The Making of "Nalu"

Matthias Wloka
NVIDIA Corporation

Acknowledgement

- Hubert Nguyen
- William Donnelly

Long, Blonde Hair Rendering

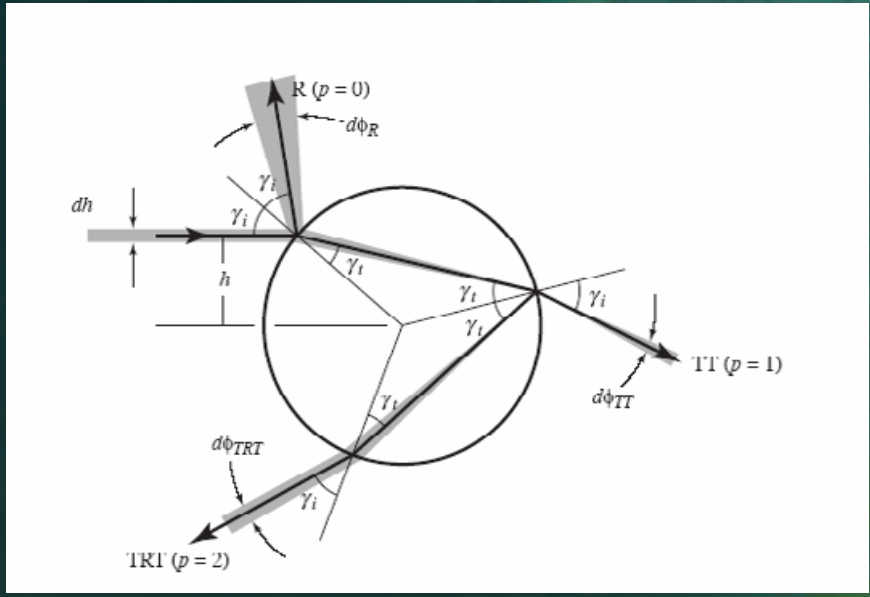
Long, Blonde Hair Rendering

- **Long**
  - Requires dynamic animation
    - Thus cannot bake lighting
  - Requires lots of hair
    - Thus shading has to be fast
- **Blonde**
  - Three visible highlights, black only has one
  - Shadows much more visible

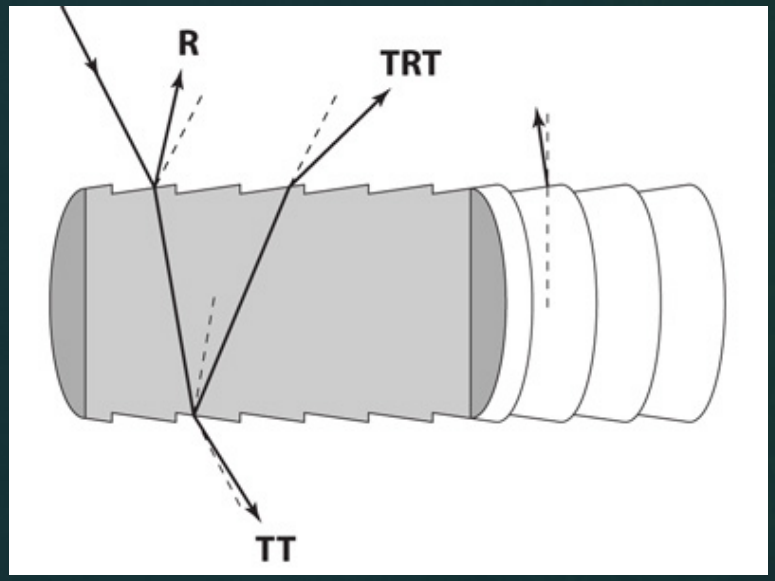
Acknowledgements

- "Light Scattering from Human Hair Fibers"
- By Steve Marschner, Henrik Wann Jensen, Mike Cammarano, Steve Worley, and Pat Hanrahan
- SIGGRAPH 2003

Paper Models three Distinct Highlights



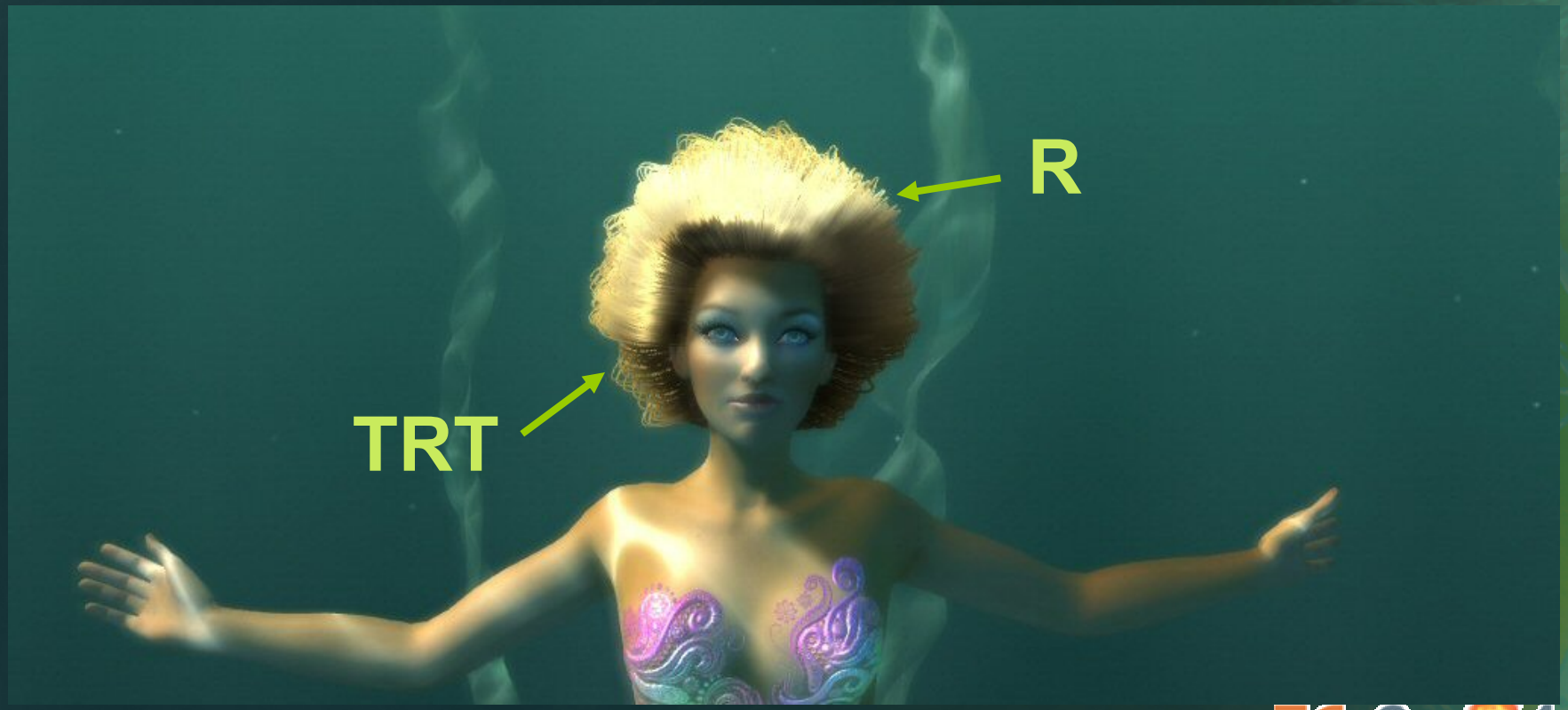
- Consider only 3 most significant terms
  - R, TT, TRT



