VGP352 – Week 6

- Agenda:
  - Illuminating infinitesimal strands
    - Piles of math leading to the Banks BRDF
    - General “strand” model for anisotropic surfaces
  - Goldman's “fakefur”
  - Implementing BRDFs in real-time
  - Fins-and-shells for fur

Hair

- How do we calculate illumination for an infinitesimal strand or fiber?

Terminology – Codimension

- Definition:

  Given an object of dimension n in a k dimensional space, with $k > n$, the codimension, c, is equal to $k - n$

  – For a surface in 3-space, $n = 2$ and $k = 3$

    – When $c = 1$, we can trivially assign a normal to the object

Terminology – Codimension

- Definition:

  Given an object of dimension n in a k dimensional space, with $k > n$, the codimension, c, is equal to $k - n$

  - For a surface in 3-space, $n = 2$ and $k = 3$
    - When $c = 1$, we can trivially assign a normal to the object
  - For a line in 3-space, $n = 1$ and $k = 3$
    - When $c > 1$, things get a little weird...

Terminology – Codimension

- Another way to think of it: The normal has c degrees of freedom
  - For a plane in 3-space, the normal can point in one of two directions (up or down)
    - $k - n = c \Rightarrow 3 - 2 = 1$
  - It's only degree of freedom is its magnitude
    - If we restrict the space to normalized vectors, there are only two possible values

Terminology – Vector Spaces

- Definition:

  A vector space is a mathematical structure formed by a collection of vectors: objects that may be added together and multiplied ("scaled") by numbers, called scalars in this context.[1]

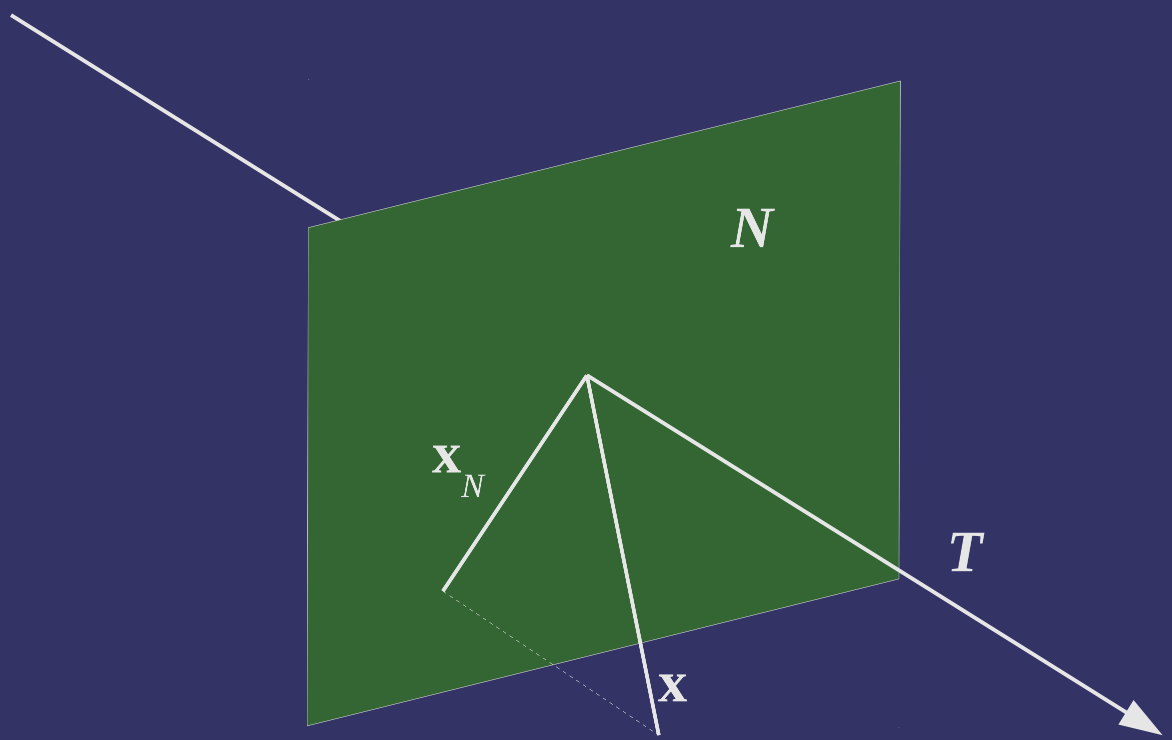
[1]From http://en.wikipedia.org/wiki/Vector_space

Terminology – Vector Spaces

- T is the tangent-space at some point on the object
  - Vector space tangent to the point on the object
    - Specifically, all of the possible tangent vectors at that location
  - Has dimension k (same as the object)
- N is the normal-space at some point on the object
  - Vector space orthogonal to T
    - Specifically, all of the possible normal vectors at that location
  - Has dimension c (codimension of the object)

Terminology – Vector Projections

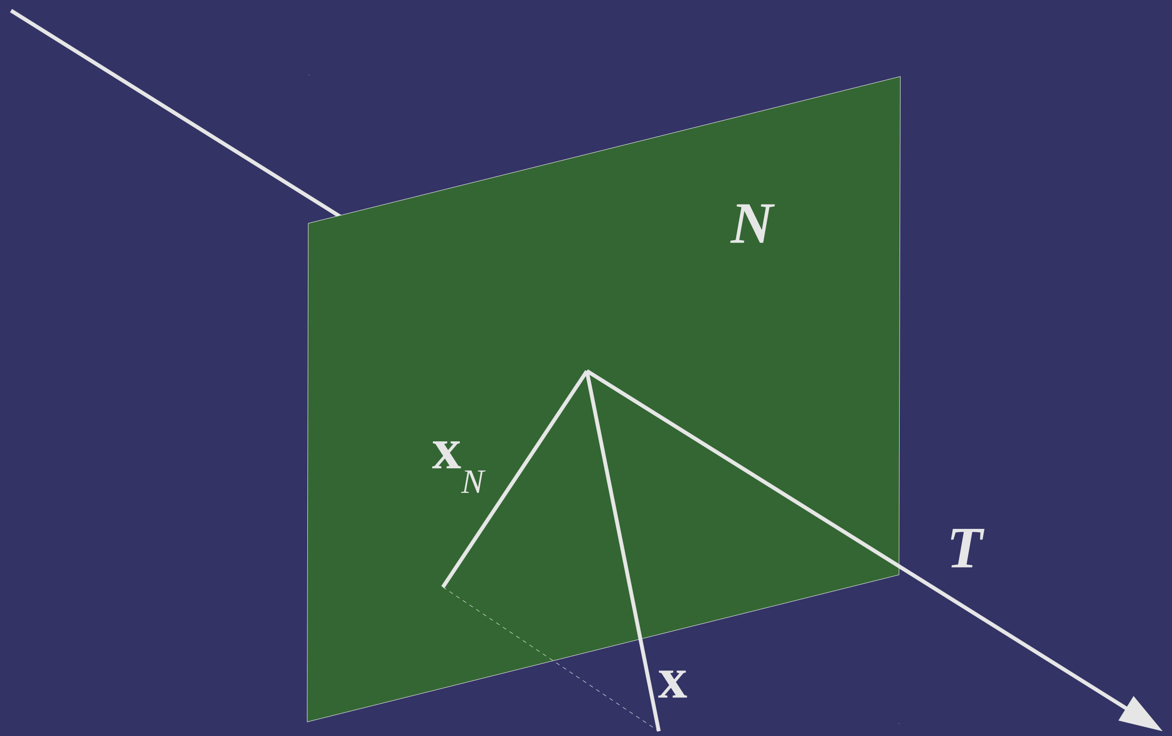
- $\mathbf{x}_N$ is the projection of vector $\mathbf{x}$ onto N
- $\mathbf{x}_T$ is the projection of vector $\mathbf{x}$ onto T



