上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

1896 1935 1987 2006

Game Design & Development


Game Physics：Hair

Digital ART Lab,
School of Software,SJTU

Human Hairs

- Estimated Numbers
  - — Blonde, 150,000
  - — Brown, 110,000
  - — Black, 100,000
  - — Red, 90,000
- 120 to 200 hairs per cm^2
- Life span for each hair: ~ 3 year
  - — Grow speed: ~0.3mm/day
  - — Drop speed: 50~100/day
- Dry hair extension: ~30%
- Wet hair extension: ~50%
- 1 hair can lift 140 g or 111 g or 70 g
  - — 120K hairs can lift 1.5~2 tons

Digital ART Lab,SJTU

Case Study: Nvidia's Nalu Demo

- Long, blonde, flowing hair underwater
  - — Hair movement simulation
  - — Self-shadowing
  - — Reflectance model: simulate light scattering through individual strands of hair
  - — Real time dynamics and rendering !


Digital ART Lab,SJTU

Overview of Techniques

- Steps:
  - — Geometric generation
  - — Dynamics/collisions computation
  - — Shading

4095 individual hairs
123,000 vertices for hair rendering

Too much computation for physics!

- *Control hairs:*
  - — *a smaller set of hundreds of hairs*
  - — *for dynamics/collisions*

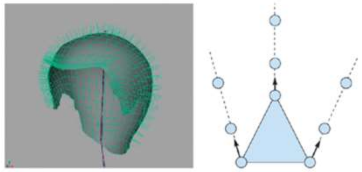
Hand animating control hairs is hard !

- — Physically based animation helped a lot, while lost 90% of the control

Digital ART Lab,SJTU

Geometry: Growing Hair

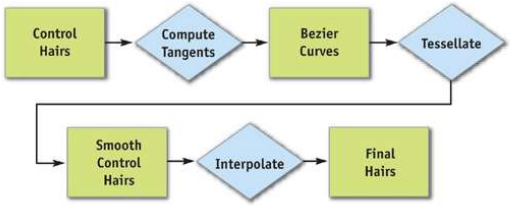
- Scalp geometry
- Grow <u>control hair</u> from each vertex of the scalp along the normal
- Balance: dynamics (physics) + human-controllable



(a) (b)

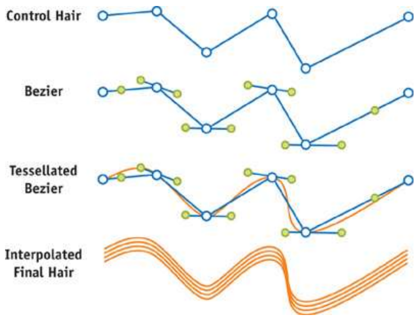
Data flow for hair construction

- Hair changes every frame
- Need to rebuild the final rendering hair set every frame



- Smoothing
- Increase density
- Use a dynamic vertex buffer to hold the vertex data

Tessellation and interpolation



(a)

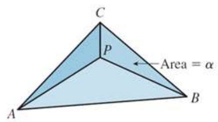
(b)

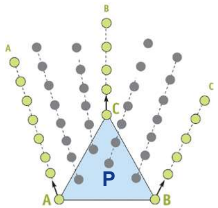
Hair Interpolation

- The interpolated hair is created using the scalp mesh topology
- Use <u>barycentric coordinates</u> to create new hairs inside

$P = A \times b_A + B \times b_B + C \times b_C$

$b_A + b_B + b_C = 1$
$b_A, b_B, b_C >= 0$





Dynamics using Verlet integration

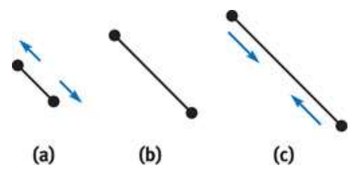
- Verlet integration
  - —More stable than Euler integration
  - —Simpler than Runge-Kutta integration