- Uses path notation
  - R is reflection
  - T is transmission

R and TRT Highlights

- R – White primary highlight
- TRT – Colored secondary highlight



TT Highlight

- TT – strong forward scattering component
  - Important for underwater hair

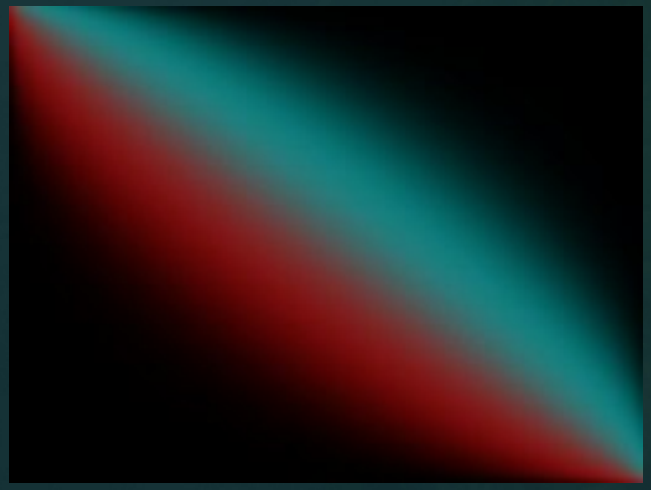
The Reflectance Model

- Three main angles
  - Light
  - Eye
  - Anisotropy
- Factor into lower dimensional terms
  - M_R (thetaH) * N_R (thetaD, phiD)
    + M_TT (thetaH) * N_TT (thetaD, phiD)
    + M_TRT(thetaH) * N_TRT(thetaD, phiD)
- 2D functions are encoded in textures
  - Texture maps are faster than heavy math
  - Use of mip maps eliminates “shader aliasing”

Use 2D textures to encode look-up tables

- **cos(thetaL), cos(thetaE)**
  → M_R, M_TT, M_TRT, cos(thetaH)

- cos(thetaH), cos(phiD)
  → N_R, N_TT, N_TRT



It's tweakable

- Highlights:
  - Separation
  - Strength
  - Width
- Hair color
- Extinction coefficient
- Index of refraction

Shadowing

- Based on
  "Opacity Shadow Maps" (OSM)
- By Tae-Yong Kim and Ulrich Neumann
  SIGGRAPH 2001

Why Opacity Shadow Maps?

- **Opacity Shadow Maps vs Shadow Maps**

  "What percentage of light is blocked from here?"

  vs.

  "Is the light blocked from here?"

- **Thus supports AA edges and volumetric rendering**

- **Regular shadow maps alias around edges**
  - Hair is 100% edges!

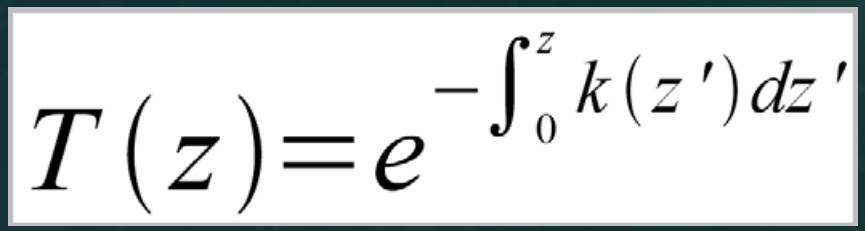
Results from Kim & Neumann

No Shadows

15 slices

255 slices

Opacity Equation

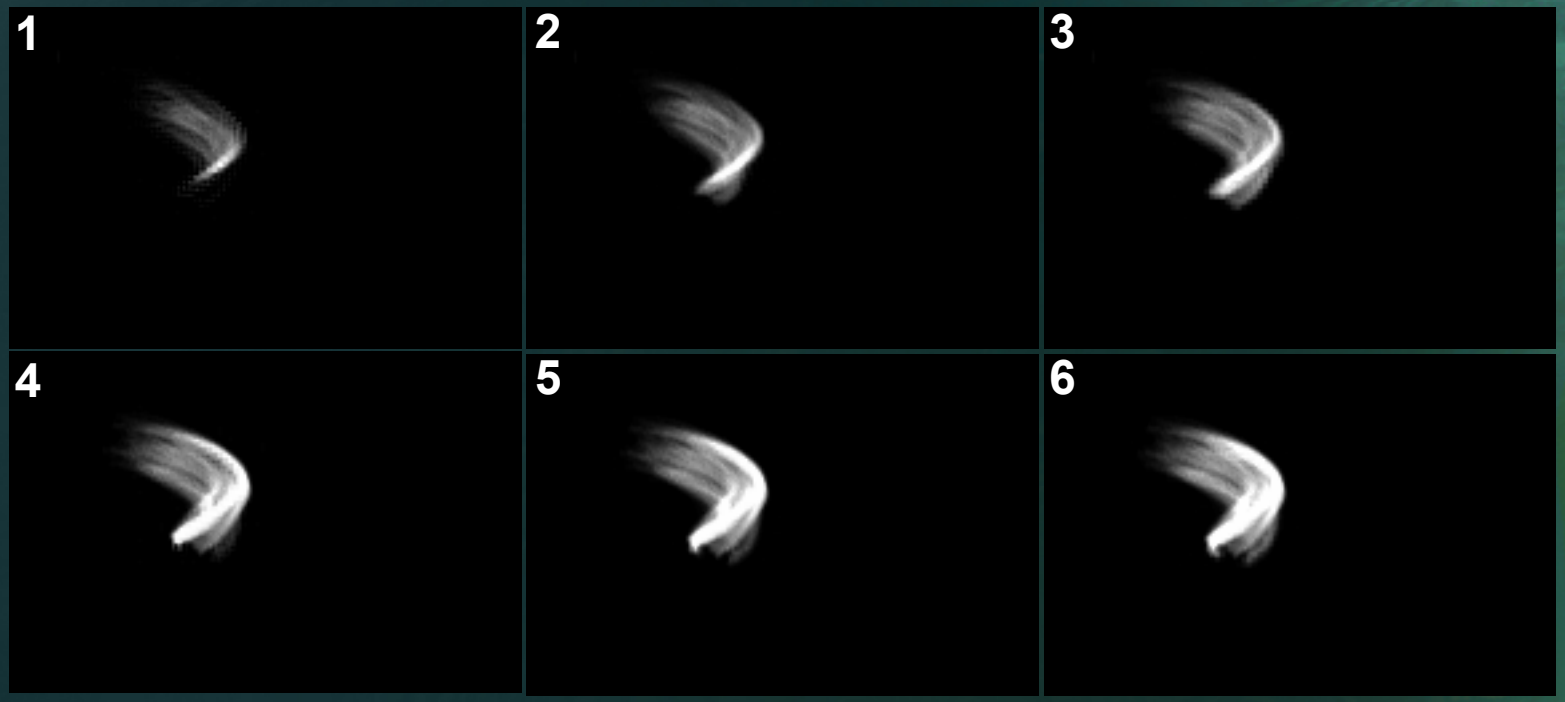


- T(z): amount of light penetrating to depth z
- For discrete case (hair):
  - Integral is sum over all strands between light and point being shadowed
- Compute sum via additive blending
  - “Extinction coefficient” K controls darkness of shadows

Creating the Opacity Maps

- **Choose 16 slicing planes in hair**
  - Uniform distribution based on hair bounding sphere
- **For each hair-pixel and for each plane**
  - Is hair-pixel closer to light than plane?
    - Yes: add hair to contribution (plane)
    - No: do nothing