16-February-2010

Terminology – Vector Projections

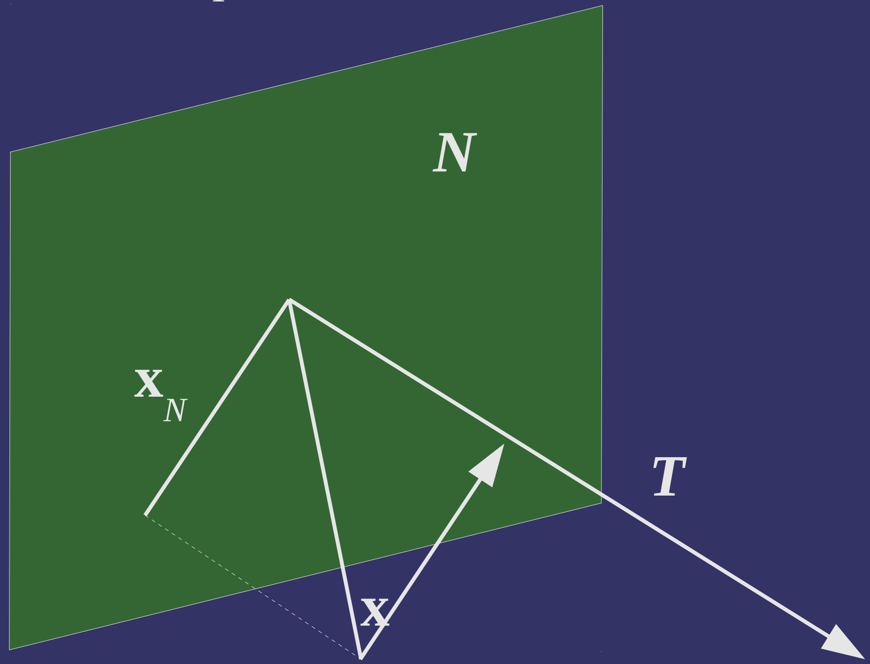
- How do we project a vector onto a plane?



16-February-2010

Terminology – Vector Projections

- How do we project a vector onto a plane?
  - Fun facts:
    - $\mathbf{x}_N = \mathbf{x} - \mathbf{x}_T$
    - $|\mathbf{x}_T| = \cos(\mathbf{x}, T) = \mathbf{x} \cdot T \rightarrow \mathbf{x}_T = (\mathbf{x} \cdot T)T$
    - $|\mathbf{x}_N| = \sin(\mathbf{x}, T)$



Diffuse Reflection

- Using this terminology, diffuse reflection can be calculated as:

$$\mathbf{i}_{\text{diffuse}} = \mathbf{k}_{\text{d}} \frac{\cos(\mathbf{l}, \mathbf{l}_{\text{N}})}{|\mathbf{l}|\,|\mathbf{l}_{\text{N}}|}$$

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

$$\mathbf{i}_{\text{diffuse}} = \mathbf{k}_{\text{d}} \frac{\sin(\mathbf{l}, \mathbf{l}_{\text{T}})}{|\mathbf{l}|\,|\mathbf{l}_{\text{T}}|}$$

Specular Reflection

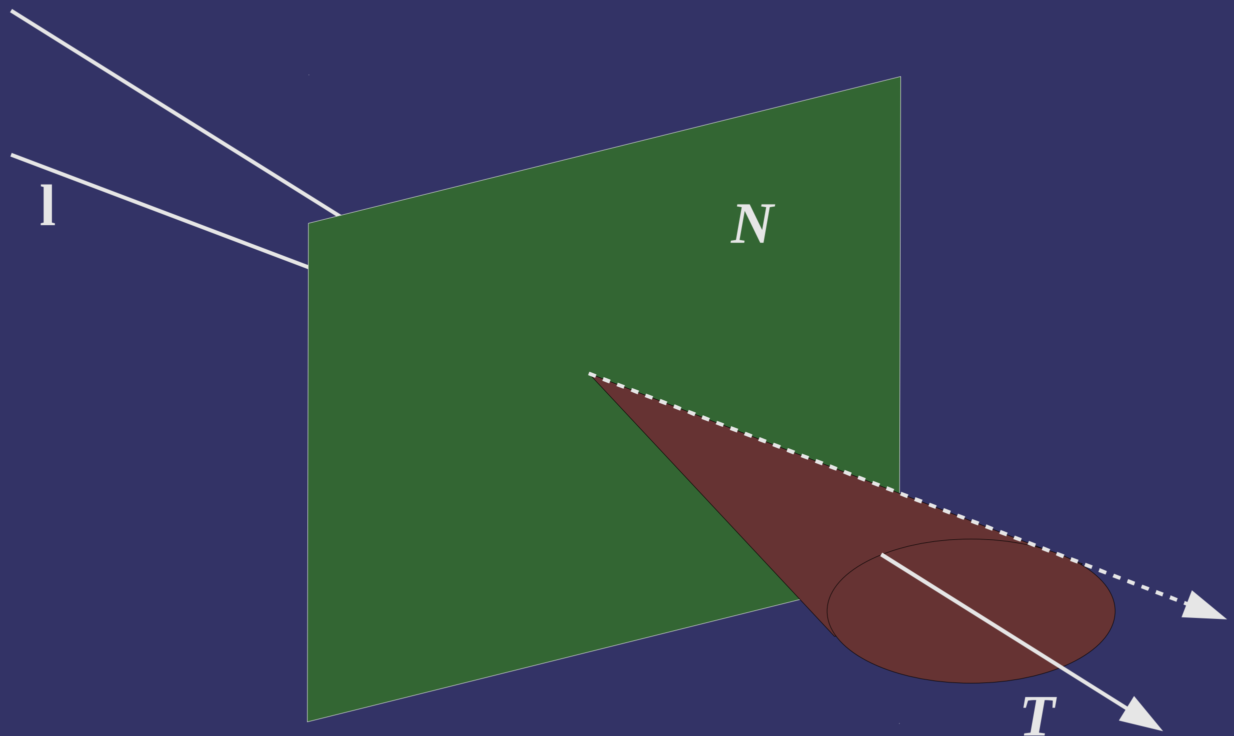
- Phong specular reflection:

$$\mathbf{r} = \mathbf{n} - 2(\mathbf{n} \cdot \mathbf{l})\mathbf{l}$$

$$\mathbf{i}_{\text{specular}} = \mathbf{k}_s \mathbf{i}_{\text{light}} \cos(\mathbf{v}, \mathbf{r})^s$$

Specular Reflection

- When $c > 1$, there are infinite possible $\mathbf{n}$ vectors, so there are infinite possible $\mathbf{r}$ vectors



Fermat's Principle

- Fermat's principle says that light travels on the shortest length path
  - This means that $\mathbf{l}$, $\mathbf{l}_N$, and $\mathbf{r}$ are coplanar
  - Skipping a bit of derivation, this means that $\mathbf{l}_N$ is equal to $\mathbf{r}_N$
  - Skipping a bit more derivation, this means that $\mathbf{v} \cdot \mathbf{r}$ can be calculated as:

$$\mathbf{v} \cdot \mathbf{r} = \mathbf{v}_T \cdot \mathbf{l}_T - |\mathbf{v}_N| |\mathbf{l}_N|$$

Specular Reflection

- $\mathbf{v} \cdot \mathbf{r}$ can be calculated as:

$$\mathbf{v} \cdot \mathbf{r} = \mathbf{v}_T \cdot \mathbf{l}_T - |\mathbf{v}_N||\mathbf{l}_N|$$

  - But we don't initially know $\mathbf{v}_T$, $\mathbf{l}_T$, $\mathbf{v}_N$, or $\mathbf{l}_N$

Specular Reflection

- $\mathbf{v} \cdot \mathbf{r}$ can be calculated as:

$$\mathbf{v} \cdot \mathbf{r} = \mathbf{v}_T \cdot \mathbf{l}_T - |\mathbf{v}_N||\mathbf{l}_N|$$

  - But we don't initially know $\mathbf{v}_T$, $\mathbf{l}_T$, $\mathbf{v}_N$, or $\mathbf{l}_N$

$$\begin{aligned} \mathbf{v} \cdot \mathbf{r} &= \mathbf{v}_T \cdot \mathbf{l}_T - |\mathbf{v}_N||\mathbf{l}_N| \\ &= \mathbf{v}_T \cdot \mathbf{l}_T - \sin(\mathbf{v}, \mathbf{t}) \sin(\mathbf{l}, \mathbf{t}) \end{aligned}$$

