VGP352 – Week 6

- Agenda:
  - Fur rendering 3 ways
    - Goldman's “fake fur”
    - “Shells and fins” fur
    - Banks BRDF on large hairs

12-February-2008

© Copyright Ian D. Romanick 2008

fakefur

- Developed by Dan Goldman at ILM
  - A *much* faster version of the “realfur” algorithm used at ILM for close-up shots

© Copyright Ian D. Romanick 2008

fakefur

- Developed by Dan Goldman at ILM
  - A *much* faster version of the “realfur” algorithm used at ILM for close-up shots
- Makes several simplifying assumptions:
  - Geometry of individual hairs is not visible
  - Hairs are truncated cones
  - The length of each cone is much greater than the radius of either end
    - Can't be used to render 5 o'clock shadow!
  - Radius of the base is greater than the radius of the other end
  - All hairs in an area have identical geometry

© Copyright Ian D. Romanick 2008

Algorithm Overview

- Compute average hair geometry in sample area
- For each light:
  - Compute hair-over-hair shadow attenuation
  - Compute reflected luminance of hair
  - Compute hair-over-skin shadow attenuation
  - Compute reflected luminance of skin
  - Compute hair / skin visibility ratio
  - Blend skin and hair reflected luminances using hair / skin visibility ratio
- Sum per-light calculated values

© Copyright Ian D. Romanick 2008

Illumination Function

$$\Psi_{diffuse} = K_d \sin(T, L)$$
$$\Psi_{specular} = K_s \left((T \cdot L)(T \cdot E) + \sin(T, L) \sin(T, E) \right)^p$$
$$\Psi_{hair} = \Psi_{diffuse} + \Psi_{specular}$$

- Why is sin used instead of the usual cos?

© Copyright Ian D. Romanick 2008

Illumination Function

$$\Psi_{diffuse} = K_d \sin(T, L)$$
$$\Psi_{specular} = K_s \left|(T \cdot L)(T \cdot E) + \sin(T, L)\sin(T, E)\right|^p$$
$$\Psi_{hair} = \Psi_{diffuse} + \Psi_{specular}$$

- Why is sin used instead of the usual cos?
  - A hair is an infinitesimal cylinder and has infinite normals
  - The tangent pointing along the length of the hair is used instead
  - N and T are 90° out of phase, so cos(N, L) = sin(T, L)

© Copyright Ian D. Romanick 2008

Illumination Function

$$\Psi_{diffuse} = K_d \sin(T, L)$$
$$\Psi_{specular} = K_s \left((T \cdot L)(T \cdot E) + \sin(T, L) \sin(T, E) \right)^p$$
$$\Psi_{hair} = \Psi_{diffuse} + \Psi_{specular}$$

- Why is sin used instead of the usual cos?
  - A hair is an infinitesimal cylinder and has infinite normals
  - The tangent pointing along the length of the hair is used instead
  - N and T are 90° out of phase, so cos(N, L) = sin(T, L)
- How can we calculate sin(T, L)?

© Copyright Ian D. Romanick 2008

Illumination Function

$$\Psi_{diffuse} = K_d \sin(T, L)$$
$$\Psi_{specular} = K_s \left((T \cdot L)(T \cdot E) + \sin(T, L) \sin(T, E) \right)^p$$
$$\Psi_{hair} = \Psi_{diffuse} + \Psi_{specular}$$

- Why is sin used instead of the usual cos?
  - A hair is an infinitesimal cylinder and has infinite normals
  - The tangent pointing along the length of the hair is used instead
  - N and T are 90° out of phase, so cos(N, L) = sin(T, L)
- How can we calculate sin(T, L)?

$$\frac{a \times b}{|a||b|} = \sin\theta\, \hat{n} \rightarrow \left| \frac{a \times b}{|a||b|} \right| = \sin\theta$$

© Copyright Ian D. Romanick 2008

Illumination Function

$$\Psi_{diffuse} = K_d \sin(T, L)$$
$$\Psi_{specular} = K_s\left((T \cdot L)(T \cdot E) + \sin(T, L)\sin(T, E)\right)^p$$
$$\Psi_{hair} = \Psi_{diffuse} + \Psi_{specular}$$

- What's the problem here?

