consists of many small micro-polygons falling along a specified curve with their normals pointing directly towards the camera. In comparison to long tubes, these primitives render extremely efficiently. However, we still need to solve the problem of how to properly shade these primitives as if they were thin hairs.

Kajiya and Kay [11] present an approach to this problem. Their approach is similar in concept to the idea of phases of the moon, where the percentage of the light hitting a hair segment depends on the light's orientation to that hair segment's direction. They also assume that each hair has a width approximately equal to or less then one pixel and then calculate the average amount of light reflecting off each segment of the hair. Based on this idea they develop a diffuse and a specular model for the hair. These models provide a great starting point for lighting the hair. However, in practice we have encountered several difficulties. The biggest problem is the model's tendency to produce darker hair as the tangents of the hairs point more towards the light. This results in dark hairs closer to the light and brighter hairs towards the edges of the model. If we also included radiosity calculations within our model, that is if we calculated light hitting the skin and adjacent hairs and bouncing off, the shading model would provide a good solution. However, this is extremely expensive to calculate. Another problem with this approach is that it does not present a very intuitive method for users to light the fur. For these reasons, we have implemented a more ad hoc approach.

In order to obtain a *shading* normal at the current point on the hair, we mix the surface normal vector at the base of the hair with the normal vector at the current point on the hair. The amount with which each of these vectors contributes to the mix is based on the angle between the tangent vector at the current point on the hair, and the surface normal vector at the base of the hair. The smaller this angle, the more the surface normal contributes to the shading normal. We then use a Lambertian model to calculated the intensity of the hair at that point using this shading normal. This has the benefit of allowing the user to light the underlying skin surface and then get very predictable results when fur is turned on. It also accounts for shading differences between individual hairs, and along the length of each hair.

Hair shadowing presents several problems with white fur[13]. Since we use shadow maps, incorporating the hair into the shadow maps generates several unwanted side-effects. One problem is that of dark streaking on brightly lit fur because of fur self-shadowing. Dark streaks look wrong on brightly light fur because normally light bounces off the skin and hair to prevent dark shadows on brightly lit fur. In order to get around this we shorten the hair in shadow map renders based on certain criteria. For example, the length and the density of the hair dictate the percentage to shorten the hair. This fixes the hair self shadowing, and still produces a broken up shadow on the terminator lines for lights falling on the fur.

[13] This color was chosen for Stuart Little at the outset.

Backlighting is achieved in a similar fashion, using a shadow map for each light located behind the furred object, and again shortening the hair on the basis of density and length in these shadow map renders. Our lighting model for hairs also allows each light to control its diffuse falloff angles. Thus, lights directly behind the furred object can wrap around the object. Using these lighting controls together with shadow maps we are able to achieve reasonable backlighting effects. Figure 17 illustrates such an effect, which is particularly observable on the silhouette hairs around the left ear.

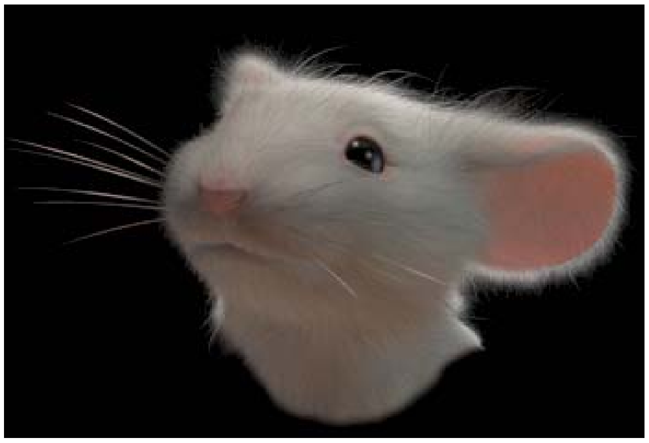


Figure 17: Backlighting Effect.

For the wet fur look, we change two aspects of the hair shading. First, we increase the amount of specular on the fur. Second, we account for clumping in the shading model. Geometrically, as explained earlier, we model fur in clumps to simulate what actually happens when fur gets wet. In the shading model, for each hair, we calculate for each light, what side of the clump the hair is on with respect to the light's position, and then either darken or brighten the hair based on this. Thus, hairs on the side of a clump facing the light are brighter then hairs on a clump farther from the light.