Specular Reflection

- $\mathbf{v} \cdot \mathbf{r}$ can be calculated as:

$$\mathbf{v} \cdot \mathbf{r} = \mathbf{v}_T \cdot \mathbf{l}_T - |\mathbf{v}_N||\mathbf{l}_N|$$

  - But we don't initially know $\mathbf{v}_T$, $\mathbf{l}_T$, $\mathbf{v}_N$, or $\mathbf{l}_N$

$$\begin{aligned}
\mathbf{v} \cdot \mathbf{r} &= \mathbf{v}_T \cdot \mathbf{l}_T - |\mathbf{v}_N||\mathbf{l}_N| \\
&= \mathbf{v}_T \cdot \mathbf{l}_T - \sin(\mathbf{v}, \mathbf{t}) \sin(\mathbf{l}, \mathbf{t}) \\
&= \left((\mathbf{v} \cdot \mathbf{t})\mathbf{t}\right) \cdot \left((\mathbf{l} \cdot \mathbf{t})\mathbf{t}\right) - \sin(\mathbf{v}, \mathbf{t}) \sin(\mathbf{l}, \mathbf{t})
\end{aligned}$$

Specular Reflection

- $\mathbf{v} \cdot \mathbf{r}$ can be calculated as:

$$\mathbf{v}\cdot\mathbf{r} = \mathbf{v}_T \cdot \mathbf{l}_T - |\mathbf{v}_N||\mathbf{l}_N|$$

  - But we don't initially know $\mathbf{v}_T$, $\mathbf{l}_T$, $\mathbf{v}_N$, or $\mathbf{l}_N$

$$\begin{aligned}
\mathbf{v}\cdot\mathbf{r} &= \mathbf{v}_T \cdot \mathbf{l}_T - |\mathbf{v}_N||\mathbf{l}_N| \\
&= \mathbf{v}_T \cdot \mathbf{l}_T - \sin(\mathbf{v},\mathbf{t})\sin(\mathbf{l},\mathbf{t}) \\
&= \left((\mathbf{v}\cdot\mathbf{t})\mathbf{t}\right)\cdot\left((\mathbf{l}\cdot\mathbf{t})\mathbf{t}\right) - \sin(\mathbf{v},\mathbf{t})\sin(\mathbf{l},\mathbf{t}) \\
&= \left((\mathbf{v}\cdot\mathbf{t})(\mathbf{l}\cdot\mathbf{t})(\mathbf{t}\cdot\mathbf{t})\right) - \sin(\mathbf{v},\mathbf{t})\sin(\mathbf{l},\mathbf{t})
\end{aligned}$$

Specular Reflection

- $\mathbf{v} \cdot \mathbf{r}$ can be calculated as:

$$\mathbf{v}\cdot\mathbf{r} = \mathbf{v}_T\cdot\mathbf{l}_T - |\mathbf{v}_N||\mathbf{l}_N|$$

  – But we don't initially know $\mathbf{v}_T$, $\mathbf{l}_T$, $\mathbf{v}_N$, or $\mathbf{l}_N$

$$\begin{aligned}
\mathbf{v}\cdot\mathbf{r} &= \mathbf{v}_T\cdot\mathbf{l}_T - |\mathbf{v}_N||\mathbf{l}_N| \\
&= \mathbf{v}_T\cdot\mathbf{l}_T - \sin(\mathbf{v},\mathbf{t})\sin(\mathbf{l},\mathbf{t}) \\
&= \left((\mathbf{v}\cdot\mathbf{t})\mathbf{t}\right)\cdot\left((\mathbf{l}\cdot\mathbf{t})\mathbf{t}\right) - \sin(\mathbf{v},\mathbf{t})\sin(\mathbf{l},\mathbf{t}) \\
&= \left((\mathbf{v}\cdot\mathbf{t})(\mathbf{l}\cdot\mathbf{t})(\mathbf{t}\cdot\mathbf{t})\right) - \sin(\mathbf{v},\mathbf{t})\sin(\mathbf{l},\mathbf{t}) \\
&= (\mathbf{v}\cdot\mathbf{t})(\mathbf{l}\cdot\mathbf{t}) - \sin(\mathbf{v},\mathbf{t})\sin(\mathbf{l},\mathbf{t})
\end{aligned}$$

Specular Reflection

- $\mathbf{v} \cdot \mathbf{r}$ can be calculated as:

$$\mathbf{v}\cdot\mathbf{r} = \mathbf{v}_T\cdot\mathbf{l}_T - |\mathbf{v}_N||\mathbf{l}_N|$$

  – But we don't initially know $\mathbf{v}_T$, $\mathbf{l}_T$, $\mathbf{v}_N$, or $\mathbf{l}_N$

$$\begin{aligned}
\mathbf{v}\cdot\mathbf{r} &= \mathbf{v}_T\cdot\mathbf{l}_T - |\mathbf{v}_N||\mathbf{l}_N| \\
&= \mathbf{v}_T\cdot\mathbf{l}_T - \sin(\mathbf{v},\mathbf{t})\sin(\mathbf{l},\mathbf{t}) \\
&= \left((\mathbf{v}\cdot\mathbf{t})\mathbf{t}\right)\cdot\left((\mathbf{l}\cdot\mathbf{t})\mathbf{t}\right) - \sin(\mathbf{v},\mathbf{t})\sin(\mathbf{l},\mathbf{t}) \\
&= \left((\mathbf{v}\cdot\mathbf{t})(\mathbf{l}\cdot\mathbf{t})(\mathbf{t}\cdot\mathbf{t})\right) - \sin(\mathbf{v},\mathbf{t})\sin(\mathbf{l},\mathbf{t}) \\
&= (\mathbf{v}\cdot\mathbf{t})(\mathbf{l}\cdot\mathbf{t}) - \sin(\mathbf{v},\mathbf{t})\sin(\mathbf{l},\mathbf{t}) \\
&= (\mathbf{v}\cdot\mathbf{t})(\mathbf{l}\cdot\mathbf{t}) - \left(\sqrt{1-\cos(\mathbf{v},\mathbf{t})^2}\sqrt{1-\cos(\mathbf{l},\mathbf{t})^2}\right)
\end{aligned}$$

Specular Reflection

- $\mathbf{v} \cdot \mathbf{r}$ can be calculated as:

$$\mathbf{v}\cdot\mathbf{r} = \mathbf{v}_T\cdot\mathbf{l}_T - |\mathbf{v}_N||\mathbf{l}_N|$$

  – But we don't initially know $\mathbf{v}_T$, $\mathbf{l}_T$, $\mathbf{v}_N$, or $\mathbf{l}_N$

$$\begin{aligned}
\mathbf{v}\cdot\mathbf{r} &= \mathbf{v}_T\cdot\mathbf{l}_T - |\mathbf{v}_N||\mathbf{l}_N| \\
&= \mathbf{v}_T\cdot\mathbf{l}_T - \sin(\mathbf{v},\mathbf{t})\sin(\mathbf{l},\mathbf{t}) \\
&= \left((\mathbf{v}\cdot\mathbf{t})\mathbf{t}\right)\cdot\left((\mathbf{l}\cdot\mathbf{t})\mathbf{t}\right) - \sin(\mathbf{v},\mathbf{t})\sin(\mathbf{l},\mathbf{t}) \\
&= \left((\mathbf{v}\cdot\mathbf{t})(\mathbf{l}\cdot\mathbf{t})(\mathbf{t}\cdot\mathbf{t})\right) - \sin(\mathbf{v},\mathbf{t})\sin(\mathbf{l},\mathbf{t}) \\
&= (\mathbf{v}\cdot\mathbf{t})(\mathbf{l}\cdot\mathbf{t}) - \sin(\mathbf{v},\mathbf{t})\sin(\mathbf{l},\mathbf{t}) \\
&= (\mathbf{v}\cdot\mathbf{t})(\mathbf{l}\cdot\mathbf{t}) - \left(\sqrt{1-\cos(\mathbf{v},\mathbf{t})^2}\sqrt{1-\cos(\mathbf{l},\mathbf{t})^2}\right) \\
&= (\mathbf{v}\cdot\mathbf{t})(\mathbf{l}\cdot\mathbf{t}) - \left(\sqrt{1-(\mathbf{v}\cdot\mathbf{t})^2}\sqrt{1-(\mathbf{l}\cdot\mathbf{t})^2}\right)
\end{aligned}$$