© Copyright Ian D. Romanick 2008

Illumination Function

$$\Psi_{diffuse} = K_d \sin(T, L)$$

$$\Psi_{specular} = K_s \left((T \cdot L)(T \cdot E) + \sin(T, L) \sin(T, E) \right)^p$$

$$\Psi_{hair} = \Psi_{diffuse} + \Psi_{specular}$$

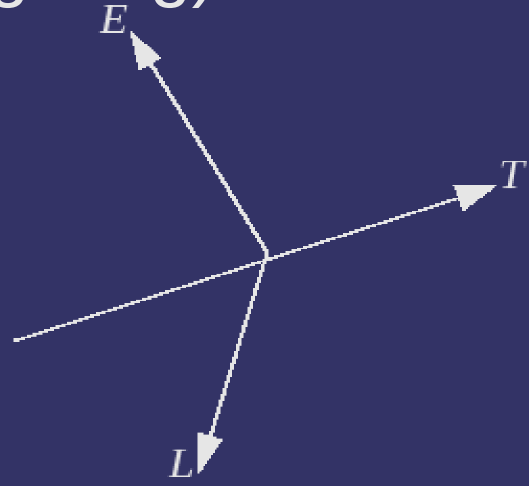
- What's the problem here?
  - Lacks directionality – hairs are fully lit even if L is opposite E
  - Fix this by adding some new attenuation factors

© Copyright Ian D. Romanick 2008

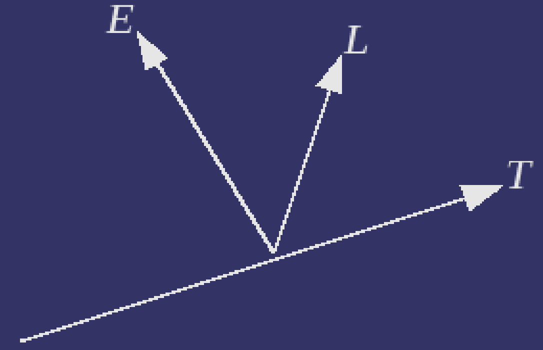
Relative Directionality

$$\kappa = \frac{(T \times L)(T \times E)}{|T \times L||T \times E|}$$

- $\kappa > 0$ when L and E are on the same side of the hair (frontlighting)
- $\kappa < 0$ when L and E are on opposite sides of the hair (backlighting)



Back lighting



Front lighting

© Copyright Ian D. Romanick 2008

Directional Attenuation Factor

$$f_{dir} = \frac{1+\kappa}{2}\rho_{reflect} + \frac{1-\kappa}{2}\rho_{transmit}$$

- ρ_{reflect} and ρ_{transmit} are parameters of the hair on the range [0, 1]
- White and gray hairs have ρ_{reflect} and ρ_{transmit} equal or nearly equal
- Colored hairs have $\rho_{\text{reflect}} > \rho_{\text{transmit}}$

© Copyright Ian D. Romanick 2008

Directional Attenuation Factor

$$f_{dir} = \frac{1+\kappa}{2}\rho_{reflect} + \frac{1-\kappa}{2}\rho_{transmit}$$

- ρ_{reflect} and ρ_{transmit} are parameters of the hair on the range [0, 1]
- White and gray hairs have ρ_{reflect} and ρ_{transmit} equal or nearly equal
- Colored hairs have $\rho_{\text{reflect}} > \rho_{\text{transmit}}$
- Unless you're a kitten...

© Copyright Ian D. Romanick 2008

Self-Shadowing

- Controlled by a second attenuation factor and 3 new parameters:

$$f_{surface} = 1 + \rho_{surface}\left(smoothstep(N \cdot L, \theta_{min}, \theta_{max}) - 1\right)$$

  - $\rho_{surface}$ controls the amount of self-shadowing
  - θ_{min} is the minimum angle where shadowing occurs
  - θ_{max} is the angle beyond which there is total occlusion

© Copyright Ian D. Romanick 2008

Fur Opacity

- How much of the surface *below* the fur can be seen through the fur

$$\alpha_f = 1 - \frac{1}{e^{D A_h g(E,T,N)}}$$

$$g(E,T,N) = \frac{\sin(E,T)}{E \cdot N}$$

$$A_h = l_{hair}(r_{base} + r_{top})/2$$

  - D is the local hair density
  - A_h is the projection of the surface area of a hair onto the view plane