The shading model also takes into effect individual attributes for each hair, such as color, opacity, special treatment of guard hairs and ear hairs. In terms of fur look, white is probably one of most challenging colors to render. Specular highlights are less noticeable to the human eye on white fur. Also, white fur suffers from lack of detail in fur renders, which is usually provided by color variations between adjacent hairs. In fact, most of the detail in white fur comes from fur self-shadowing which is expensive to account for. We have achieved some self-shadowing effects with respect to dry and wet clumping of fur.

5 Newer Features

Ever since completing the first implementation of our hair/fur pipeline used in the production of "Stuart Little", we have been constantly improving, optimizing and adding new features to it. We even developed a new, more powerful hair and feather system for "Stuart Little 2" [6], which incorporated, among other things, an expression language for parameters, node based execution and blending between different solutions, additional hair and feather placement method based on particle repulsion, and automated control hair animation to prevent interpenetration with skin.

However, the original, more lightweight hair pipeline introduced here has survived. Besides bug fixes, most of the newly added features have been to support the production of (longer) human hair in recent productions. For instance, we have developed additional combing tools. Whereas the first “curve” combing tool (see section 3.2) was appropriate for reference combing of short hair/fur, we needed a better, direct manipulation tool for styling longer hair and for shot-specific animation of hair. Two ideas we implemented are a “chain” combing tool, where the user defines a skeletal chain over a control hair (with as many segments as the number of control points of the control hair) which can influence any number of other control hairs and can be directly manipulated and animated via forward and inverse kinematics. The other idea is a “paint” combing tool similar to the Maya fur approach, where the user can paint direction, bend and lay down angle with a brush. We have also employed the dynamic animation capabilities of the underlying animation system to define some of the motion of longer hair. An example is shown in Figure 18.

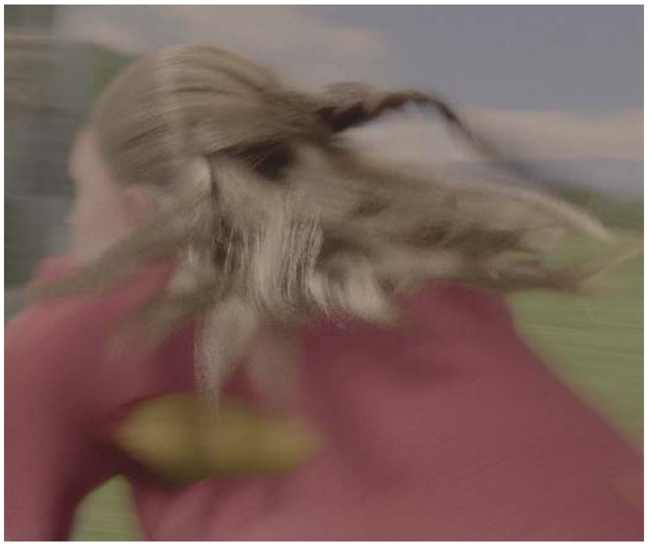


Figure 18: Long digital human hair in motion.

Another area in which we extended the original system has been clumping tools. The clumping method introduced in section 4.2 works well for fur where we do not need control over where exactly the clumps are. But for human hair we want to manually and interactively place and form clumps, perhaps combined with automatically generated clumps. Also, we need more fine control over the shape of clumps (besides percent and rate, we now provide additional parameters like falloff and taper), as well as introduce more “controlled” randomness to the final hair such as weave and splay. Figure 19 illustrates the result of applying our improved clumping tools to an imaginary character, and Figure 20 shows clumping and combing applied to match the hair of a real human being.

A last example of how we have been improving our hair/fur pipeline has to do with workflow. In the original system we combed control hairs, but it wasn’t until we rendered an image that we “saw” the final hair/fur. This is time consuming particularly during character development, where we would like to see immediately at least approximately what the final hair does if we tweak a control hair, manual clump or any geometric hair parameter for that matter. Thus, we have been implementing an interactive hair viewer in the animation system as shown in Figure 21 (note that only about 10% of the final hairs are displayed in this case for clarity).