Inherited Self-Shadowing

- When c = 1, the object has at most 2 sides
  - One side of the surface “self-shadows” the other
  - We get that calculation for free from $\mathbf{n} \cdot \mathbf{l}$

Inherited Self-Shadowing

- When c = 1, the object has at most 2 sides
  - One side of the surface “self-shadows” the other
  - We get that calculation for free from $\mathbf{n} \cdot \mathbf{l}$
- Consider a surface with a 2D tangent space, T, and a 1D vector field, V

Think of bristles on a surface

  - If T is used to calculate the illumination, $\mathbf{n}_{surface} \cdot \mathbf{l}$ works
  - If V is used to calculate the illumination, there is no unique $\mathbf{n}$ to use

Inherited Self-Shadowing

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

$$\mathbf{i}_{conditioned} = \max(\mathbf{n} \cdot \mathbf{l}, 0)(\mathbf{i}_{diffuse} + \mathbf{i}_{specular})$$

Vector Field Shadowing

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

$$d = h / \sin(\mathbf{t}, \mathbf{l})$$
$$\mathbf{i}_{\text{atten}} = \mathbf{i}_{\text{source}} (1 - \rho)^d$$

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

Strand Based Anisotropic Lighting

- Why limit the use of this lighting model to individual strands?
  - We can treat many types of anisotropic surfaces as a collection of many strands... and apply the same lighting technique!

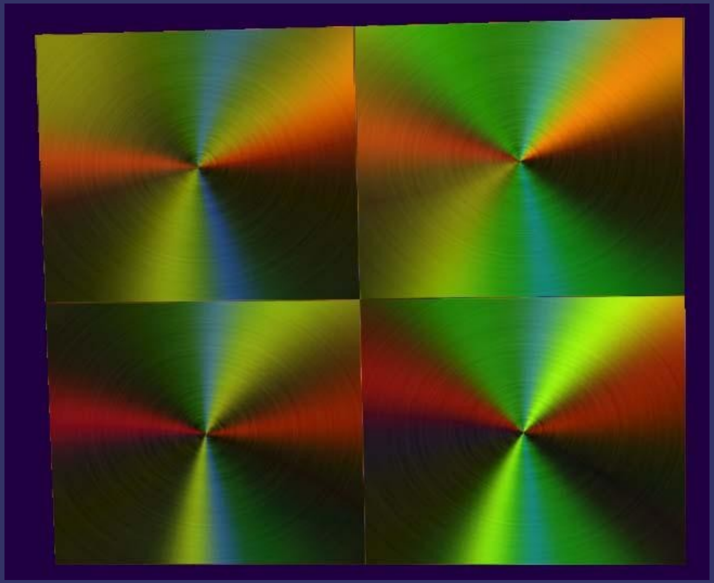


Image from ShaderX, Wordware Publishing, Inc.

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/

Isidoro, John and Brennan, Chris. "Per-Pixel Strand Based Anisotropic Lighting" in Engel, Wolfgang F. (editor) ShaderX, Wordware Publishing, Inc., May 2002. http://developer.amd.com/documentation/reading/pages/ShaderX.aspx

fakefur

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

fakefur

- 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

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

Illumination Function

$$\Psi_{\text{diffuse}} = \mathbf{k}_{\text{d}} \sin(\mathbf{t}, \mathbf{l})$$

$$\Psi_{\text{specular}} = \mathbf{k}_{\text{s}} \left((\mathbf{t} \cdot \mathbf{l})(\mathbf{t} \cdot \mathbf{v}) + \sin(\mathbf{t}, \mathbf{l}) \sin(\mathbf{t}, \mathbf{v}) \right)^{p}$$

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

- Why is sine used instead of cosine?

Illumination Function

$$\Psi_{\text{diffuse}} = \mathbf{k}_{\text{d}} \sin(\mathbf{t}, \mathbf{l})$$

$$\Psi_{\text{specular}} = \mathbf{k}_{\text{s}} \left((\mathbf{t} \cdot \mathbf{l})(\mathbf{t} \cdot \mathbf{v}) + \sin(\mathbf{t}, \mathbf{l}) \sin(\mathbf{t}, \mathbf{v}) \right)^{p}$$

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

- Why is sine used instead of cosine?
  - We treat the hair as having dimension = 1
    - There are infinite possible normals
    - There is only one tangent

This should look familiar!

Illumination Function

$$\Psi_{\text{diffuse}} = \mathbf{k}_{\text{d}} \sin(\mathbf{t}, \mathbf{l})$$
$$\Psi_{\text{specular}} = \mathbf{k}_{\text{s}} \left((\mathbf{t} \cdot \mathbf{l})(\mathbf{t} \cdot \mathbf{v}) + \sin(\mathbf{t}, \mathbf{l}) \sin(\mathbf{t}, \mathbf{v}) \right)^{p}$$
$$\Psi_{\text{hair}} = \Psi_{\text{diffuse}} + \Psi_{\text{specular}}$$

- What's wrong here?

Illumination Function

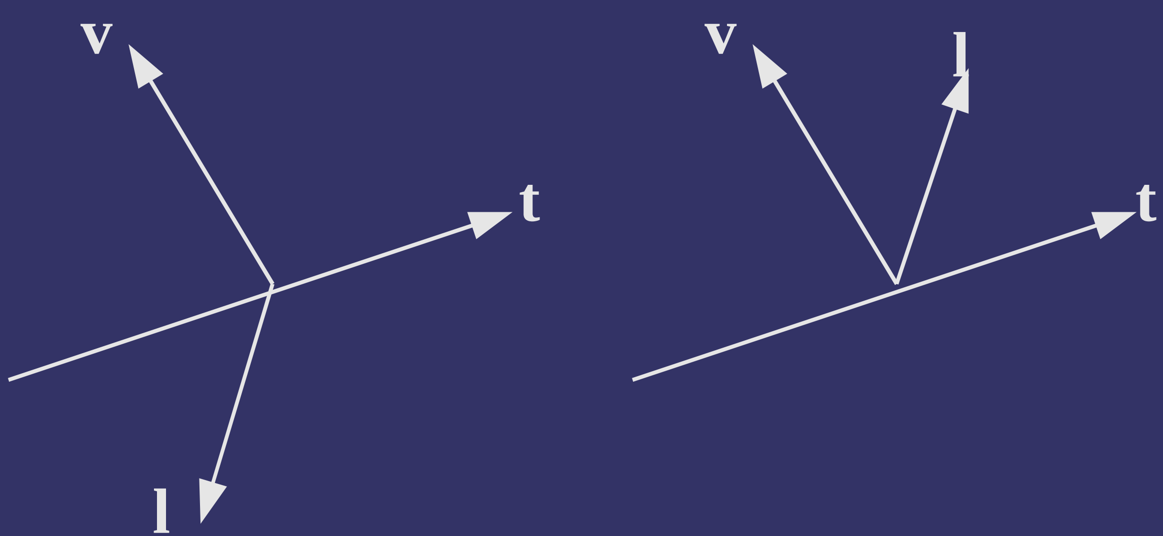
$$\Psi_{\text{diffuse}} = \mathbf{k}_{\text{d}} \sin(\mathbf{t}, \mathbf{l})$$
$$\Psi_{\text{specular}} = \mathbf{k}_{\text{s}} \left((\mathbf{t} \cdot \mathbf{l})(\mathbf{t} \cdot \mathbf{v}) + \sin(\mathbf{t}, \mathbf{l}) \sin(\mathbf{t}, \mathbf{v}) \right)^{p}$$
$$\Psi_{\text{hair}} = \Psi_{\text{diffuse}} + \Psi_{\text{specular}}$$

- What's wrong here?
  - Lacks directionality
    - Hairs are fully lit even if $\mathbf{l}$ is opposite $\mathbf{v}$
  - Fix this by adding some new attenuation factors

Relative Directionality

$$\kappa = \frac{(\boldsymbol{t} \times \boldsymbol{l}) \cdot (\boldsymbol{t} \times \boldsymbol{v})}{|\boldsymbol{t} \times \boldsymbol{l}||\boldsymbol{t} \times \boldsymbol{v}|}$$