© Copyright Ian D. Romanick 2008

Putting it all together

- Put the attenuation factors together with the opacity and skin color:

$$\Psi_{hair} = f_{dir} f_{surface} (\Psi_{diffuse} + \Psi_{specular})$$

$$\lambda_{skin} = K_{light} (1 - \alpha_f) \Psi_{skin}$$

$$\lambda_{hair} = K_{light} \left(1 - \frac{\alpha_f}{2}\right) \Psi_{hair}$$

$$f = \alpha_f \lambda_{hair} + (1 - \alpha_f) \lambda_{skin}$$

  – Ψ_{skin} is calculated by some other means

© Copyright Ian D. Romanick 2008

Break

© Copyright Ian D. Romanick 2008

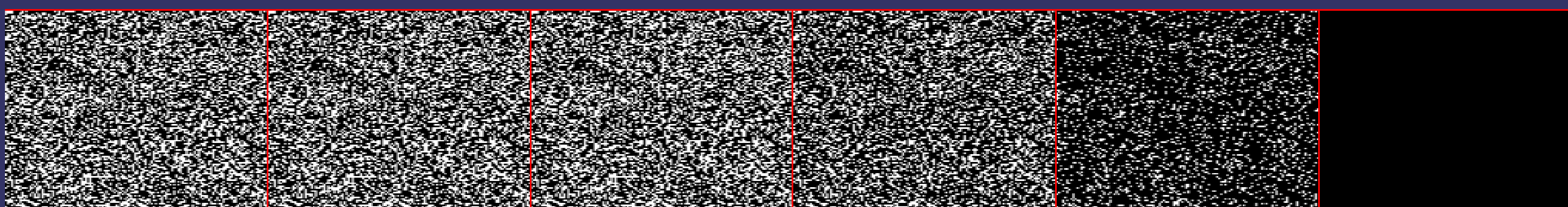
Volumetric Fur

- Close-up, fur appears as a volumetric effect
- Kajika and Kay presented an algorithm at SIGGRAPH '89 implementing fur via 3D textures
  - Volumetric textures are *very* memory intensive
  - Kajika and Kay's model involves several computationally expensive steps
- Not practical for real-time
  - There has to be a different way!

© Copyright Ian D. Romanick 2008

Shells and Fins

- Instead of a 3D texture, fur can be implemented with a “stack” of 2D textures
  - Each layer in the stack represents the fur at a different depth



  - Draw each layer in a progressively larger “shell” around the original object geometry

