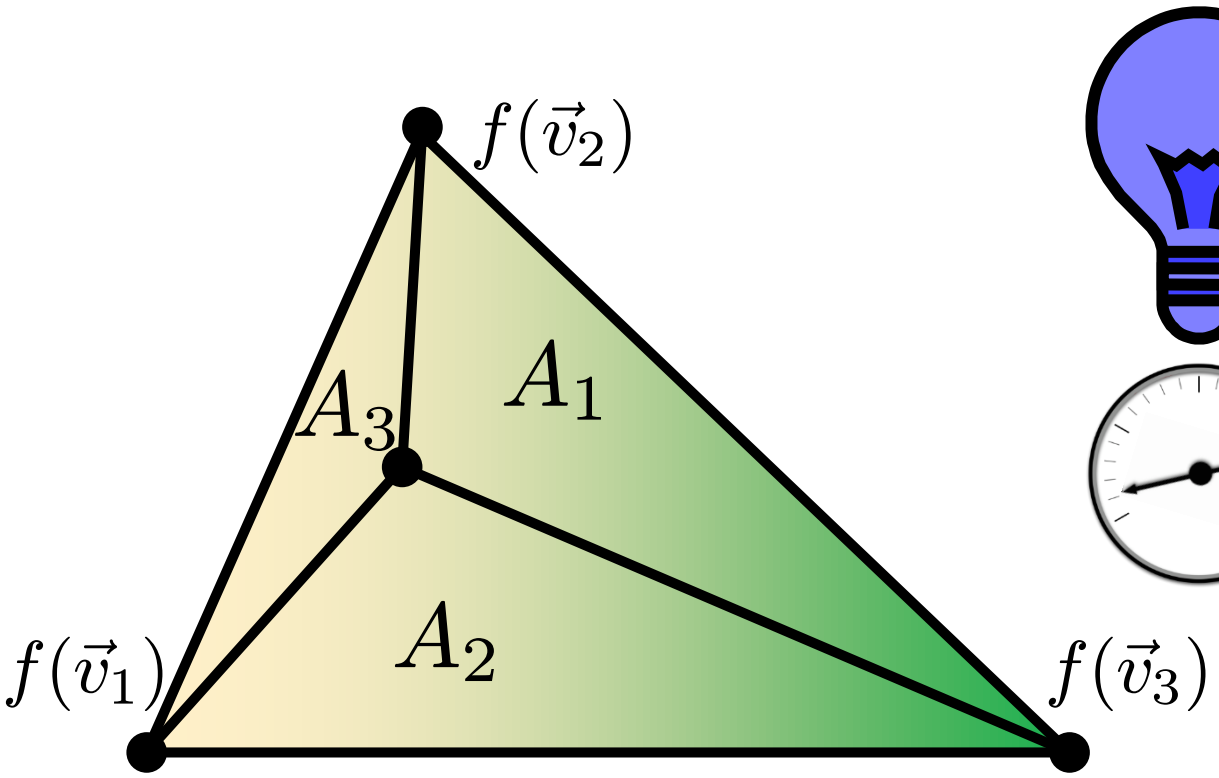
http://www.ann.jussieu.fr/~frey/papers/scivi/Cook%20R.L.,%20A%20reflectance%20model%20for%20computer%20graphics.pdf