- $\kappa > 0$ when $\mathbf{l}$ and $\mathbf{v}$ are on the same side of the hair (frontlighting)
- $\kappa < 0$ when $\mathbf{l}$ and $\mathbf{v}$ are on opposite sides of the hair (backlighting)



Back lighting | Front lighting

Directional Attenuation Factor

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

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

Directional Attenuation Factor

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

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

Self-Shadowing

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

$$f_{\text{surface}} = 1 + \rho_{\text{surface}} \left(\text{smoothstep}\left(\mathbf{n} \cdot \mathbf{l}, \theta_{\min}, \theta_{\max} \right) - 1 \right)$$

  - $\rho_{surface}$ 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

Fur Opacity

- How much of the surface *below* the fur can be seen through the fur?
  - Contributing factors:
    - Hair density: More hairs result in more occlusion
    - Hair size: Larger (thicker) hairs individually occlude more
    - Hair orientation: Hairs “laying down” occlude more than hairs on end
      - Orientation relative to both the viewer and the underlying surface are factors

Fur Opacity

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

$$
\begin{aligned}
\alpha_{\mathrm{f}} &= 1 - \frac{1}{e^{d\,a_{\mathrm{h}}\,\mathrm{g}(\mathbf{v},\mathbf{t},\mathbf{n})}} \\
\mathrm{g}(\mathbf{v},\mathbf{t},\mathbf{n}) &= \frac{\sin(\mathbf{v},\mathbf{t})}{\mathbf{v}\cdot\mathbf{n}} \\
a_{\mathrm{h}} &= l_{\mathrm{hair}}(r_{\mathrm{base}} + r_{\mathrm{top}})/2
\end{aligned}
$$

  - d is the local hair density
  - a_h is the projection of the surface area of a hair onto the view plane

Putting It All Together

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

$$\Psi_{\text{hair}} = f_{\text{dir}} f_{\text{surface}} \left(\Psi_{\text{diffuse}} + \Psi_{\text{specular}} \right)$$
$$\lambda_{\text{skin}} = \mathbf{k}_{\text{light}} \left(1 - \alpha_{\text{f}} \right) \Psi_{\text{skin}}$$
$$\lambda_{\text{hair}} = \mathbf{k}_{\text{light}} \left(1 - \frac{\alpha_{\text{f}}}{2} \right) \Psi_{\text{hair}}$$
$$f = \alpha_{\text{f}} \lambda_{\text{hair}} + \left(1 - \alpha_{\text{f}} \right) \lambda_{\text{skin}}$$

  - Ψ_{skin} is calculated by some other means

Implementing BRDFs in Real-Time

- BRDF formulations assume integration over all incoming light in the positive hemisphere
  - Clearly impractical for real-time rendering!
  - Not very practical for off-line rendering either...
- Four high-level strategies:
  - Only implement point lights
    - Direction implementation
    - Factorization
  - Reflection map based pre-filtering / pre-calculation
  - Monte Carlo sampling
  - Deferred shading based techniques

Point Light Direct Implementation

- Use $\mathbf{l}$ for ω_i and $\mathbf{v}$ for ω_o and directly implement the math
  - We already do this for Phong lighting
  - More complex lighting equations can be prohibitively expensive
  - Since we're *not* integrating over the hemisphere, multiply the BRDF by π

Factorization

- Expensive equations can be factored into sums or products of functions of fewer variables
  - Each input vector (i.e., $\mathbf{v}$, $\mathbf{l}$, $\mathbf{h}$, $\mathbf{n}$, etc.) or dot-product of vectors becomes an input to one function
  - Each function is stored in some sort of texture
  - This technique works really well for sampled BRDFs
- Using two textures, the Poulin-Fournie anisotropic satin BRDF can be implemented as:

$$\alpha\, p(\mathbf{v})\, q(\mathbf{h})\, p(\mathbf{l})$$

  - $p()$ and $q()$ represent texture look-ups and α is a special scaling factor

Factorization

- Remember the Banks BRDF for strands:

$$\mathbf{v}\cdot\mathbf{r} = (\mathbf{v}\cdot\mathbf{t})(\mathbf{l}\cdot\mathbf{t}) - \left(\sqrt{1-(\mathbf{v}\cdot\mathbf{t})^2}\sqrt{1-(\mathbf{l}\cdot\mathbf{t})^2}\right)$$

Factorization

- Remember the Banks BRDF for strands:

$$\mathbf{v}\cdot\mathbf{r} = (\mathbf{v}\cdot\mathbf{t})(\mathbf{l}\cdot\mathbf{t}) - \left(\sqrt{1-(\mathbf{v}\cdot\mathbf{t})^2}\sqrt{1-(\mathbf{l}\cdot\mathbf{t})^2}\right)$$

  - Note that $\mathbf{v}\cdot\mathbf{r}$ is a function of two dot-products
  - Store all possible values of $\mathbf{v}\cdot\mathbf{r}$ in a 2D texture and sample this texture using $\mathbf{v}\cdot\mathbf{t}$ and $\mathbf{l}\cdot\mathbf{t}$

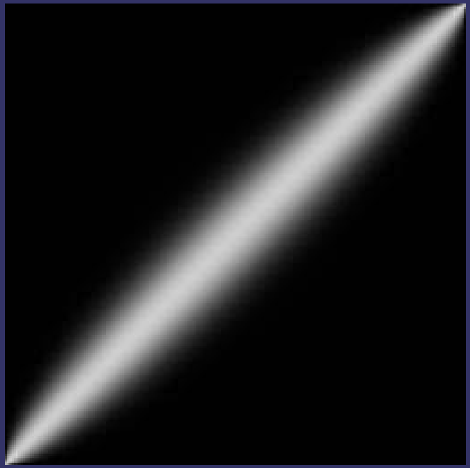


Image from ShaderX, Wordware Publishing, Inc.

References

University of Waterloo Factored BRDF Repository: http://www.cgl.uwaterloo.ca/Projects/rendering/Shading/database.html

Michael D. McCool, Jason Ang, Anis Ahmad, *Homomorphic Factorization of BRDFs for High-Performance Rendering*, SIGGAPH 2001, August 12-17, 2001. http://www.cgl.uwaterloo.ca/Projects/rendering/Papers/

Reflection Maps

- Reflection maps present additional challenges
  - Decent lighting require multiple samples
- As with the Phong lighting model, reflection maps can be pre-filtered using complex BRDFs
  - Doesn't work well with dynamic env. maps
  - Doesn't work *at all* with aniostropic BRDFs
    - The ideal reflection vector isn't enough information!

Grid Sampling

- Sample the reflection map at multiple, predetermined locations, use the sample vectors and the sampled values in the lighting equation

Grid Sampling

- Sample the reflection map at multiple, predetermined locations, use the sample vectors and the sampled values in the lighting equation
  - Might not sample the most important vectors for the lighting equation
    - For most equations, samples closer $\mathbf{r}$ are more important