© Copyright Ian D. Romanick 2008

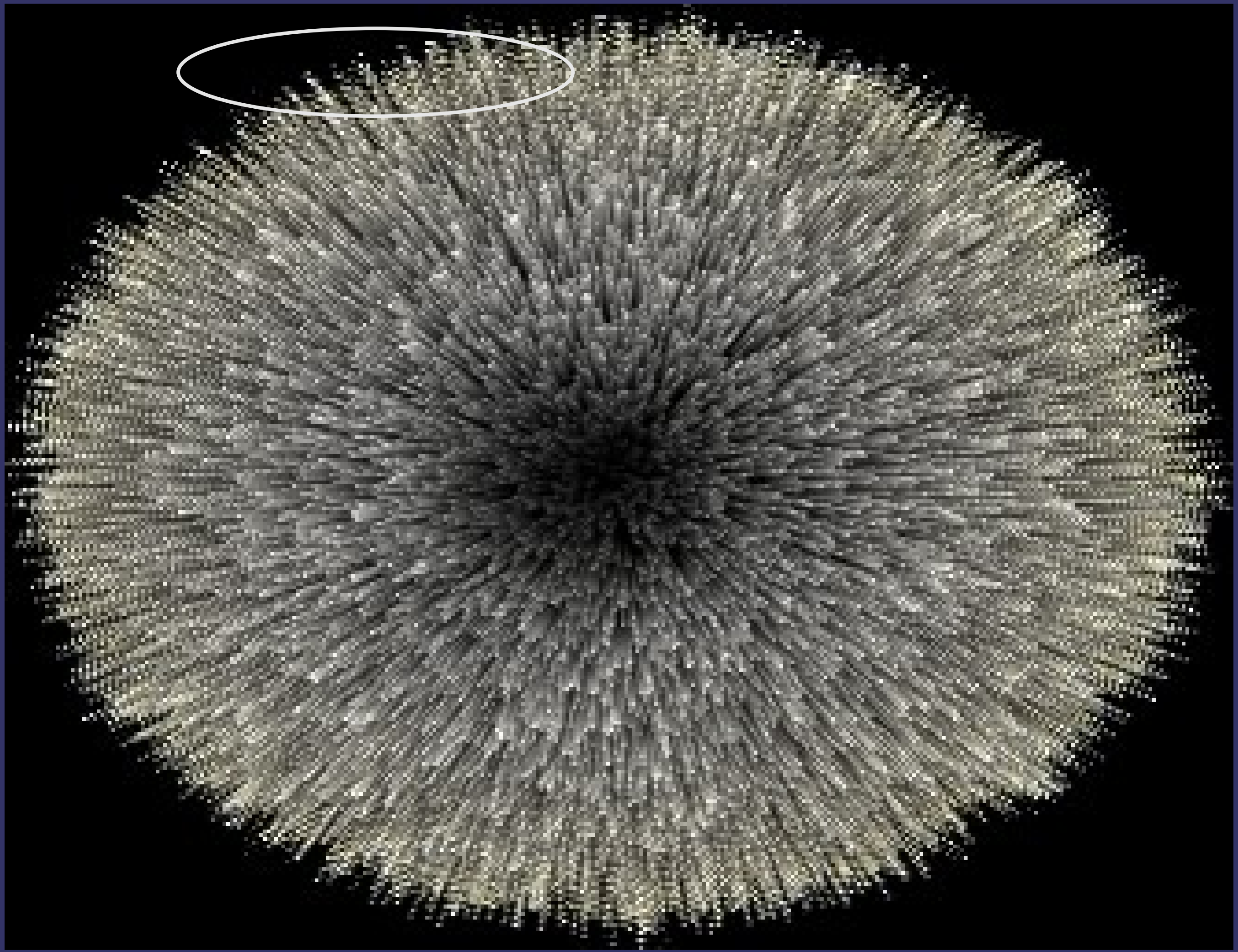
Shells and Fins

- Drawing loop:
  - Draw base object with inner-most (call it level 0) fur texture
    - Disable alpha blending
    - Enable z-testing
    - Enable z-writing
  - Draw base geometry moved out some small step along the normals
    - Enable alpha blending
    - Enable z-testing
    - Disable z-writing