OSM Slices

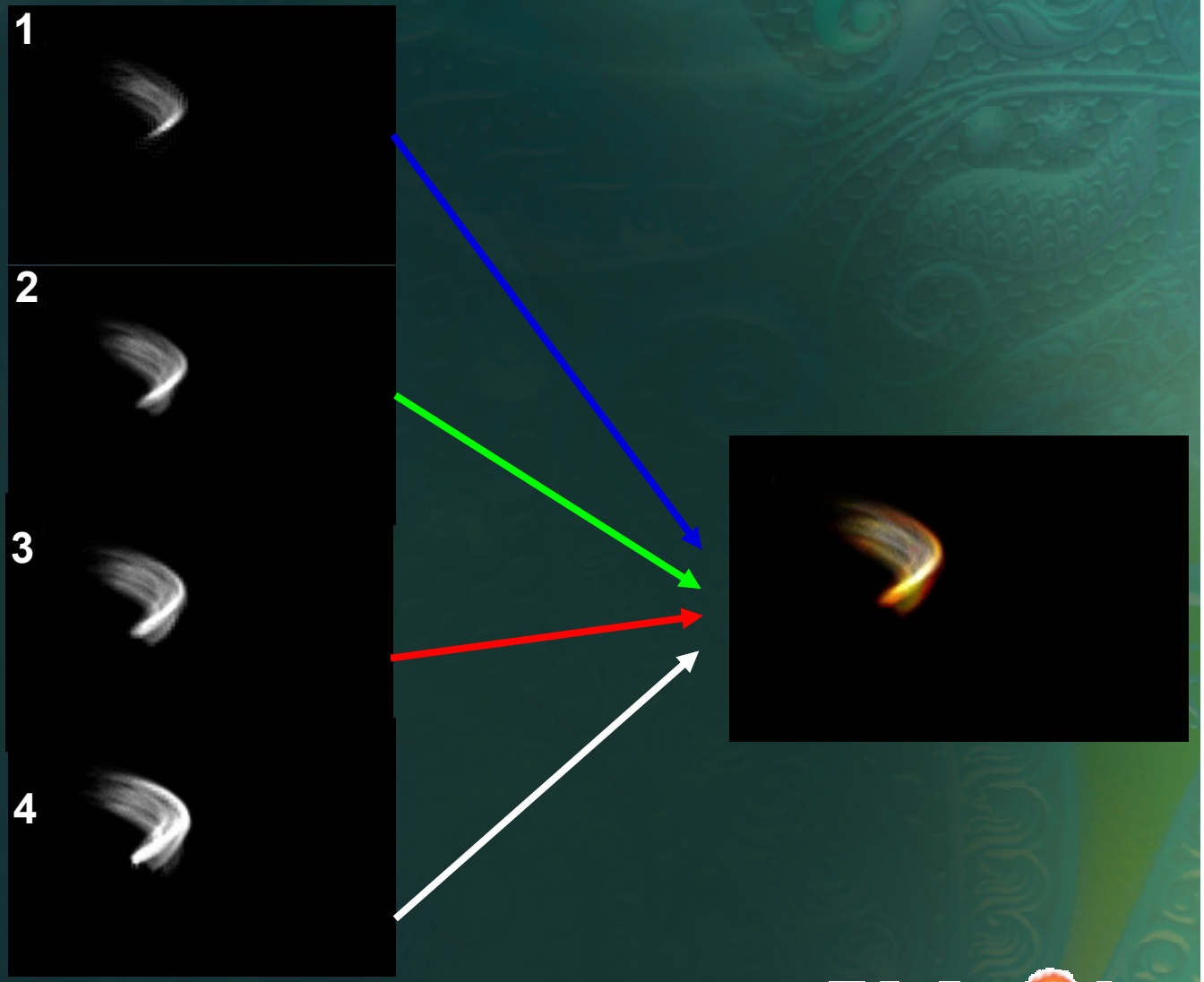


… up to 16

OSM Creation

- **Render hairs to 16 slices**
  - Original implementation : 16 render passes (RP)
- **In practice, 4 Render Passes**
  - 4 RP x 4 BGRA targets = 16 components/slices
- **Bonus: use Multiple Render Targets**
  - Single render pass instead of 4
    - Saves geometry computations
  - MRT shader is simple
    - 4 SLT and 4 MUL instructions
- **Can use lower hair LOD**

OSM Slices, 1 Render Pass BGRA packing



Vertex Shader Implementation

- **Compute light space position of the (Hair) fragment**
  - Add z-bias to counter limited z-resolution
    - Just like regular shadow maps
- **Hair-pixel position in light space determines:**
  - Which opacity maps to look in
    - Depending on light space Z
  - Where in opacity map to look in
    - Depending on light space X,Y

Pixel Shader Implementation

$$T(z) = e^{-\int_0^z k(z')\,dz'}$$

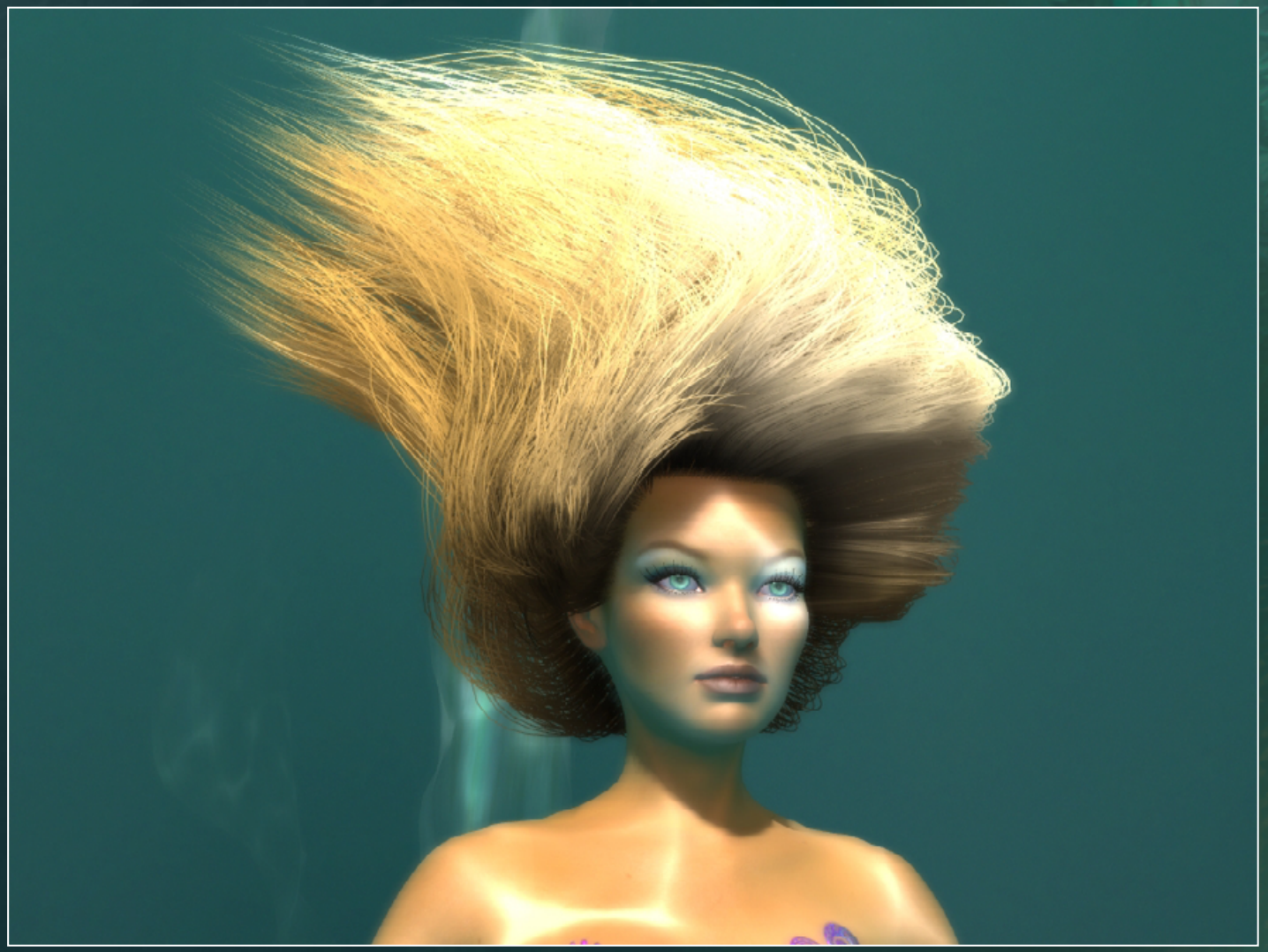
We know the value of the integral at each plane

- Compute in-between values by linear interpolation
- Interpolated value is a linear combination of plane values
- Equivalent to a 16 component dot-product:
  - dot(osm1weight, lookup1) + … + dot(osm4weight, lookup4)