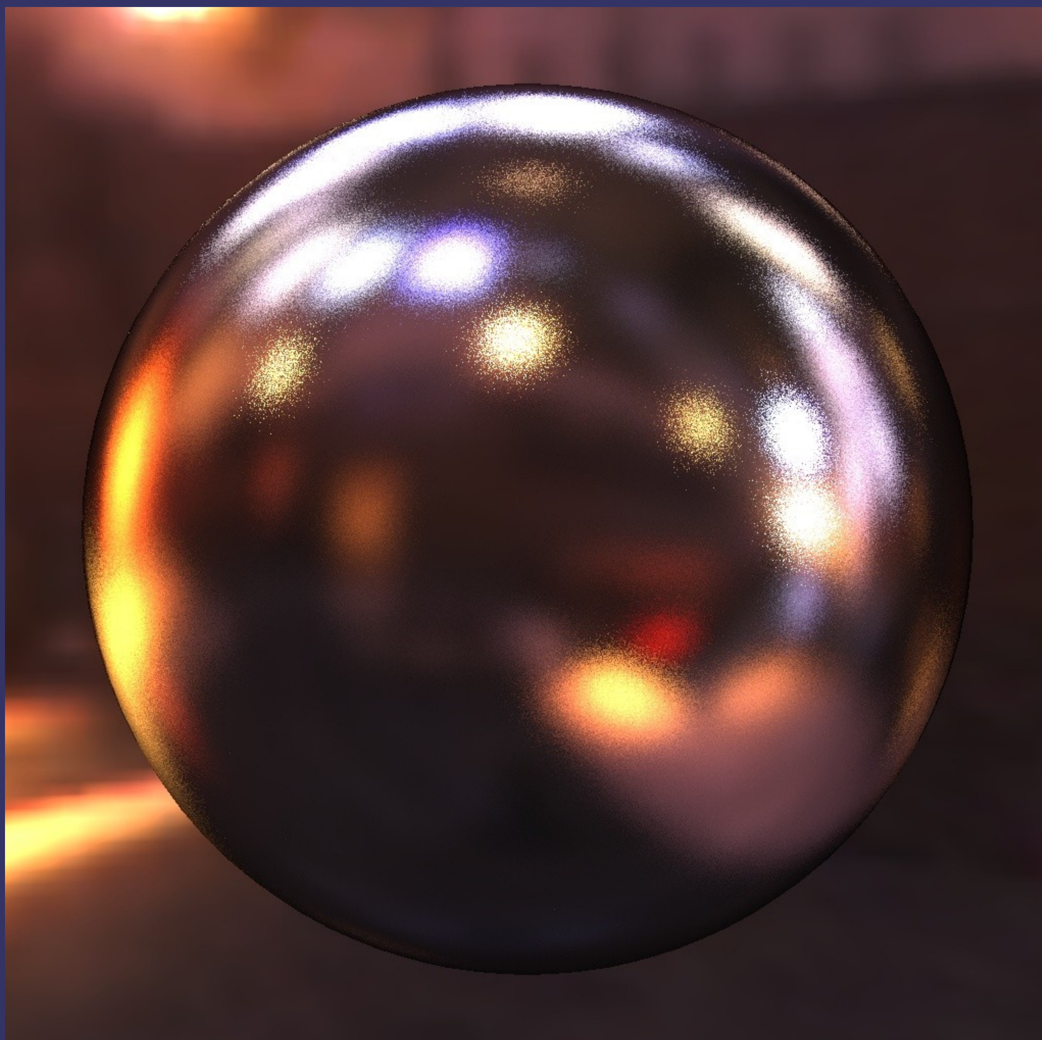
Grid Sampling

- Sample the reflection map at multiple, predetermined locations, use the sample vectors and the sampled values in the lighting equation
  - Might not sample the most important vectors for the lighting equation
    - For most equations, samples closer $\mathbf{r}$ are more important
  - Might not sample the most important vectors for the reflection map
    - If most of the refelction map is dark with just a few bright spots, those bright spots are more important
    - This problem is especially difficult to solve

Monte Carlo Integration

- Instead of sampling at regular intervals, sample at pseudo-random locations

Monte Carlo Integration



Images rendered with 40 samples per-fragment

Images from “Real-time Shading with Filtered Importance Sampling”
http://graphics.cs.ucf.edu/gpusampling/filter_is_intel.ppt
16-February-2010

Monte Carlo Integration

- Instead of sampling at regular intervals, sample at pseudo-random locations
  - Must sample many locations to eliminate noise
    - Where “many” may mean *thousands*
  - Or determine the random locations with a BRDF-dependent probability density function (PDF)
    - Several of the papers from this term include a PDF for the BRDF
    - Still several problems:
      - Generating good random numbers on the GPU is hard
      - Requires quite a few samples
        - Colbert and Křivánek found that around 40 looks good

Monte Carlo Integration

- Monte Carlo estimator for a BRDF:

$$\mathrm{L}(\mathbf{v}) \approx \frac{1}{n} \sum_{k=1}^{n} \frac{\mathrm{L_i}(\mathbf{u}_k) f(\mathbf{u}_k, \mathbf{v}) \cos\theta_{\mathbf{u}_k}}{p(\mathbf{u}_k, \mathbf{v})}$$

  - p is the PDF
  - $\mathbf{u}_k$ is a random light direction generated based on the PDF
    - Typically generate a uniform random value and remap it based on the PDF

Monte Carlo Integration

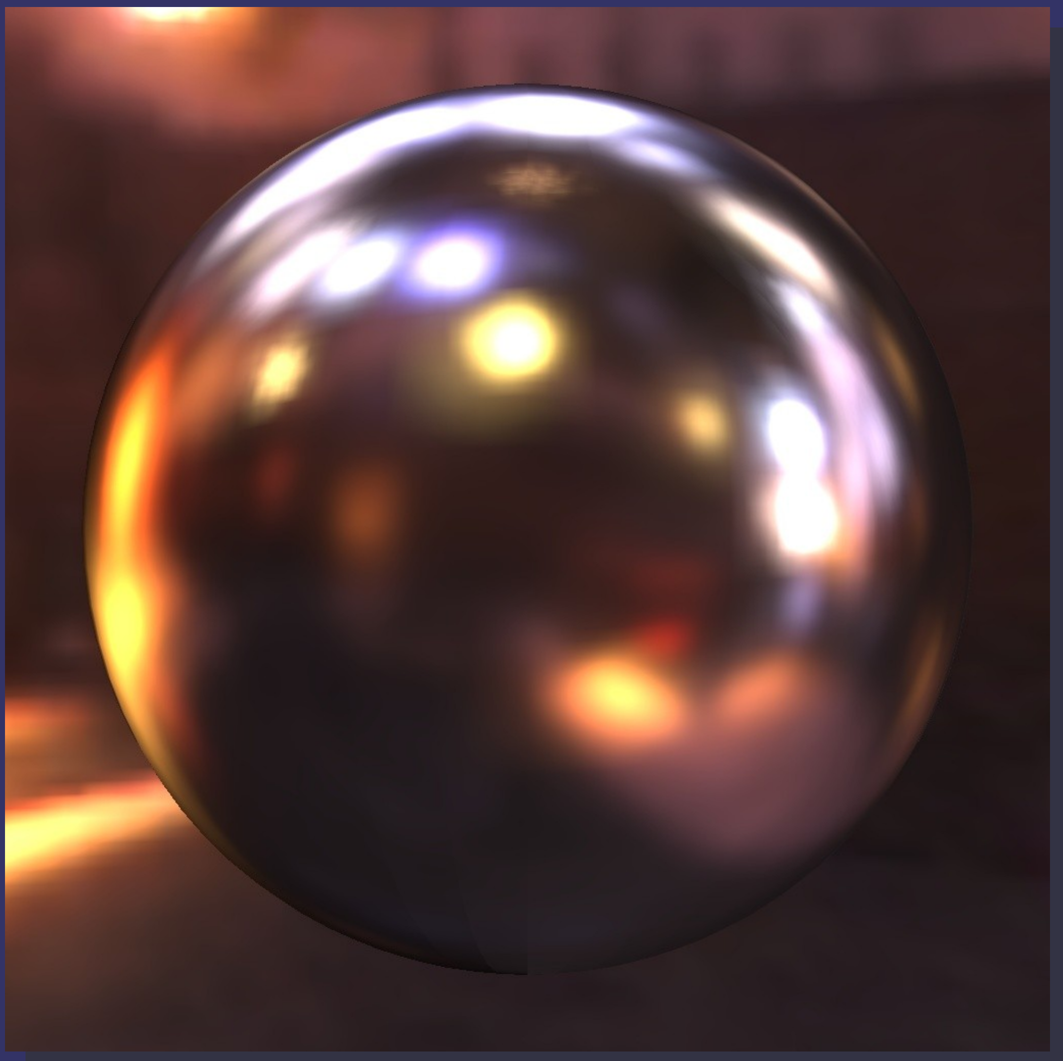
Images rendered with 40 samples per-fragment

Images from “Real-time Shading with Filtered Importance Sampling”
http://graphics.cs.ucf.edu/gpusampling/filter_is_intel.ppt
16-February-2010

Monte Carlo Integration

- Deterministic importance sampling causes unacceptable aliasing effects
  - Less important samples (i.e., less probable) are low-weighted individual points
  - Observe that neighbors of less probably samples are unlikely to be sampled
    - Allow those neighbors to contribute by using the PDF to select a mipmap level in the reflection map
    - Higher mipmap levels average larger regions into a single texel
    - Cube maps have weird distortion away from the axes, so a different type of reflection map should be used
      - The paper suggests dual-paraboloid maps