© Copyright Ian D. Romanick 2008

Shells and Fins

- *But* this looks bad along the silhouette



© Copyright Ian D. Romanick 2008

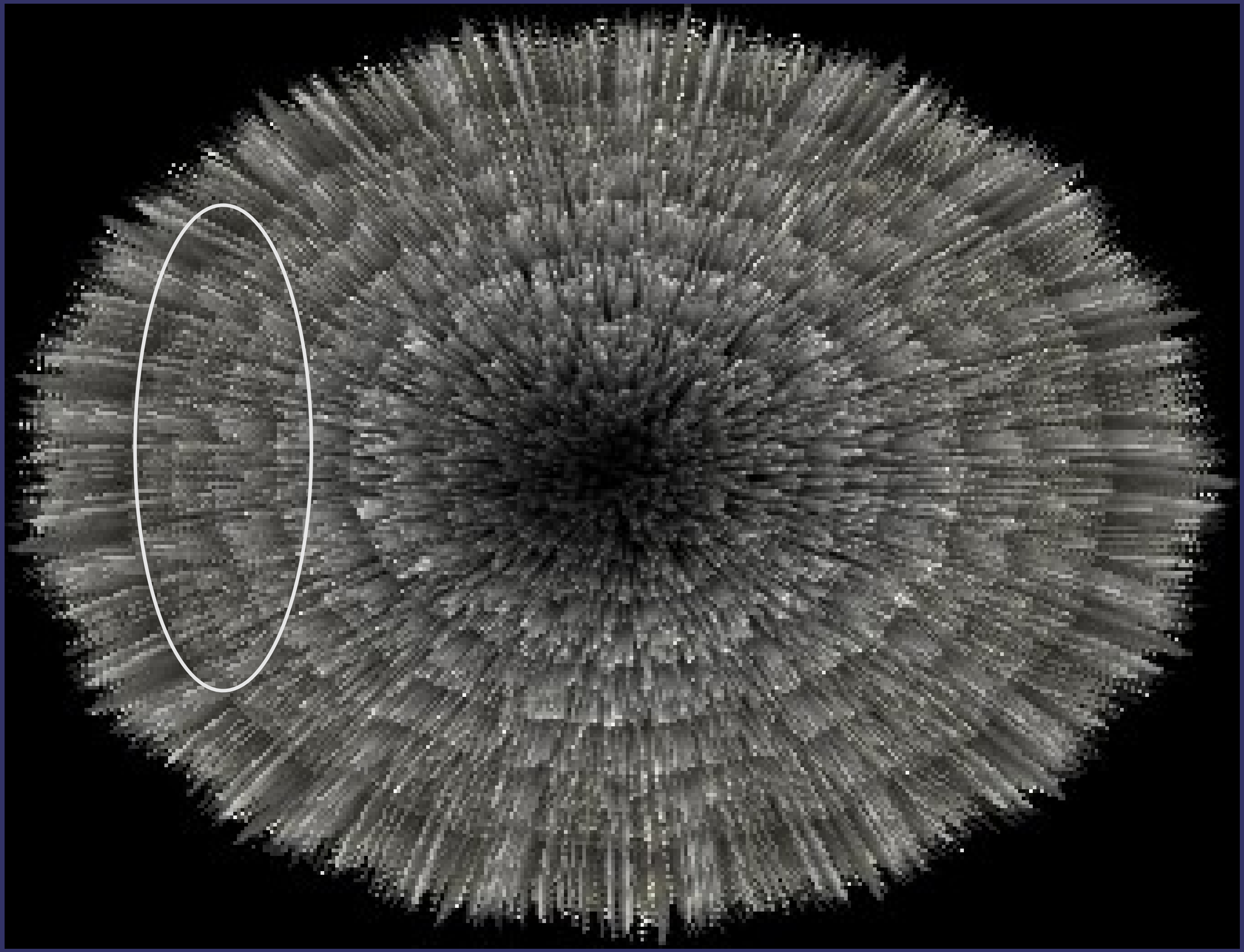
Shells and Fins

- Add fin geometry to each polygon
  - Create fin textures to look like side-on view of fur
  - Draw fin after drawing all shells
    - Enable alpha blending
    - Enable z-testing
    - Disable z-writing
- Generate fin geometry in the vertex shader:
  - Draw each vertex *twice*
    - Once with w = 0
    - Once with w = 1
  - Use the w value to determine whether or not to extrude the vertex in the normal direction
  - Draw the vertices as a quad in the order 0, 1, 1, 0

© Copyright Ian D. Romanick 2008

Shells and Fins

- *But* this looks bad in non-silhouette areas



© Copyright Ian D. Romanick 2008

Shells and Fins

- Gradually blend in fins as they approach the silhouette

$$\alpha_{fin} = max\left(0, 2\left|\cos\left(V, N_{fin}\right)\right| - 1\right)$$

- We don't really have a fin normal...what to do?

© Copyright Ian D. Romanick 2008

Shells and Fins

- Gradually blend in fins as they approach the silhouette

$$\alpha_{fin} = max(0, 2|\cos(V, N_{fin})| - 1)$$

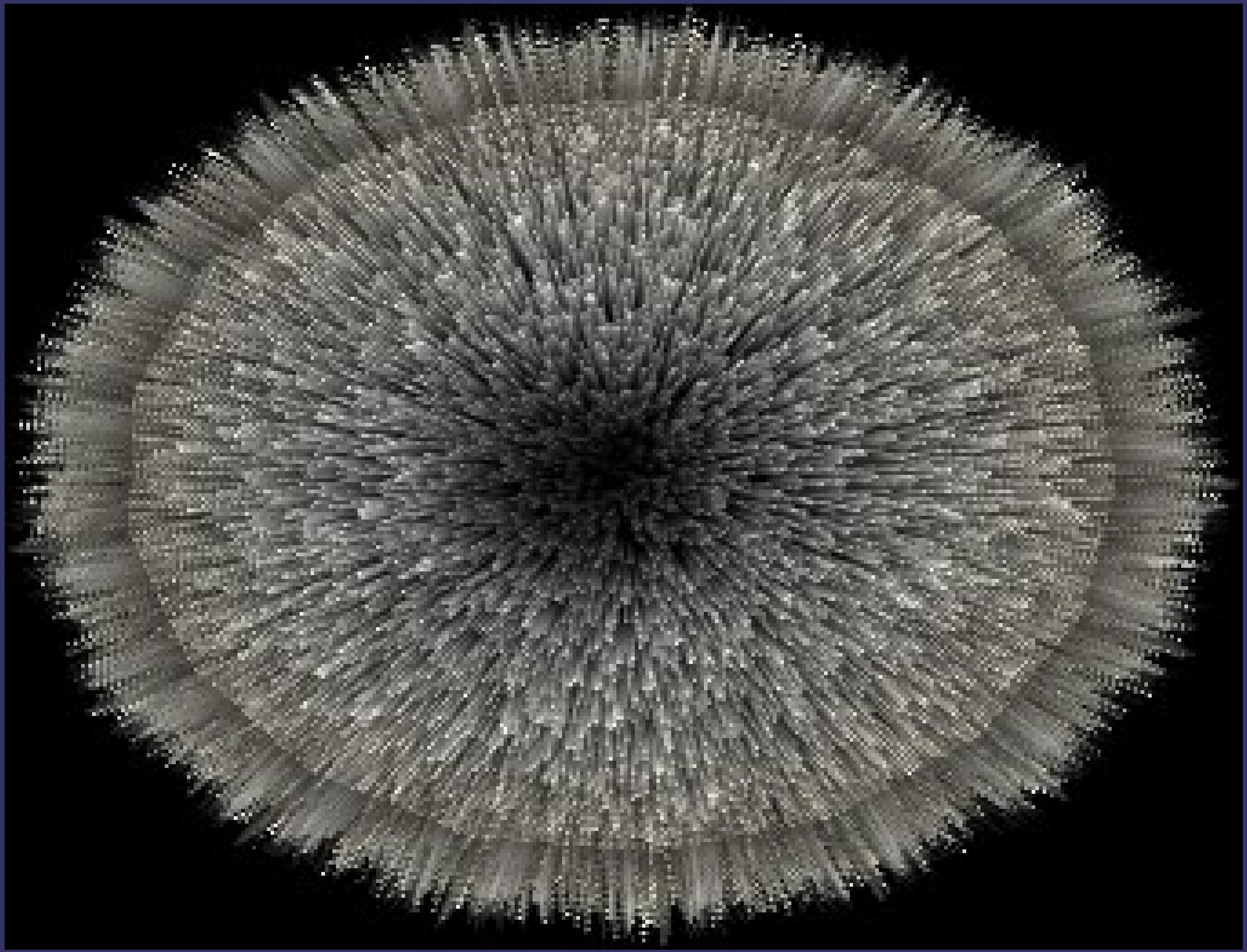
- We don't really have a fin normal...what to do?
  - The surface's normal is the fin's tangent