- **Compute opacity by exponentiation**

In the demo…

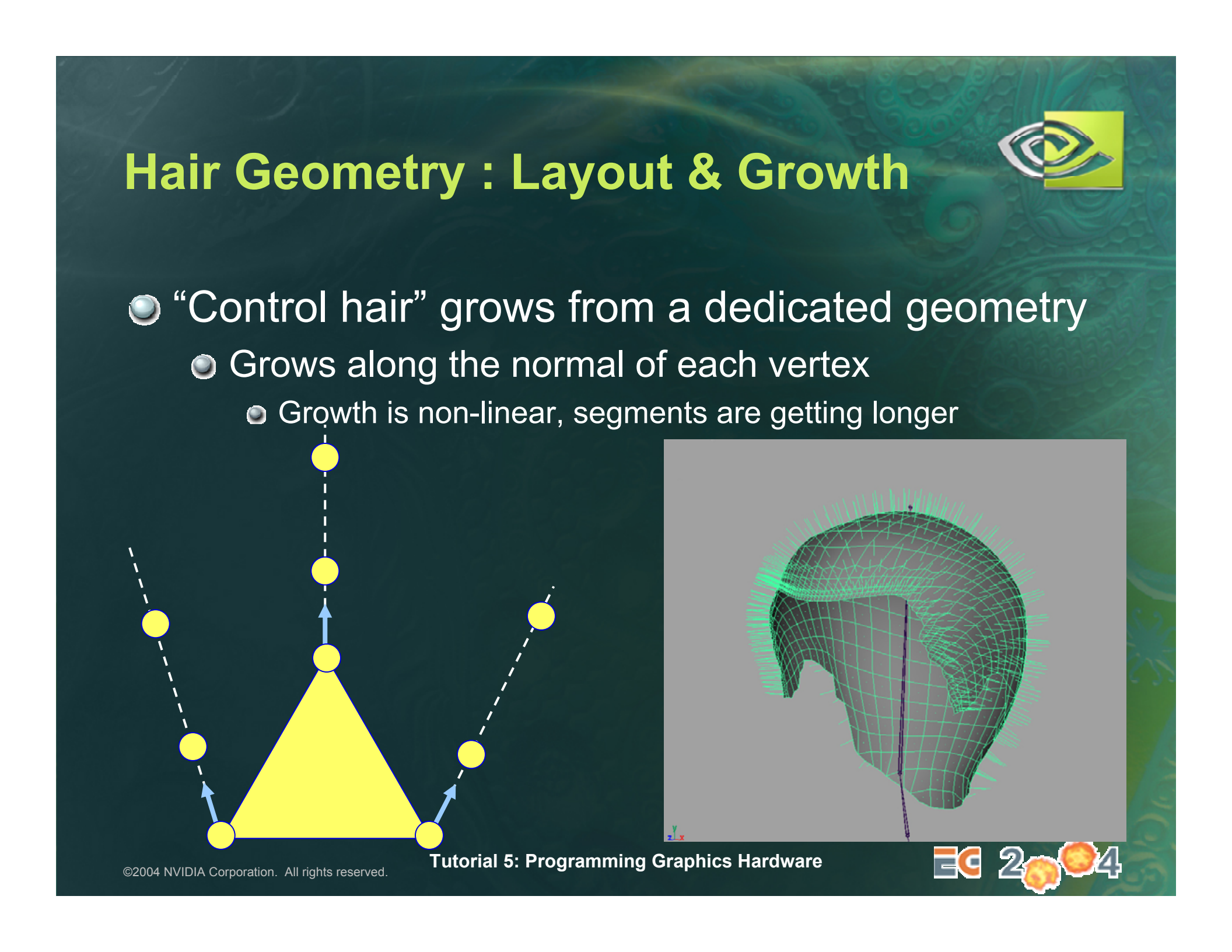
Hair Geometry & Dynamics



Hair Geometry : Overview

- 4095 individual hairs driven by 762 “control hairs”
- “Control hairs “
  - Set of hairs that is really driven by dynamics/collisions
  - Based on a particle system, where particles are connected by distance constraints
  - Grown from a reference geometry
- “Fine hair” geometry is created by smoothing & interpolating the “control hair”

Hair Geometry : Layout & Growth



- "Control hair" grows from a dedicated geometry
  - Grows along the normal of each vertex
    - Growth is non-linear, segments are getting longer

Hair Geometry : Control Hairs (left image)

- Physics/dynamics/collisions are performed on the control hairs

Control Hair

Fine Hair

Hair Dynamics

- Based on a particle system
- Uses the “Verlet” integration
  - previous frame position to compute velocity
    - Less sensitive to frame rate, less prone to explosion

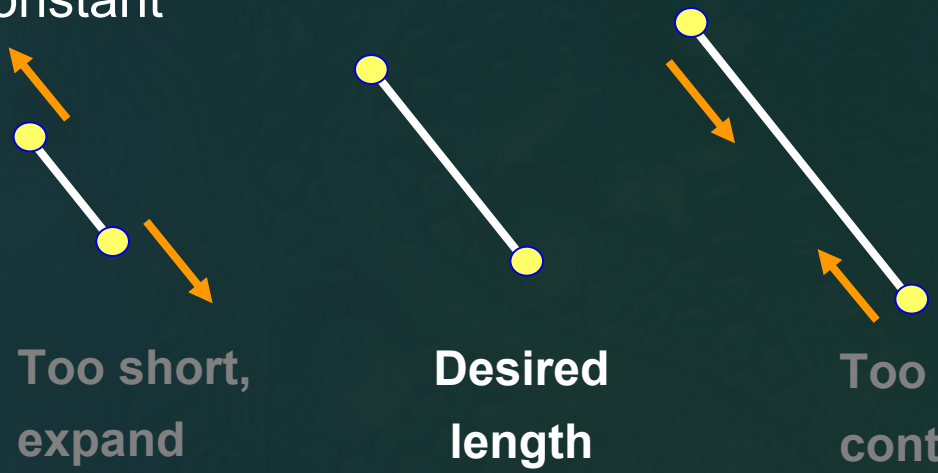
$$\mathbf{x}' = 2\mathbf{x} - \mathbf{x}^* + \mathbf{a} \cdot \Delta t^2$$

$$\mathbf{x}^* = \mathbf{x}$$

Reference: “**Advanced Character Physics**”
Thomas Jakobsen, IO Interactive, Denmark.

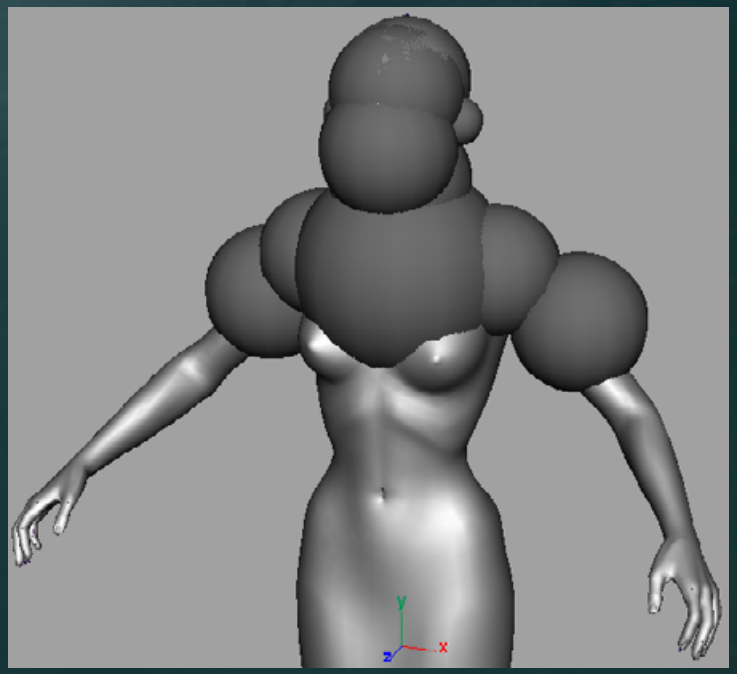
Hair Dynamics: constraints

- Two constraint types:
  - Infinite mass: applied to the “hair root” particles. Allows the head to “pull” the hair.
  - Distance constraints: forces “control hair” segments length to stay constant

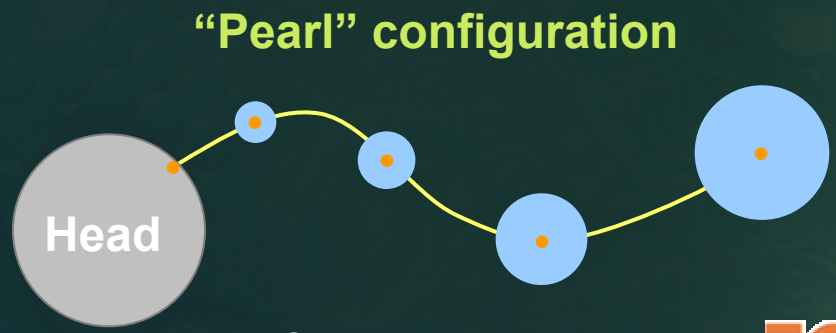


- If we apply those constraints iteratively, the particles will globally converge to the desired solution