Monte Carlo Integration



Images rendered with 40 samples per-fragment

Images from “Real-time Shading with Filtered Importance Sampling”
http://graphics.cs.ucf.edu/gpusampling/filter_is_intel.ppt
16-February-2010

References

Colbert, M. and Křivánek, J. 2007. “Real-time shading with filtered importance sampling.” In *ACM SIGGRAPH 2007 Sketches* (San Diego, California, August 05 - 09, 2007). SIGGRAPH '07. ACM, New York, NY, 71. http://graphics.cs.ucf.edu/gpusampling/

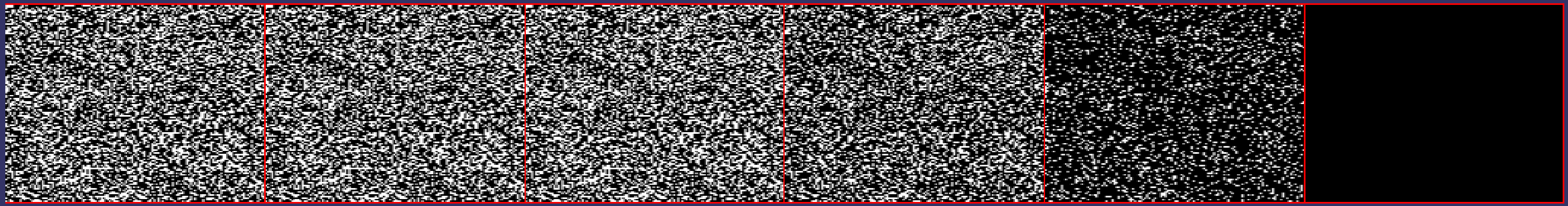
http://en.wikipedia.org/wiki/Monte_Carlo_integration

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!

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



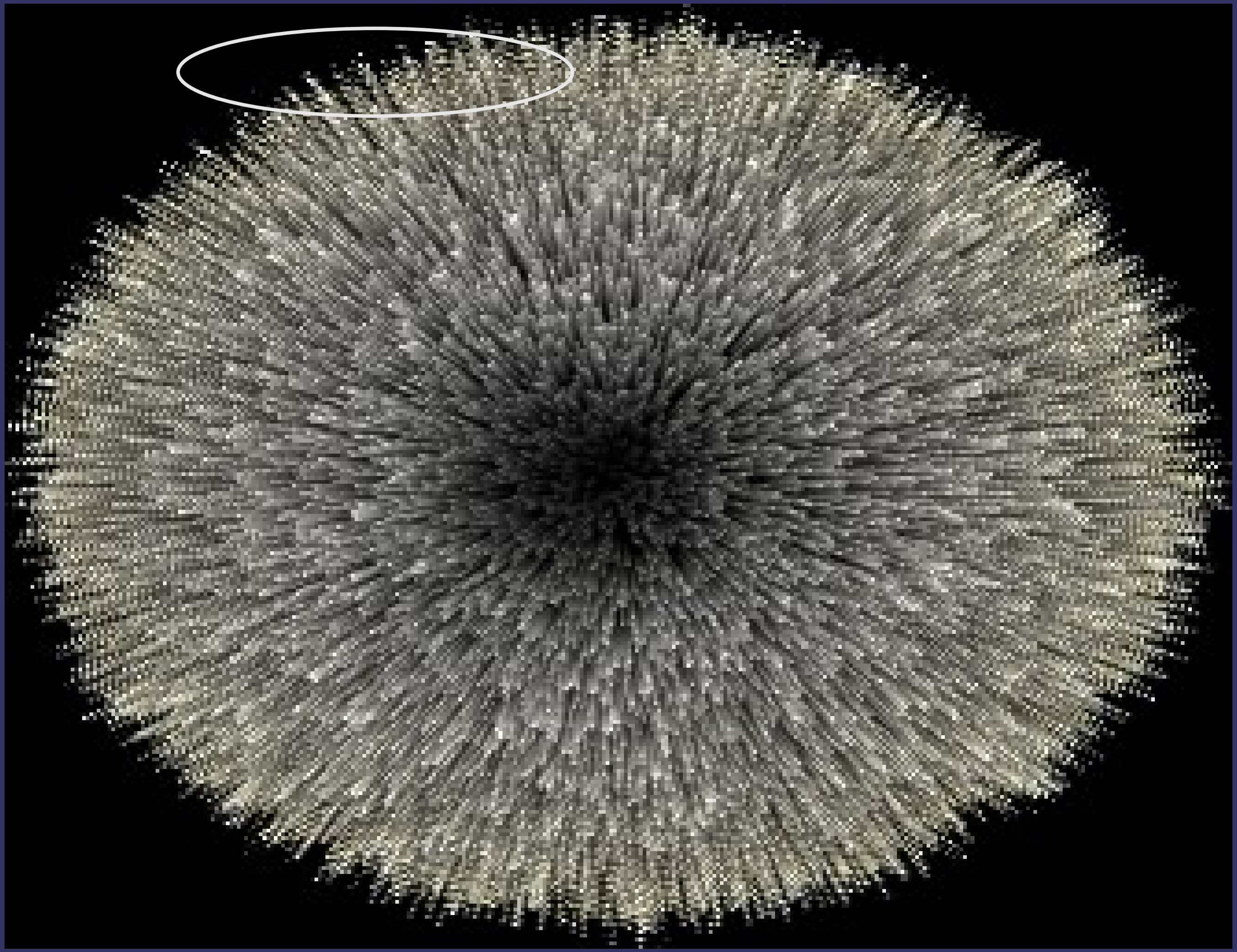
  - Draw each layer in a progressively larger “shell” around the original object geometry

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

Shells and Fins

- *But* this looks bad along the silhouette



13-May-2009

Shells and Fins

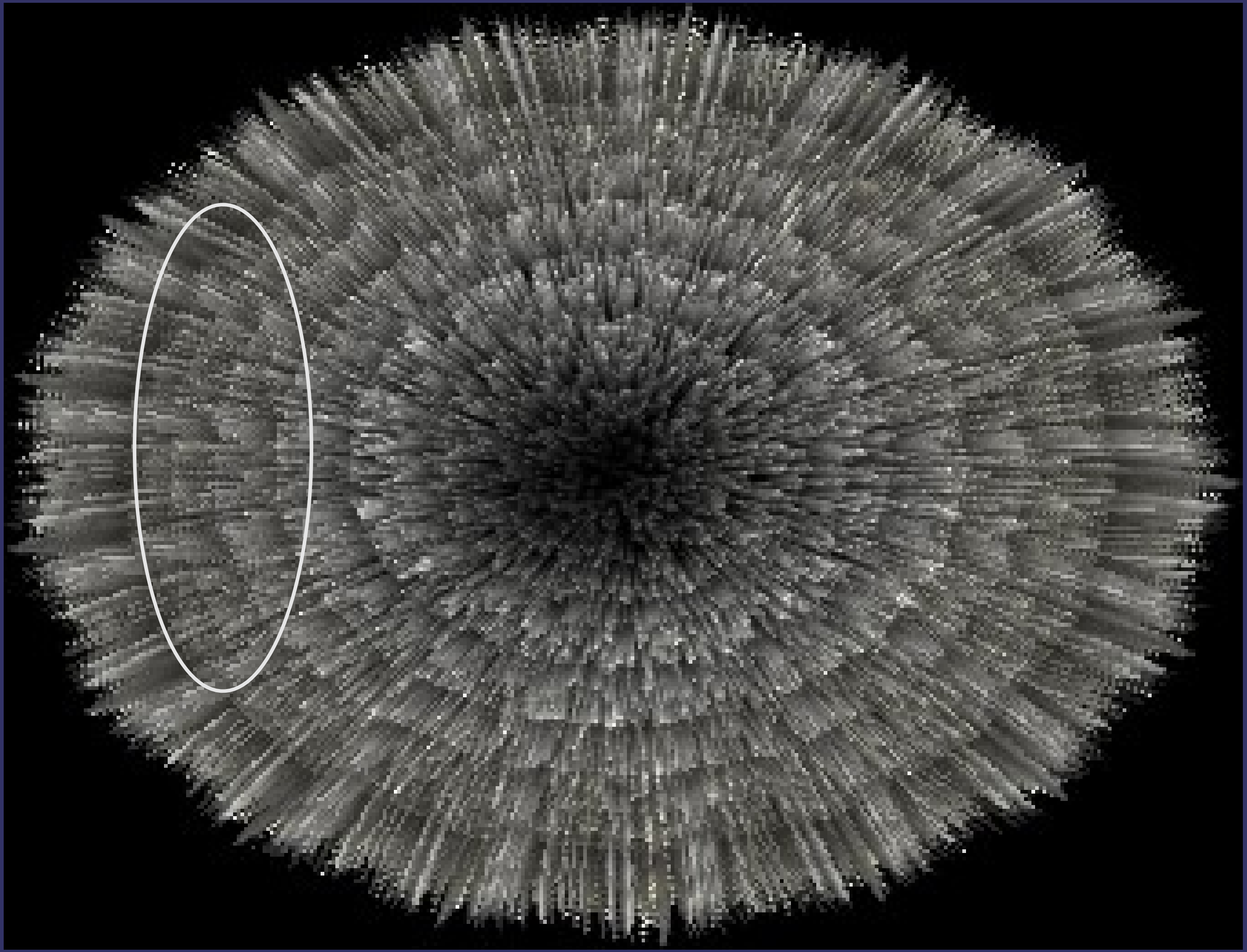
- Add fin geometry to each polygon
  - Create fin textures to look like side-on view of fur
  - Draw fin after drawing all shells
    - Enable alpha blending
    - Enable z-testing
    - Disable z-writing