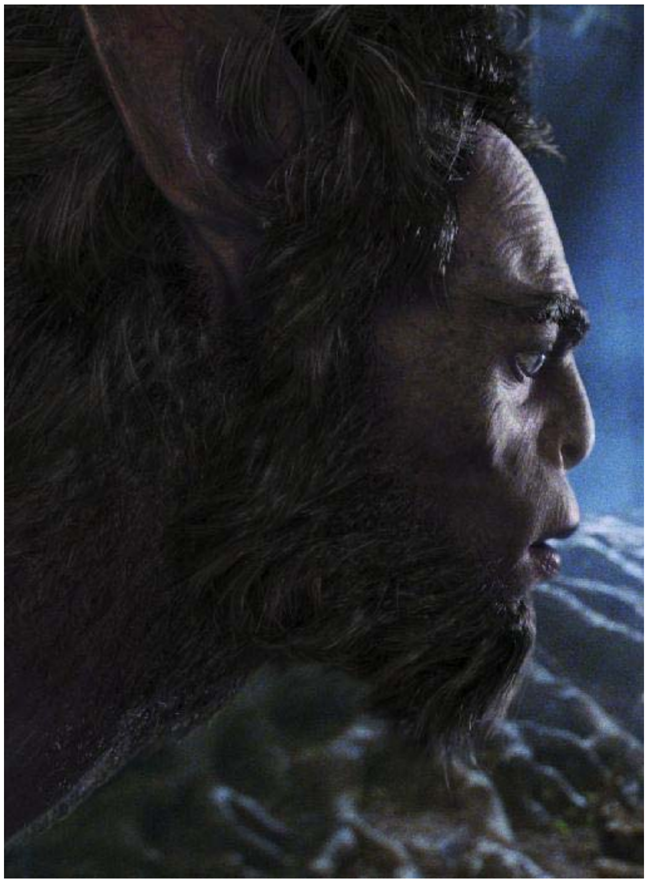


Figure 19: Facial and head hair of a centaur.

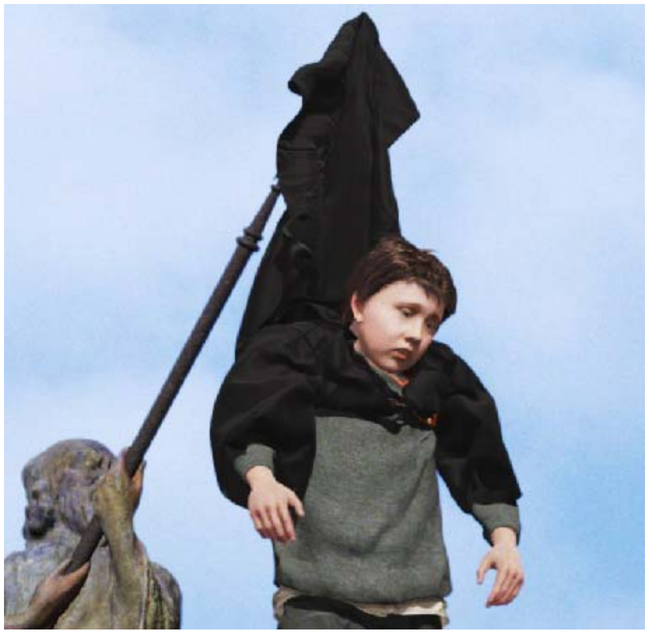


Figure 20: Digital hair to match real hair.

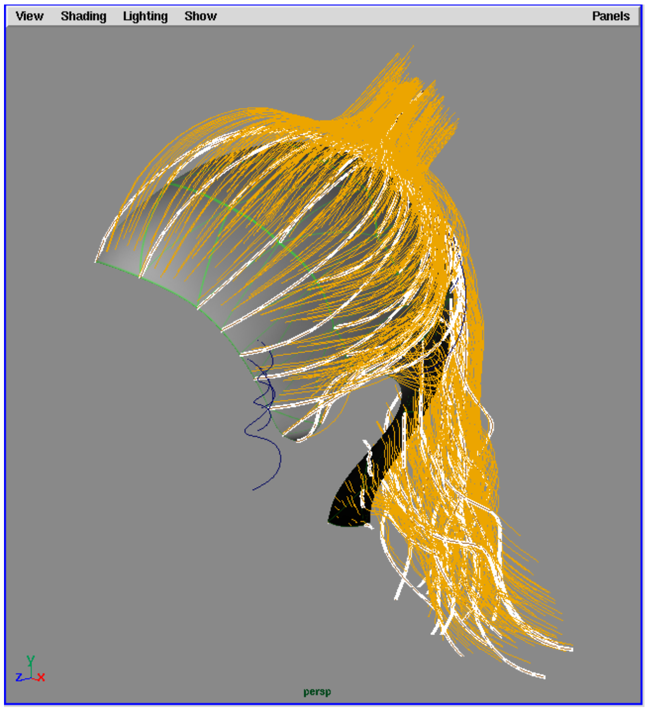


Figure 21: control (white) and final hair (orange) in our interactive hair viewer.