Hair Dynamics: Collisions

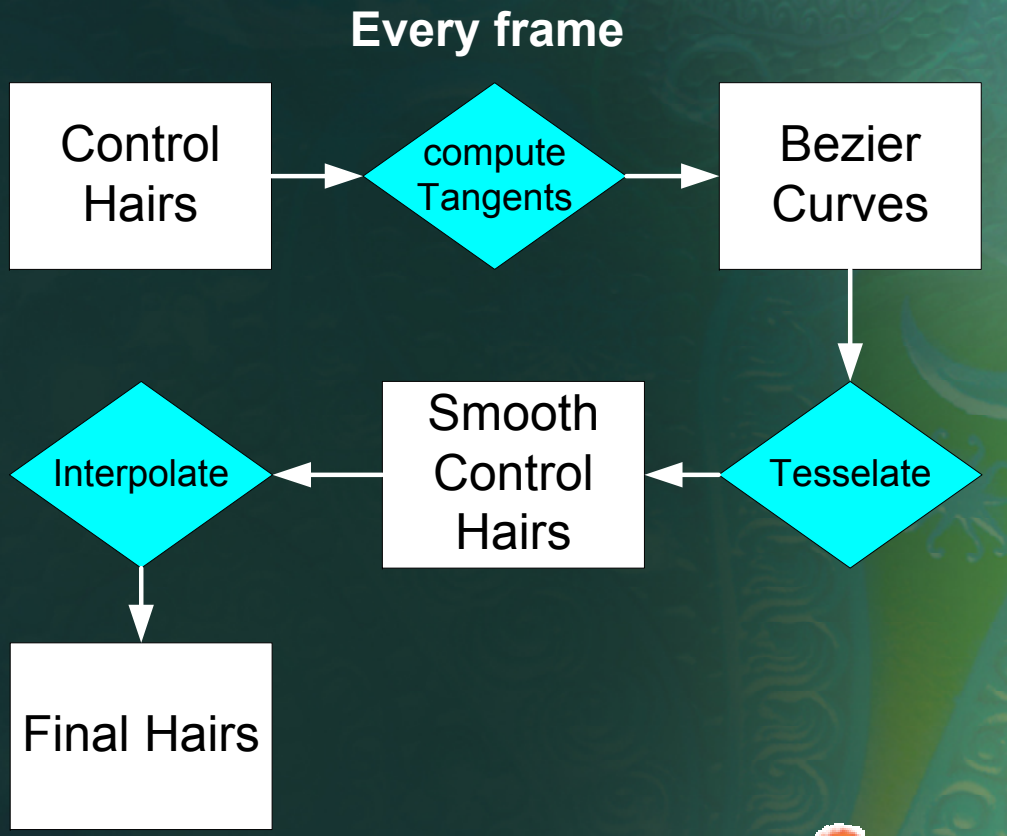


- Quite simple
- Nalu has a set of Spheres
- Control hair has 1Sphere/CV
  - CV Sphere grows towards the tip of the hair
  - Prevents collision sphere from slipping between 2 CVs