Shells and Fins

- 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 two triangles:
    - One with vertices 0, 1, 1
    - The other with vertices 1, 0, 0

Shells and Fins

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



13-May-2009

Shells and Fins

- Gradually blend in fins as they approach the silhouette

$$\alpha_{\text{fin}} = \max\left(0, 2\left|\cos\left(\mathbf{v}, \mathbf{n}_{\text{fin}}\right)\right| - 1\right)$$

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

Shells and Fins

- Gradually blend in fins as they approach the silhouette

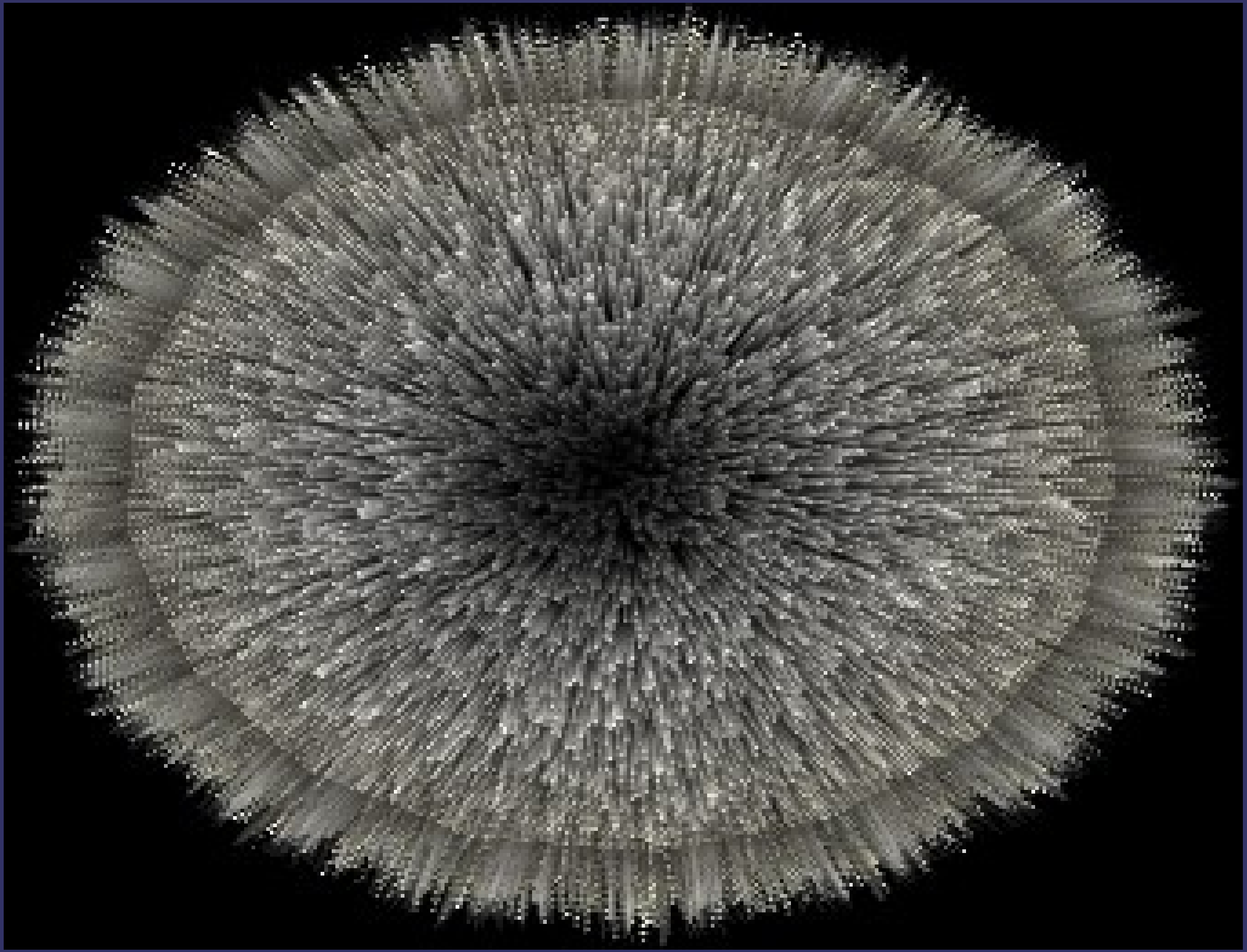
$$\alpha_{\text{fin}} = \max\left(0, 2\left|\cos\left(\mathbf{v}, \mathbf{n}_{\text{fin}}\right)\right| - 1\right)$$

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

$$\alpha_{\text{fin}} = \max\left(0, 2\left|\sin\left(\mathbf{v}, \mathbf{n}_{\text{surface}}\right)\right| - 1\right)$$

Shells and Fins

- Alpha blended fins



Lighting Shells and Fins

- Use the surface normal as the direction of the hair

$$\mathbf{k} = \mathbf{k}_\mathrm{d} \sin(\mathbf{n}_\mathrm{surface}, \mathbf{l})^{p_\mathrm{d}} + \mathbf{k}_\mathrm{s} \sin(\mathbf{n}_\mathrm{surface}, \mathbf{h})^{p_\mathrm{s}}$$

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

$$\begin{aligned} \mathbf{k} &= \mathbf{k}_\mathrm{d}\left(1-\cos(\mathbf{n}_\mathrm{surface}, \mathbf{l})^{p_\mathrm{d}/2}\right) + \mathbf{k}_\mathrm{s}\left(1-\cos(\mathbf{n}_\mathrm{surface}, \mathbf{h})^{p_\mathrm{s}/2}\right) \\ &= \mathbf{k}_\mathrm{d}\left(1-(\mathbf{n}_\mathrm{surface} \cdot \mathbf{l})^{p_\mathrm{d}/2}\right) + \mathbf{k}_\mathrm{s}\left(1-(\mathbf{n}_\mathrm{surface} \cdot \mathbf{h})^{p_\mathrm{s}/2}\right) \end{aligned}$$

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

$$s = \frac{d(1 - s_{min})}{d_{max}} + s_{min}$$

  - 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

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/FurShellsAndFins.pdf
This article focuses on optimizing shells-and-fins using Shader Model 4 features that are currently only supported in OpenGL 3.x.

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/

Next week...

- Quiz #3
- Non-photorealistic rendering
  - Cel (toon) shading
  - Silhouette edge rendering
  - Technical illustration
- Begin post-processing / image space effects

Legal Statement

This work represents the view of the authors and does not necessarily represent the view of Intel 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.