6 Discussion and Conclusions

We have presented our original pipeline for generating hair and fur. The system is still evolving today, being optimized and extended as new effects are needed. We think that it is a practical approach to generate realistic, life-like hair and fur. We also believe we have achieved our other goals of providing a flexible, robust and easy-to-use tool. With respect to performance and efficiency, breaking up the calculations into a static, frame-independent part and frame-dependent computations has made some difference. It has also helped with managing all the files, implementing the extra effects like clumping, and being able to tweak hair parameters at render-time. However, since we are rendering individual hairs, rendering times and memory requirements are still a concern. For instance, rendering the close-ups of Stuart's head in our examples with about 450,000 hairs originally took around thirty minutes per frame and 200 MB memory at video resolution on an SGI R10000 processor. At film resolution (2048x1556 pixels), the same frames took about one hour to render requiring 220 MB of memory on average. By moving over to recent Linux boxes, render times have improved by at least a factor of two. For human hair where we usually have on the order of tens of thousands rather than hundreds of thousands[14] of hair strands, render times are not so much of an issue.

Current extension plans for the hair/fur pipeline include improving animation and simulation support for the control hairs, better procedural techniques to handle interactions of hair/fur with external objects (e.g. hand touches hair), cloth/hair interactions, and collisions between hairs, which we currently ignore. Finally, another project we are currently working on is to utilize the real-time shading hardware support on our Linux platforms in order to generate RenderMan-like images in our interactive hair viewer .

[14] Or millions as for Fluffy, the three-headed dog in Harry Potter.

Acknowledgements

Many people have contributed since the original implementation by Clint Hanson and myself. Hiro Miyoshi wrote the first curve-combing tool. Alan Davidson has been there from the beginning to assist with pipeline issues. Evan Smyth helped out with several bottlenecks in the pipeline and the chain combing tool. Ross Kameny worked on the second generation, manual clumping tools as well as the initial development of the hair viewer. Murilo Coutinho implemented the paint combing tool and various other control hair improvements. I would also like to thank Bob Winter, Chris Yee, Darren Lurie, Sho Igarashi and many other people in production and software for their suggestions and feedback, as well as Sony Pictures Imageworks for their support.

References

[1] Joe Alter (http://www.joealter.com).

[2] Makoto Ando and S. Morishima, "Expression and Motion Control of Hair Using Fast Collision Detection Methods", in R. Chin et al., Editor, Image Analysis Applications and Computer Graphics, Springer Verlag, December 1995, pp. 463-470.

[3] Ken-ichi Anjyo, Y. Usami and T. Kurihara, "A Simple Method for Extracting the Natural Beauty of Hair", Computer Graphics (ACM SIGGRAPH Proceedings), 26, 2, July, 1992, pp. 111-120.

[4] Armin Bruderlin, "A Method to Generate Wet and Broken-up Animal Fur", The Journal of Vizualization & Computer Animation, 21, 5, December 2000, pp. 249-259.

[5] Agnes Daldegan, N. Magnenat Thalmann, T. Kurihara and D. Thalmann, "An Integrated System for Modeling, Animating and Rendering Hair", Eurographics'93, Proceedings, 12, 3, 1993, pp. 211-221.

[6] Rob Engle, "STUART LITTLE 2: Let the Feathers Fly", SIGGRAPH 2002 Course Notes #32, , July 2002.

[7] Allen Van Gelder and J. Wilhelms, "An Interactive Fur Modeling Technique", Graphics Interface'97, Proceedings, May, 1997, pp. 181-188.

[8] Dan B. Goldman, "Fake Fur Rendering9, Los Angeles, 1997, pp. 127-134.

[9] Larry Gritz, "RenderMan in Production", SIGGRAPH 2002 Course Notes #16, July 2002, pp. 135-160.

[10] Paul S. Heckbert, Editor, "Graphics Gems IV", Academic Press, 1994.

[11] James T. Kajiya and T. L. Kay, "Rendering Fur with Three Dimensional Textures", Computer Graphics (ACM SIGGRAPH Proceedings), 23, 3, July, 1989, pp. 271-280.

[12] Tae-Yong Kim, Ulrich Neumann, "Interactive Multiresolution Hair Modeling ans Editing", Computer Graphics (ACM SIGGRAPH Proceedings), 21, 3, July, 2002, pp. 620-629.