$$\alpha_{fin} = max(0, 2|\sin(V, N_{surface})| - 1)$$

© Copyright Ian D. Romanick 2008

Shells and Fins

- Alpha blended fins



© Copyright Ian D. Romanick 2008

Lighting Shells and Fins

- Use the surface normal as the direction of the hair

$$K = K_d \sin(N_{surface}, L)^{P_d} + K_s \sin(N_{surface}, H)^{P_s}$$

  - P_d and P_s are diffuse and specular exponents
  - Similar to Goldman's fakefur lighting model
- A little trig-identity love gets us:

$$K = K_d (1 - \cos(N_{surface}, L)^{P_d/2}) + K_s (1 - \cos(N_{surface}, H)^{P_s/2})$$

$$K = K_d (1 - (N_{surface} \cdot L)^{P_d/2}) + K_s (1 - (N_{surface} \cdot H)^{P_s/2})$$

© Copyright Ian D. Romanick 2008

Lighting Shells and Fins

- No shadowing happens!
  - Fur near the skin is occluded by the fur above it
  - Add a shadowing term to falloff to a minimum value linearly with the distance from the outermost shell

$$S = \frac{D\left(1 - S_{min}\right)}{D_{max}} + S_{min}$$

  - D is the current shell distance
    - $D = 0$ is the shell closest to the skin
  - D_{max} is the total number of shells
  - S_{min} is the minimum amount of light reaching the bottom layer

© Copyright Ian D. Romanick 2008

References

Sheppard, G. *Real-Time Rendering of Fur.* Honors Thesis, Univ. of Sheffield. 2004. http://www.gamasutra.com/education/theses/20051028/sheppard_01.shtml

Thorough overview of the various real-time fur methods.

Tariq, S. *Fur (using Shells and Fins).* Nvidia White Paper, Number WP-03021-001_v01. February 2007. http://developer.download.nvidia.com/whitepapers/2007/SDK10/FurShellsAndF

This article focuses on optimizing shells-and-fins using Shader Model 4 features that are currently only supported in OpenGL on GeForce8.

Kajiya, J. T. and Kay, T. L. 1989. Rendering fur with three dimensional textures. *SIGGRAPH Comput. Graph.* 23, 3 (Jul. 1989), 271-280. http://www.icg.tu-graz.ac.at/courses/lv710.087/kajiyahair.pdf

Lake, A. and Kuah, K.. *Real-Time Fur Rendering For Short Haired Creatures*. 2006. http://softwarecommunity.intel.com/articles/eng/2597.htm

Morris, N. CS6610 Final Project. December 2005. http://www.cs.utah.edu/classes/cs5610/projects-2005/morris/

© Copyright Ian D. Romanick 2008

Break

© Copyright Ian D. Romanick 2008

Terminology – codimension

- Given an object of dimension n in a k dimensional space with $k > n$, the *codimension*, c, is equal to n-k
  - For a surface in 3-space, n is 2 and k is 3
    - When $c = 1$, we can trivially assign a normal to the object