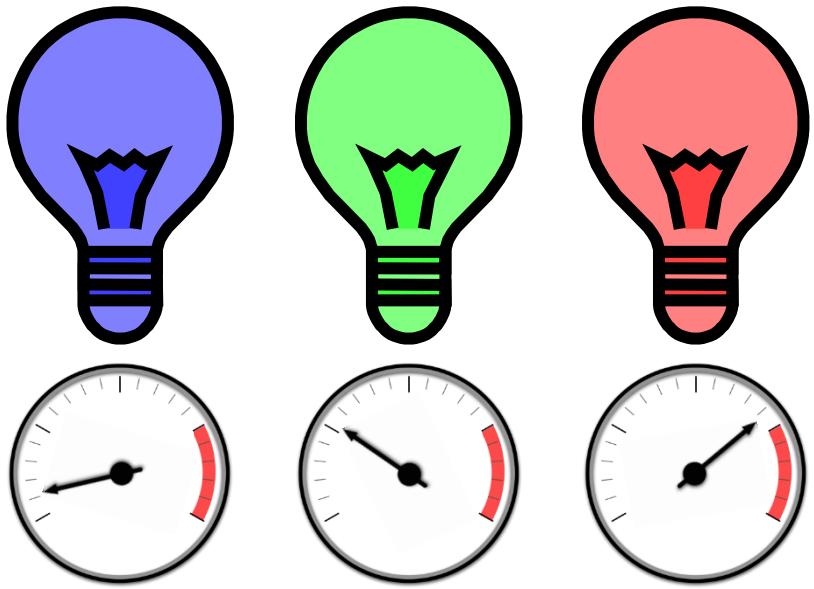
Shading, Texturing, and Materials

CS 148, Summer 2012

Introduction to Computer Graphics and Imaging

So Far



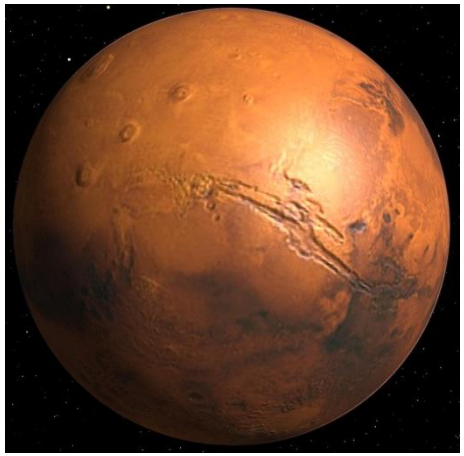


How to write shaders and relevant science, but not how to shade!

Intersection of Science and Art

Interplay between:

- **Efficiency** of rendering
- Desired **artistic** outcome
- **Physicality** of material

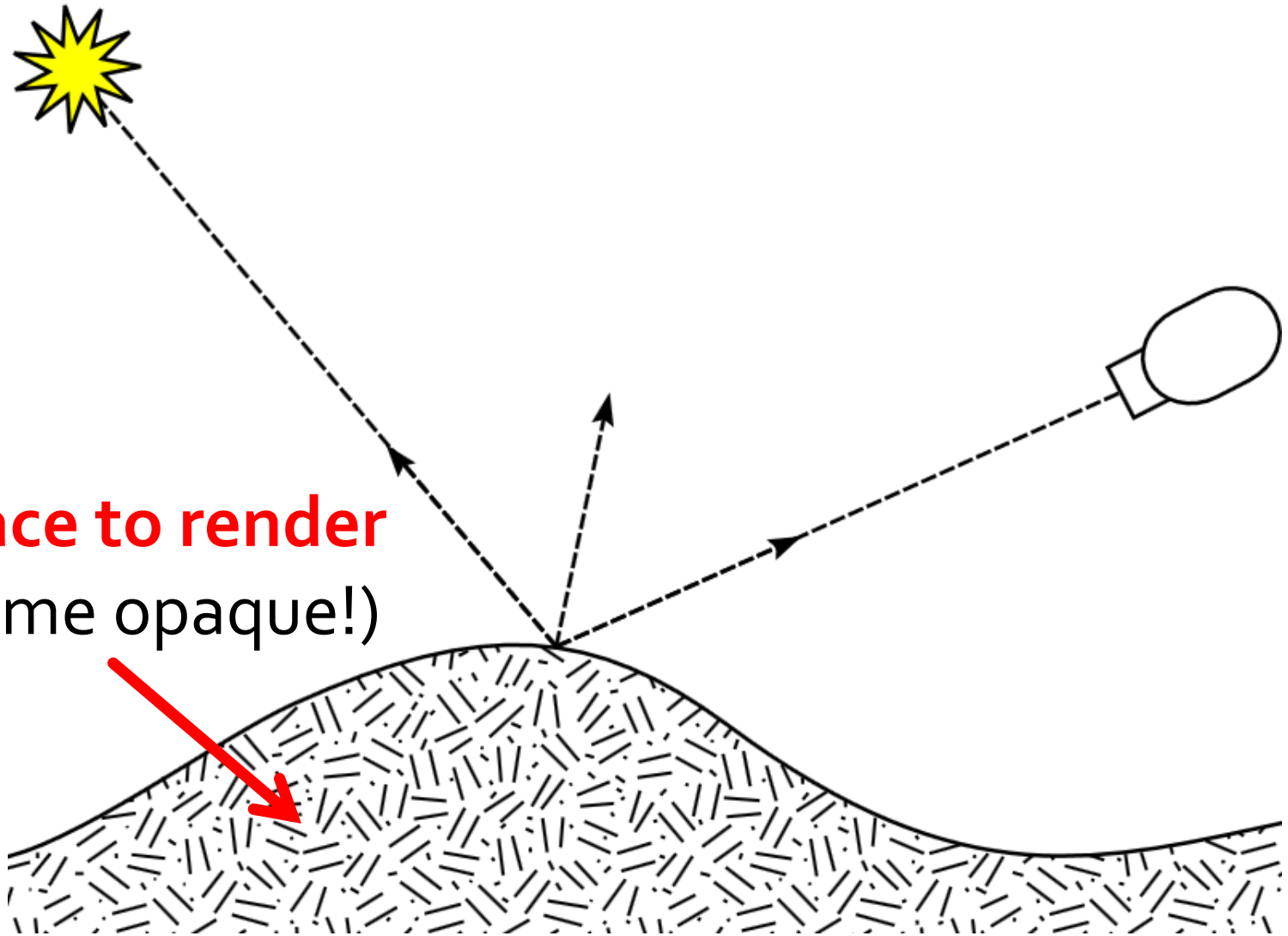


http://zanir.wz.cz/programming/dx81/img/012_Cube_Mapping.jpg
http://www.mrbluesummers.com/wp-content/uploads/2010/07/mental-ray-parti-volume-scatter-color.jpg
http://www.lepp.cornell.edu/~seb/celestia/images/mars_spec_ogl2.jpg