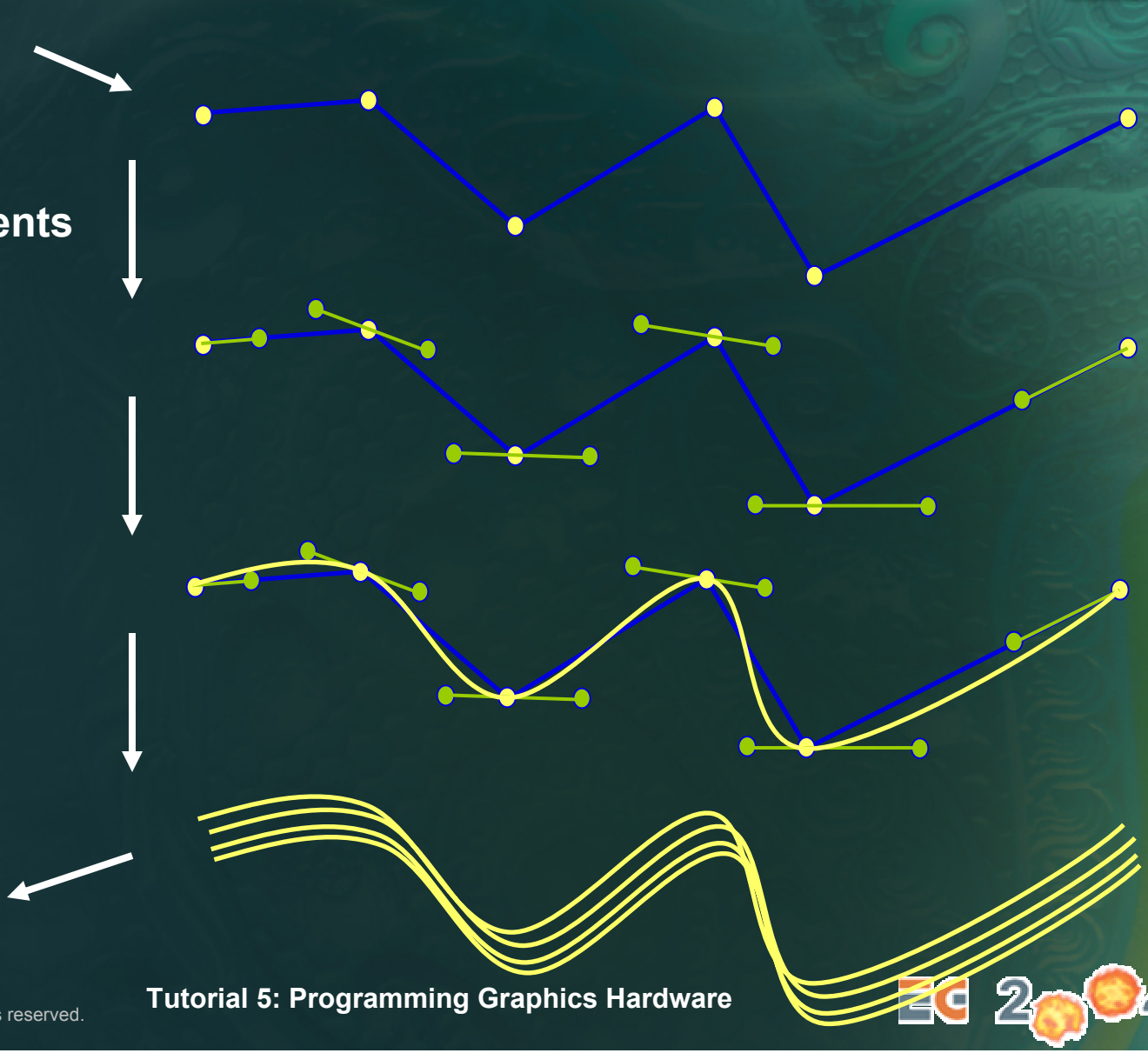


Hair Geometry : Data Flow

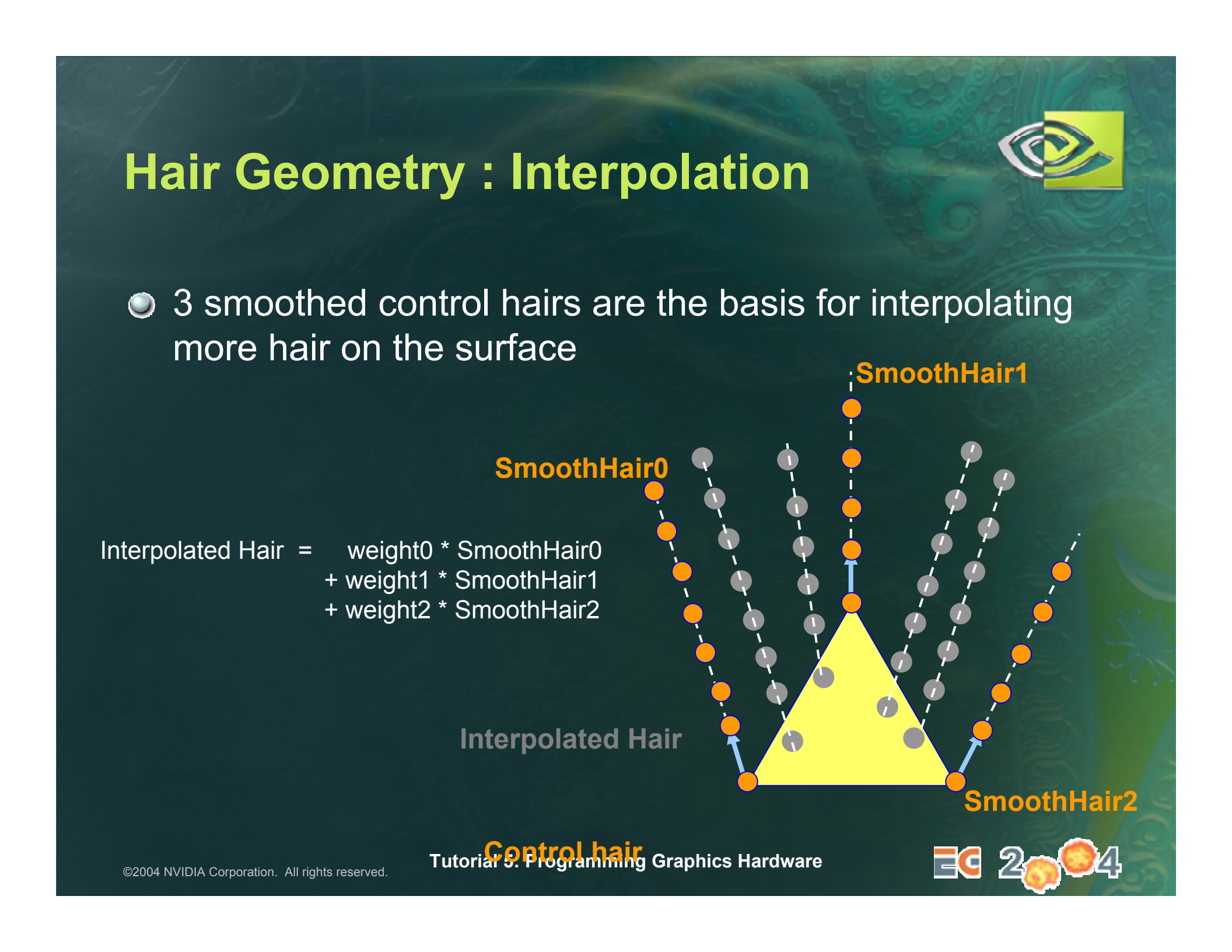
- Line lists or strips are sent to the GPU In World Space



Hair Geometry : Lifecycle (per frame)



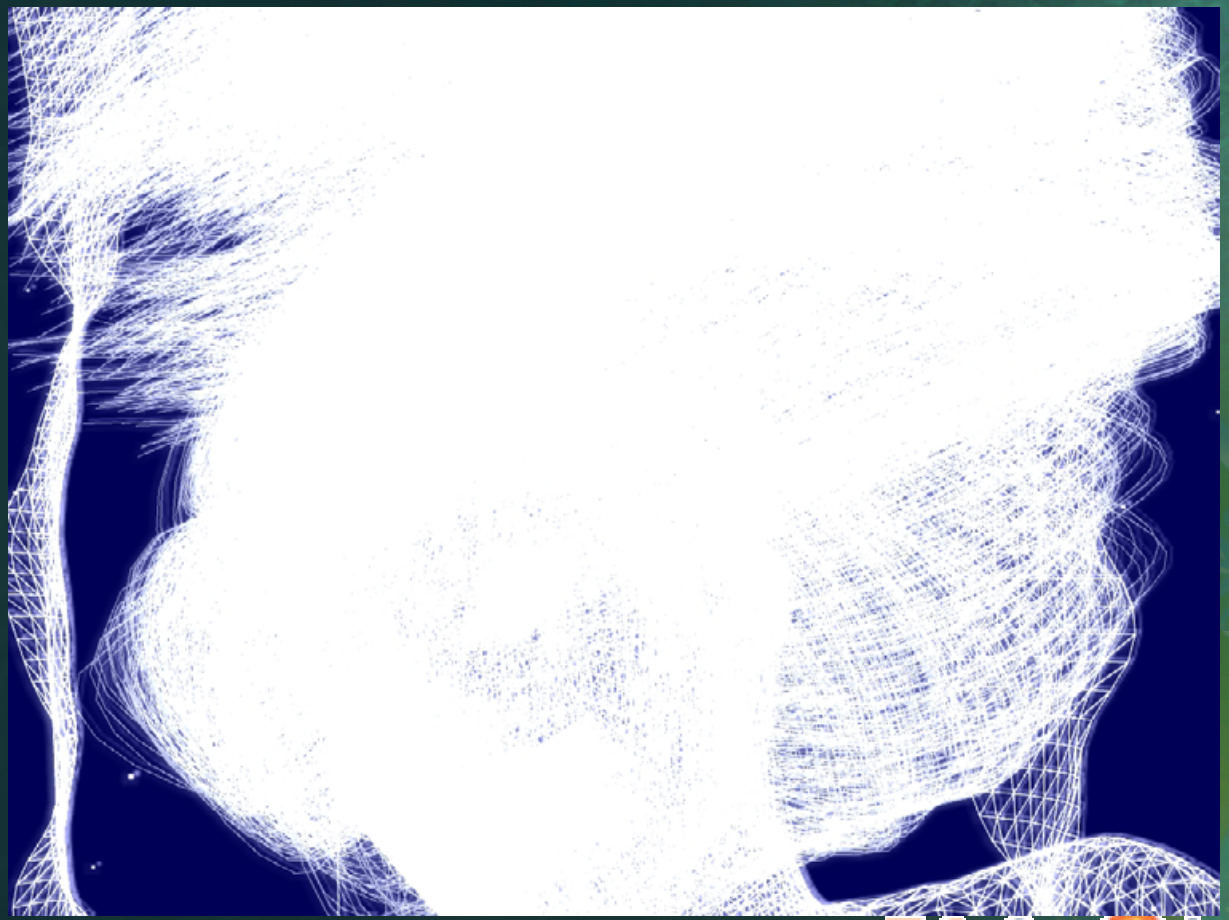
Hair Geometry : Interpolation



- 3 smoothed control hairs are the basis for interpolating more hair on the surface

Interpolated Hair = weight0 * SmoothHair0
+ weight1 * SmoothHair1
+ weight2 * SmoothHair2

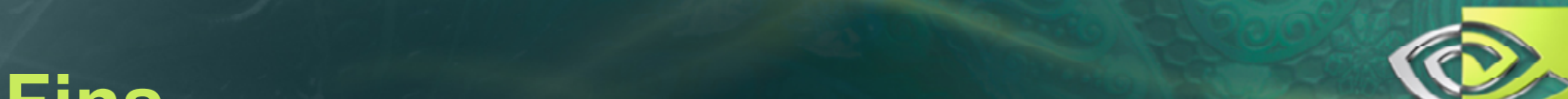
Hair Geometry : Interpolation



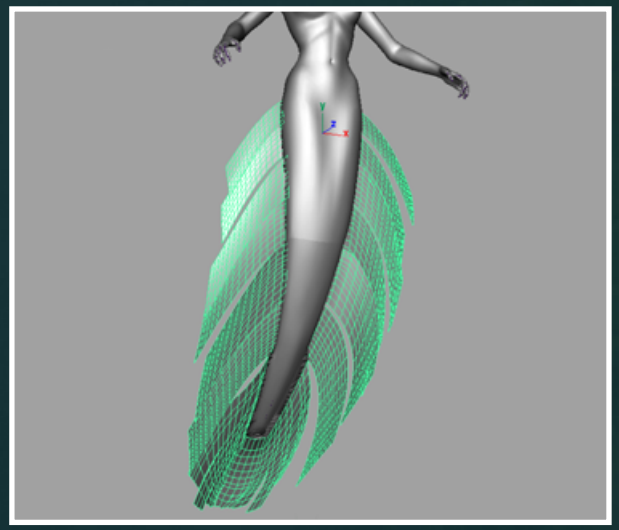
- Hair interpolation happens *after* Bezier tessellation
  - it's faster