  - —Do not need velocity
  - —Δt need to be fixed
  - —Easy to add constraints

$$x(t+\Delta t)=x(t)+d\cdot(x(t)-x(t-\Delta t))+a(t)\Delta t^2$$

Constraints

- Rest length: l_r

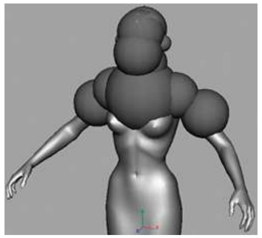


$$\mathbf{x}_1' = \mathbf{x}_1 + (\mathbf{x}_2 - \mathbf{x}_1)\cdot\frac{\|\mathbf{x}_2-\mathbf{x}_1\| - l_r}{2\,\|\mathbf{x}_2-\mathbf{x}_1\|}$$
$$\mathbf{x}_2' = \mathbf{x}_2 - (\mathbf{x}_2 - \mathbf{x}_1)\cdot\frac{\|\mathbf{x}_2-\mathbf{x}_1\| - l_r}{2\,\|\mathbf{x}_2-\mathbf{x}_1\|}$$

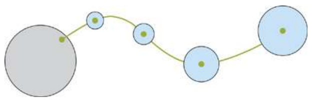
Other constraints: collisions

Collision Detection

- Keep it simple and fast
- A rig of spheres:
  - —Work well
  - —Easiest to implement

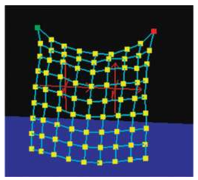


- Problem: long line segments
- Solution: Pearls for collisions



Fins

- Pre-built solid fins, skinned to the skeleton
- Problem: fins look quite stiff
- Solution: hair code ➔ cloth simulation
  - —Blend results with skinned geometry, using weight map

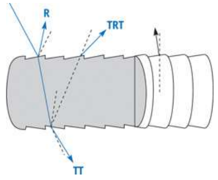

Hair Shading

- **Two parts:**
  - —**a local reflectance model for hair**
  - —**a self-shadowing computing method**
- **Real-time reflectance model for hair**
  - —Marschner model: a comprehensive, physically based representation of hair reflectance
- **Real-time volumetric shadows in hairs**
  - —Opacity shadow maps

Marschner Reflectance Model (2003)

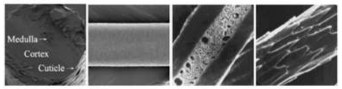
- Treats each hair fiber as a translucent cylinder
- Consider three reflectance paths for hairs: R, TT, TRT
  - —Visually significant for hair

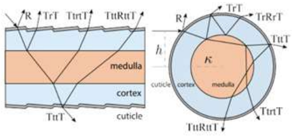
$S = S_R + S_{TT} + S_{TRT}$

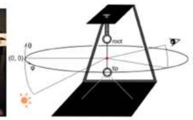


Advanced Model

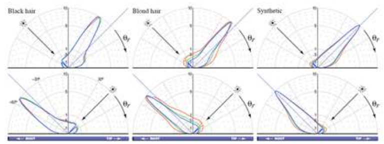
[Yan 2015]



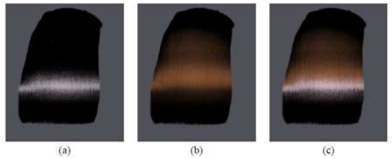


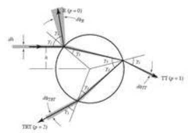


Bidirectional Scattering function



$$S(\phi_i, \theta_i; \phi_r, \theta_r)$$



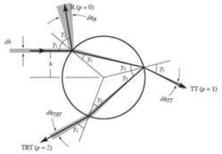


Results Comparison

Factorization

- because of the symmetry of a cylinder
  - —the 4D scattering function can be factored into a product of two 2D terms.
  - —*M,* captures the θ dependence,
  - —*N*, captures the $\emptyset$ dependence.
  - —derive *N* for the case of a circular cylinder
  - —an approximation to extend *N* to elliptical cylinders, and procedures for efficiently approximating *N*