[13] Keisuke Kishi and S. Morishima, “Dynamic Modeling of Human Hair and GUI Based Hair Style Designing System”, Computer Graphics (ACM SIGGRAPH Sketches, Conference Abstracts and Applications), Orlando, 1998, p. 255.

[14] Nadia Magnenat Thalmann et al., Virtual Clothes, Hair and Skin for Beautiful Top Models”, Computer Graphics International’96, Proceedings, South Korea, 1996, pp. 132-141.

[15] Gavin Miller, “From Wire-Frames to Furry Animals”, Graphics Interface‘88, Proceedings, June, 1988, pp. 138-145.

[16] Ken Perlin, “Hypertextures”, Computer Graphics (ACM SIGGRAPH Proceedings), 23, 3, July, 1989, pp. 253-262.

[17] Pierre Poulin and A. Fournier, “A Model for Anisotropic Reflection”, Computer Graphics (ACM SIGGRAPH Proceedings), 24, 4, August 1990, pp. 273-282.

[18] Franco P. Preparata and M. I. Shamos, “Computational Geometry: An Introduction”, Springer Verlag, 1985.

[19] Steve Upstill, “The RenderMan Companion: A Programmer’s Guide to Realistic Computer Graphics”, Addison-Wesley Publishing Co., July 1992.

[20] Yasuhiko Watanabe and Y. Suenaga, “A Trigonal Prism-Based Method for Hair Image Generation”, IEEE Computer Graphics and Applications, 12, 1, January 1992, pp. 47-53.

Bibliography:-

1) N.Magnenat-Thalmann, S.Hadap, P.Kalra, **State of the Art in Hair Simulation**, International Workshop on Human Modeling and Animation, Seoul, Korea, Korea Computer Graphics Society, pp. 3-9, June, 2002

2) Sunil Hadap, Nadia Magnenat-Thalmann, **Modeling Dynamic Hair as a Continuum**, Eurographics 2001 Proceedings, Manchester, United Kingdom, September 2001.

3) S.Hadap, N.Magnenat-Thalmann, **Interactive Hair Styler based on Fluid Flow**, Eurographics Workshop on Computer Animation and Simulation'2000, Interlaken, Switzerland, Springer-Verlag, pp. 87-99,August,2000

4) Tae-Yong Kim and Ulrich Neumann, **Interactive Multiresolution Hair Modeling and Editing,** ACM SIGGRAPH 2002 Proceedings.

5) Tae - Yong Kim and Ulrich Neumann**, Opacity Shadow Maps,** Eurographics Rendering Workshop 2001.

6) Jerome Lengyel, Emil Praun, Adam Finkelstein, Hugues Hoppe**, Real-Time Fur over Arbitrary Surfaces,** ACM 2001 Symposium on Interactive 3D Graphics.

7) Johnny Chang, Jingyi Jin, and Yizhou Yu, **A Practical Model for Hair Mutual Interactions**, ACM SIGGRAPH Symposium on Computer Animation, San Antonio, July 2002, pp.73-80.

8) Yizhou Yu, **Modeling Realistic Virtual Hairstyles**, Proceedings of Pacific Graphics 2001, Tokyo, October 2001, pp.295-304.

9) Armin Bruderlin, **A method to generate wet and broken-up animal fur**, Journal of Visualization and Computer Animation 11(5): 249-259

10) J. Kajiya and T. Kay, **Rendering fur with three-dimensional textures**, SIGGRAPH Proceedings, Vol. 23, pp. 271-280, 1989.

11) Eric Plante, Marie-Paule Cani, Pierre Poulin., **Capturing the Complexity of Hair Motion,** *GMOD* numéro 1 volume 64 , January 2002.

12) Eric Plante, Marie-Paule Cani, Pierre Poulin, **A Layered Wisp Model for Simulating Interactions inside Long Hair,** *Computer Animation and Simulation 2001Proceeding* , Sep 2001.

13) C. Koh and Z. Huang, **A simple physics model to animate human hair modeled in 2D strips in real time**, Computer Animation and Simulation '01, pages 127--138, September 2001.

14) T. Lokovic and E. Veach, **Deep shadow maps**, SIGGRAPH Proceedings, pp. 385-392, 2000.

15) A. LeBlanc, R. Turner and D. Thalmann, **Rendering hair using pixel blending and shadow buffers**, The Journal of Visualization and Computer Animation, Vol. 2, pp. 92-97, 1991.