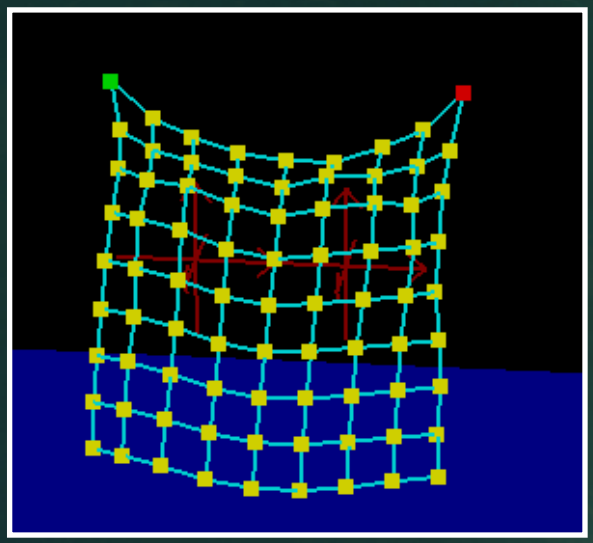
123k vertices in Hair

Fins

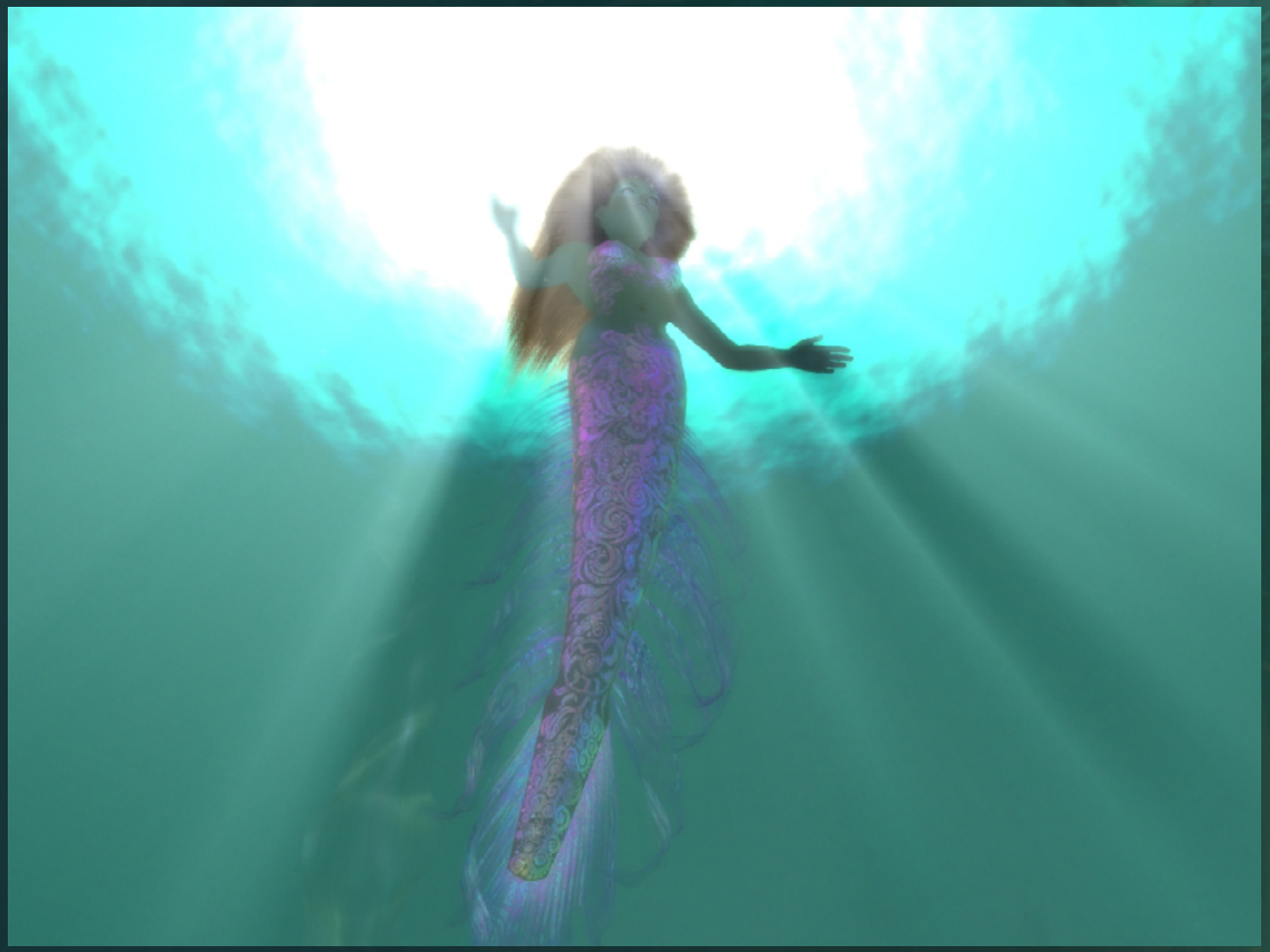


- Fins are a cloth simulation.
  - Any mesh can be turned into a cloth by using triangle edges as constraints
- Same physics as hair
- Normals are recomputed each frame



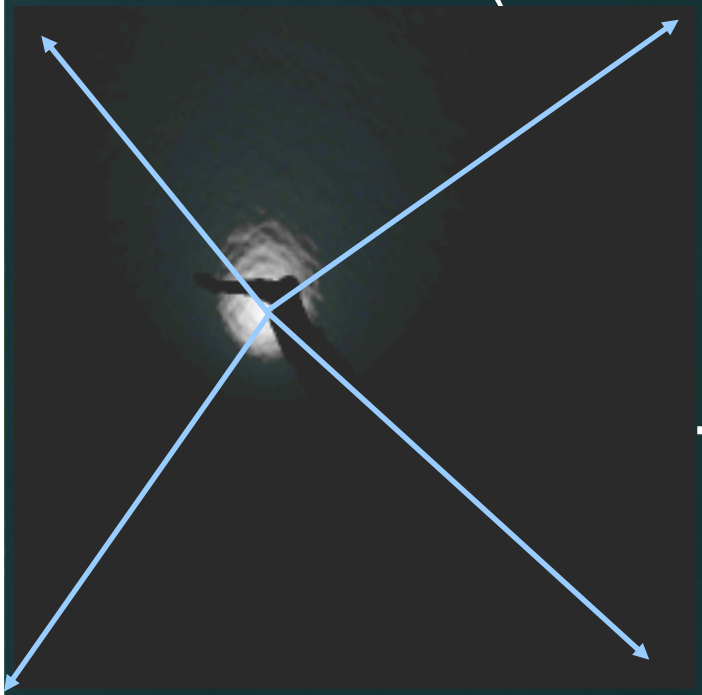


Shafts of Light : “God Rays”