© Copyright Ian D. Romanick 2008

Terminology – codimension

- Given an object of dimension n in a k dimensional space with $k > n$, the *codimension*, c, is equal to n-k
  - For a surface in 3-space, n is 2 and k is 3
    - When $c = 1$, we can trivially assign a normal to the object
  - For a line in 3-space, $n = 1$, $k = 3$, and $c = 2$

© Copyright Ian D. Romanick 2008

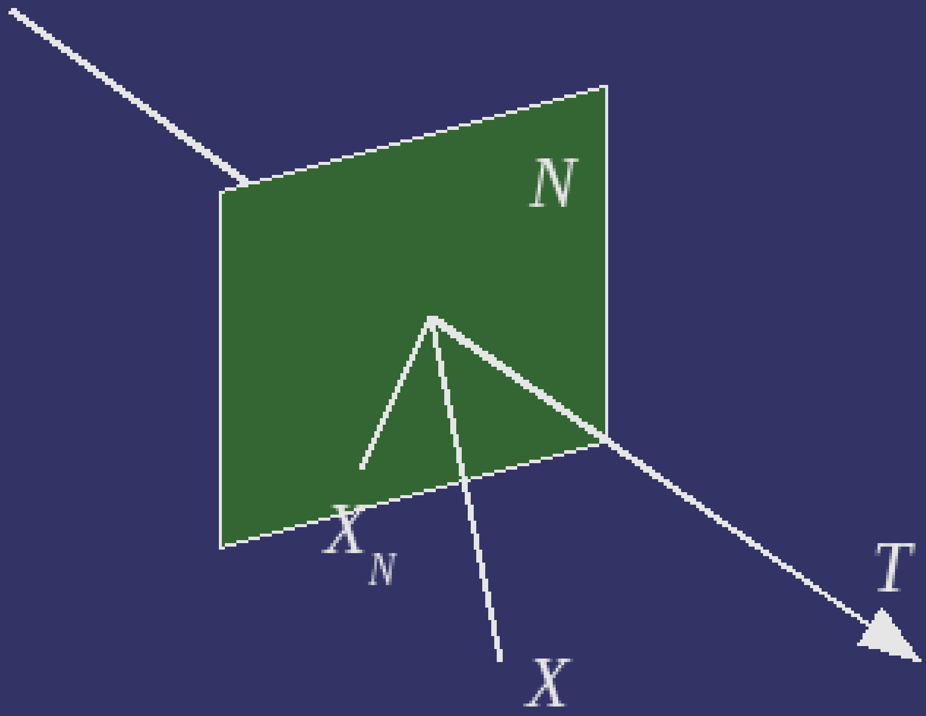
Terminology – vector spaces

- T is the tangent-space at some point on the object
  - Vector space tangent to the point on the object
  - Has dimension k (same as the object)
- N is the normal-space at some point on the object
  - Vector space orthogonal to T
  - Has dimension c (codimension of the object)

© Copyright Ian D. Romanick 2008

Terminology – vectors

- X_N is the projection of vector X onto $\boldsymbol{N}$
- X_T is the projection of vector X onto $\boldsymbol{T}$



© Copyright Ian D. Romanick 2008

Diffuse Reflection

- Applying this terminology, diffuse reflection is calculated as:

$$I_{diffuse} = K_d \cos(L, L_N)$$

© Copyright Ian D. Romanick 2008

Diffuse Reflection

- Applying this terminology, diffuse reflection is calculated as:

$$I_{diffuse} = K_d \cos(L, L_N)$$

- Since N and T are orthogonal, we can rewrite this as:

$$I_{diffuse} = K_d \sin(L, L_T)$$

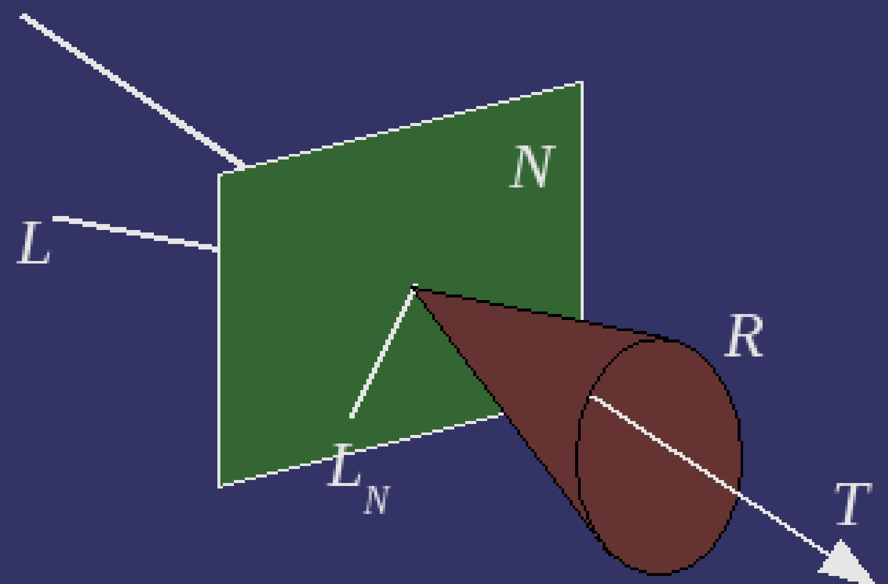
© Copyright Ian D. Romanick 2008

Specular Reflection

- Specular reflection is generally calculated as:

$$R = N - 2(N \cdot L)L$$
$$I_{specular} = k_s I_{light} \cos(V, R)$$

- If $c > 1$, there are infinite N vectors, so there are infinite possible R vectors



© Copyright Ian D. Romanick 2008

Fermat's Principle Saves the Day

- Fermat's principle says that light travels on the shortest length path
  - This means that L, L_N, and R are coplanar
  - Skipping the derivation, this means that $R_N = L_N$
  - Skipping more derivation, we can calculate cos(V, R) as:

$$V \cdot R = V_T \cdot L_T - |V_N||L_N|$$

© Copyright Ian D. Romanick 2008

Inherited Self-Shadowing

- When c = 1, the object has at most 2 sides
  - One side of the surface "self-shadows" the other, and we get that calculation for free from $N \bullet L$

© Copyright Ian D. Romanick 2008

Inherited Self-Shadowing

- When $c = 1$, the object has at most 2 sides
  - One side of the surface “self-shadows” the other, and we get that calculation for free from $N \bullet L$
- Consider a surface with a 2D tangent space, $\boldsymbol{T}$, and a 1D vector field, $\boldsymbol{V}$
  - If $\boldsymbol{T}$ is used to calculate the illumination, $N \bullet L$ works
  - If $\boldsymbol{V}$ is used to calculate the illumination, there is no unique N to use

© Copyright Ian D. Romanick 2008

Inherited Self-Shadowing

- When c = 1, the object has at most 2 sides
  - One side of the surface "self-shadows" the other, and we get that calculation for free from $N \bullet L$
- Consider a surface with a 2D tangent space, $\boldsymbol{T}$, and a 1D vector field, $\boldsymbol{V}$
  - If $\boldsymbol{T}$ is used to calculate the illumination, $N \bullet L$ works
  - If $\boldsymbol{V}$ is used to calculate the illumination, there is no unique N to use
  - If $\boldsymbol{V}$ is used to calculate the illumination, it can *inherit* $N \bullet L$ from $\boldsymbol{T}$

$$I_{conditioned} = max(N \cdot L, 0)(I_{diffuse} + I_{specular})$$

© Copyright Ian D. Romanick 2008

Vector Field Shadowing

- This shadows the vector field from the surface
- If the vectors like outside the surface (e.g., fur) the vector field can obviously shadow itself and the surface
- Input light energy is attenuated by:

$$d = h / \sin(T, L)$$

$$I_{atten} = I_{source}(1 - \rho)^d$$

  - h is the distance from the surface
  - ρ is a property of the fur
    - The paper uses ρ = 0.02

© Copyright Ian D. Romanick 2008

References

Banks, D. C. 1994. Illumination in diverse codimensions. In *Proceedings of the 21st Annual Conference on Computer Graphics and interactive Techniques SIGGRAPH '94*. ACM, New York, NY, 327-334. http://lmi.bwh.harvard.edu/~banks/

© Copyright Ian D. Romanick 2008

Next week...

- Non-photorealistic rendering
  - Cel shading (cartoon rendering)
  - Silhouette edge rendering
  - Gooch style technical illustrations

© Copyright Ian D. Romanick 2008

Legal Statement

This work represents the view of the authors and does not necessarily represent the view of IBM or the Art Institute of Portland.

OpenGL is a trademark of Silicon Graphics, Inc. in the United States, other countries, or both.

Khronos and OpenGL ES are trademarks of the Khronos Group.

Other company, product, and service names may be trademarks or service marks of others.

© Copyright Ian D. Romanick 2008