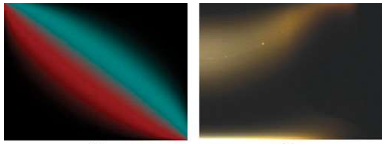
$$
\begin{aligned}
S(\phi_i,\theta_i;\phi_r,\theta_r) = \\
& M_R(\theta_h)N_R(\eta'(\eta,\theta_d);\phi)/\cos^2\theta_d + \\
& M_{TT}(\theta_h)N_{TT}(\eta'(\eta,\theta_d);\phi)/\cos^2\theta_d + \\
& M_{TRT}(\theta_h)N_{TRT}(\eta'(\eta^*(\phi_h),\theta_d);\phi)/\cos^2\theta_d
\end{aligned}
$$



Digital ART Lab,SJTU

Implementation

- both *M* and *N* are functions of only two parameters
- compute a lookup table for each of these functions and encode them in 2D textures



(a) (b)

Lookup textures

Digital ART Lab,SJTU

Real-Time Volumetric Shadows in Hair

- **Common methods**
  - —**Stencil shadow volumes**
  - —**Shadow maps**
- **Neither work well for shadows on hair**
  - —**Large amount of geometry: intractable for stencil shadow**
  - —**Highly detailed geometry: severe aliasing for shadow map**
- **Opacity shadow maps**
  - —**Extend shadow mapping to handle volumetric objects and anti-aliasing**

Digital ART Lab,SJTU

Opacity Shadow Maps

- Allow fractional shadow values
- Occlusion test
  - —rather than a simple binary test
  - —Measure the percentage of light that penetrates over a pixel

Opacity thickness

$$T(x, y, z) = e^{-k\sigma(x,y,z)},$$

$$\sigma(x, y, z) = \int_0^z r(x, y, z')\,dz'$$

Idea: compute a discrete set of z values $z_0 \ldots z_{n-1}$, then determine in-between values by interpolation

$$\sigma(z) = \frac{\sigma(z_i)(z - z_i) + \sigma(z_{i+1})(z_{i+1} - z)}{(z_{i+1} - z_i)},$$

where $z_i < z < z_{i+1}$

Implementation

- **Use GPU hardware blending**
- **n = 16**
  - —**Naïve approach, store** (x, y, z_i)**in 16 textures**
  - —**Need render 16 times to generate the opacity shadow map**
- **Optimization:**
  - —Each (x, y, z_i) requires only 1 channel
  - —pack up to 4 sigma values into a single 4-channel texture
  - —Render passes: from 16 → 4
- More optimization:
  - —Use multiple render targets(MRT), allow render to up to 4 different textures simultaneously
  - —Render passes: from 4 → 1
- Lookup

Digital ART Lab,SJTU

Results

Digital ART Lab,SJTU

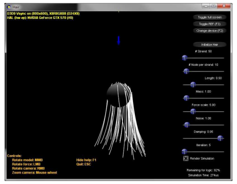
Alice's Hair (2011)

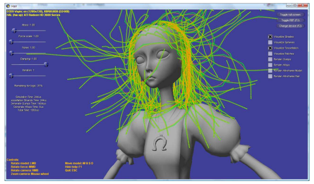
- **Game 《Alice: Madness Returns》(Spicy Horse, EA)**
- **UE3, Xbox360, PS3**

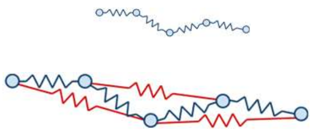
- **Hair tech developer: Milo Yip**
- **Techniques:**
  - —**Mass-Spring**
  - —**Verlet integration**
  - —**Uniform cubic B-splines => LOD**
  - —**Kajiya-Kay reflectance model**
  - —**Collision detection**
  - —**Alpha blending**
  - —**SIMD vector optimization**

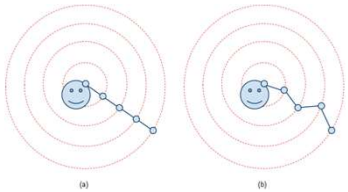
Digital ART Lab,SJTU

Some Techniques









Results

Further work

Demo