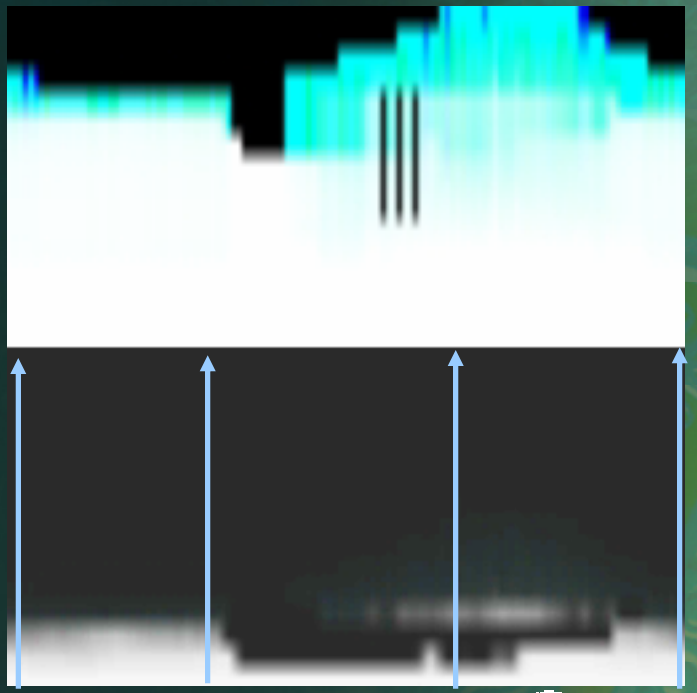


Radial Blur effect

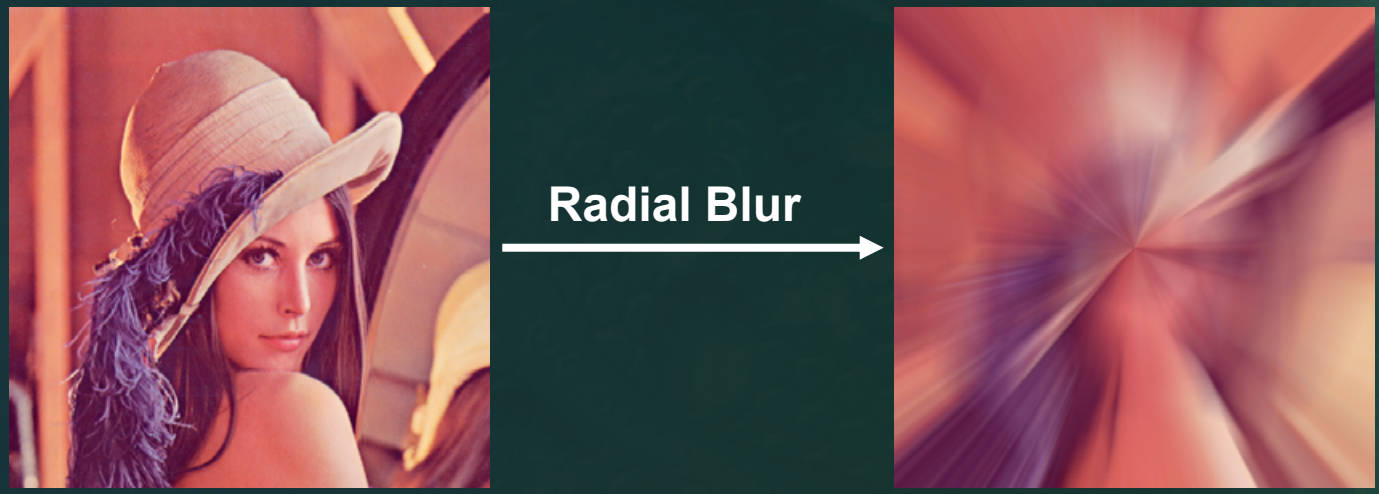
- Render bright regions
- Render shadow caster in alpha
- Blur in the radial direction, subtract occlusion factor (from alpha channel)



To Polar Coordinates



Shafts are based on a Radial Blur effect

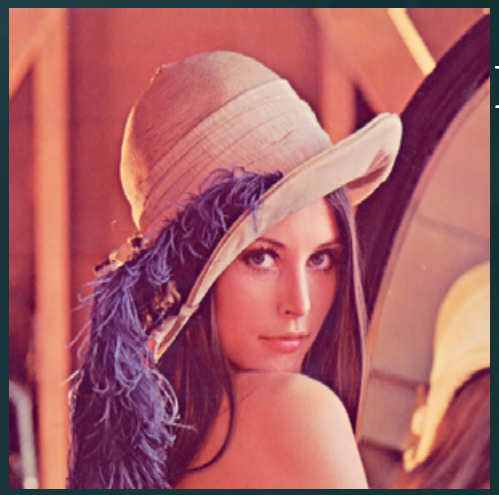


Radial Blur effect

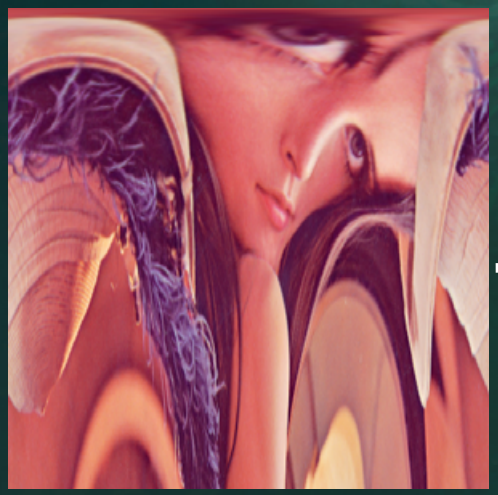
- Transform into polar co-ordinates (x,y)->(r,theta)
  - Use a grid where position = (r,theta) and texcoord = (x,y)
- Blur in the radial direction
- Transform back into Cartesian co-ordinates
  - Use the same geometric warping, but with positions and texture co-ordinates reversed

From Cartesian to Polar coordinates

Radial Blur effect : visuals



Rectangular to polar



Vertical blur

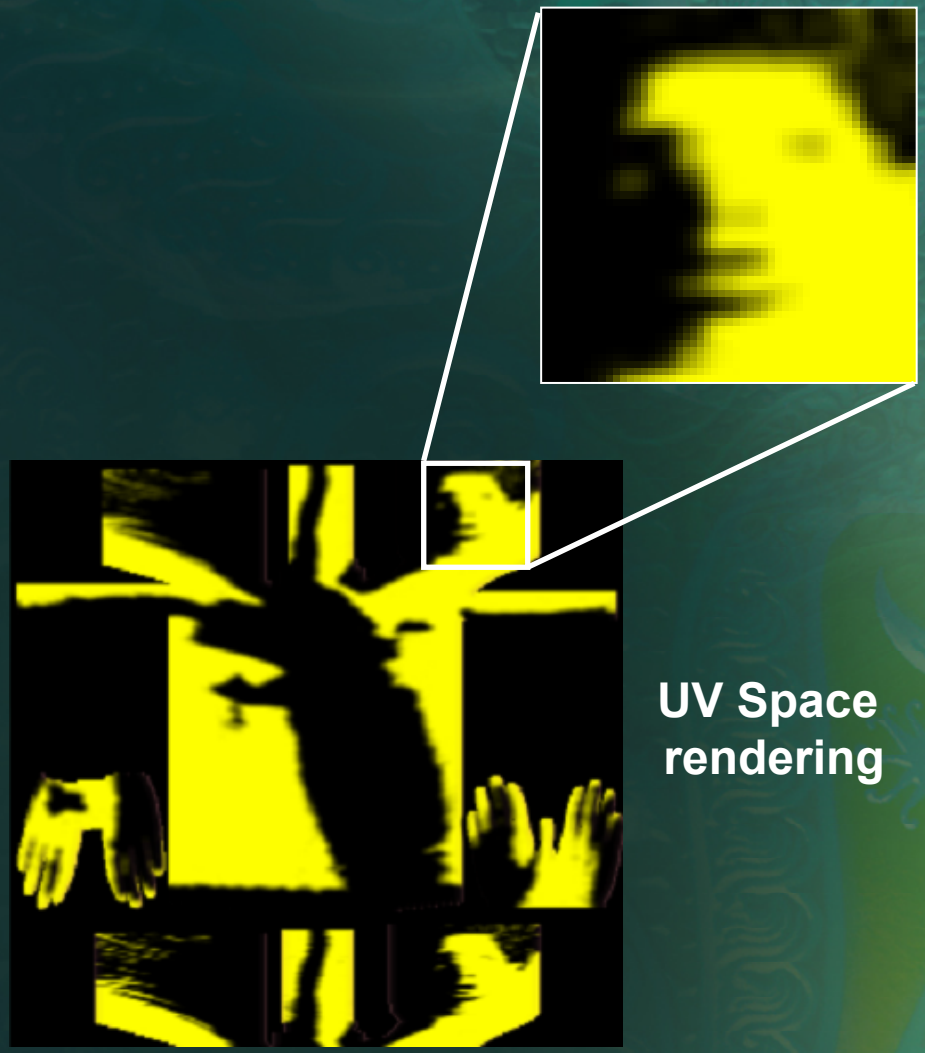
Polar to rectangular

“God Rays” in the Demo

Soft Shadows

- Soft shadowing based on Texture Space Diffusion (by Simon Green)
  - Described in “GPU GEMS” book (page 272)
- Basically:
  - Do the regular shadow mapping computations
  - But render in Texture Space
    - Using the UV coordinates as Vertex Shader Output Position
  - Blur the Texture Space B&W shadow result
  - Use the blurred shadow result in place of shadow compare when rendering the character

Soft Shadows: visualizations



UV Space rendering

Soft Shadows: challenges

- Unfold character in UV Space
- Visible Seams were UV are not continuous

- Workarounds
  - Improve UV layout
  - Do shadows in UV space, Lighting in Camera space

Future work

- Move more work to the GPU
  - Physics
  - Collisions (e.g., Simon's cloth demo)
  - Curves Tessellation
  - Normal / Tangent computation
  - Hair Interpolation
  - Anything, really :-)


Questions ?

Vertex Shader Optimization

- Compute percentage contribution of each plane to the current fragment in the vertex shader:

OSM1weight = max(0, 1 – abs(dist – depth1)*inverseDeltaD);

  - Dist is hair distance from light source
  - Depth1 is distance of first 4 planes packed in a float4
  - inverseDeltaD is 1/(distance between planes)
  - OSM[2-4]weight calculated similarly