Problem at Hand

How do physical materials interact with light?

Basic Shading Model Ingredients

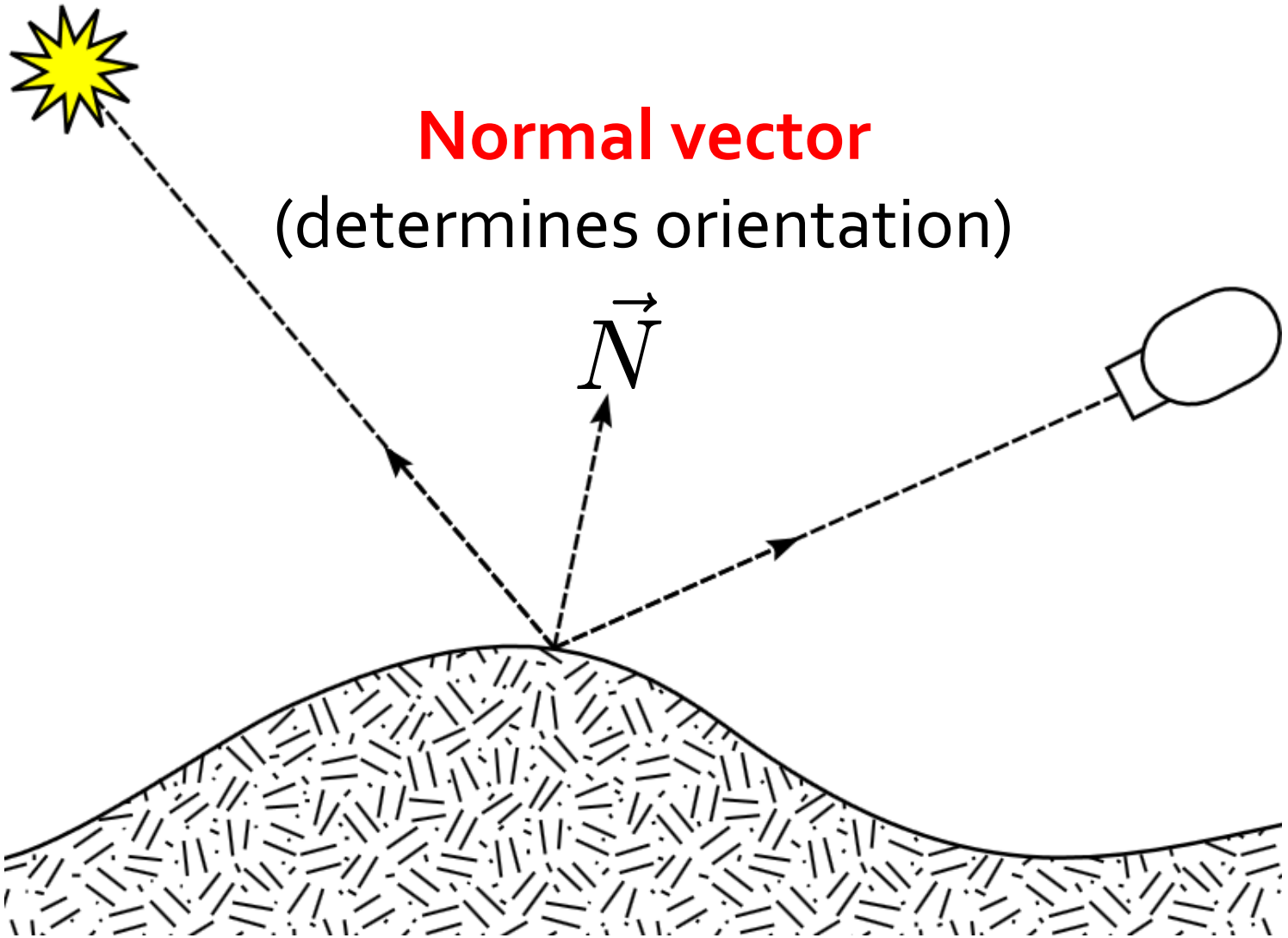


Surface to render

(assume opaque!)

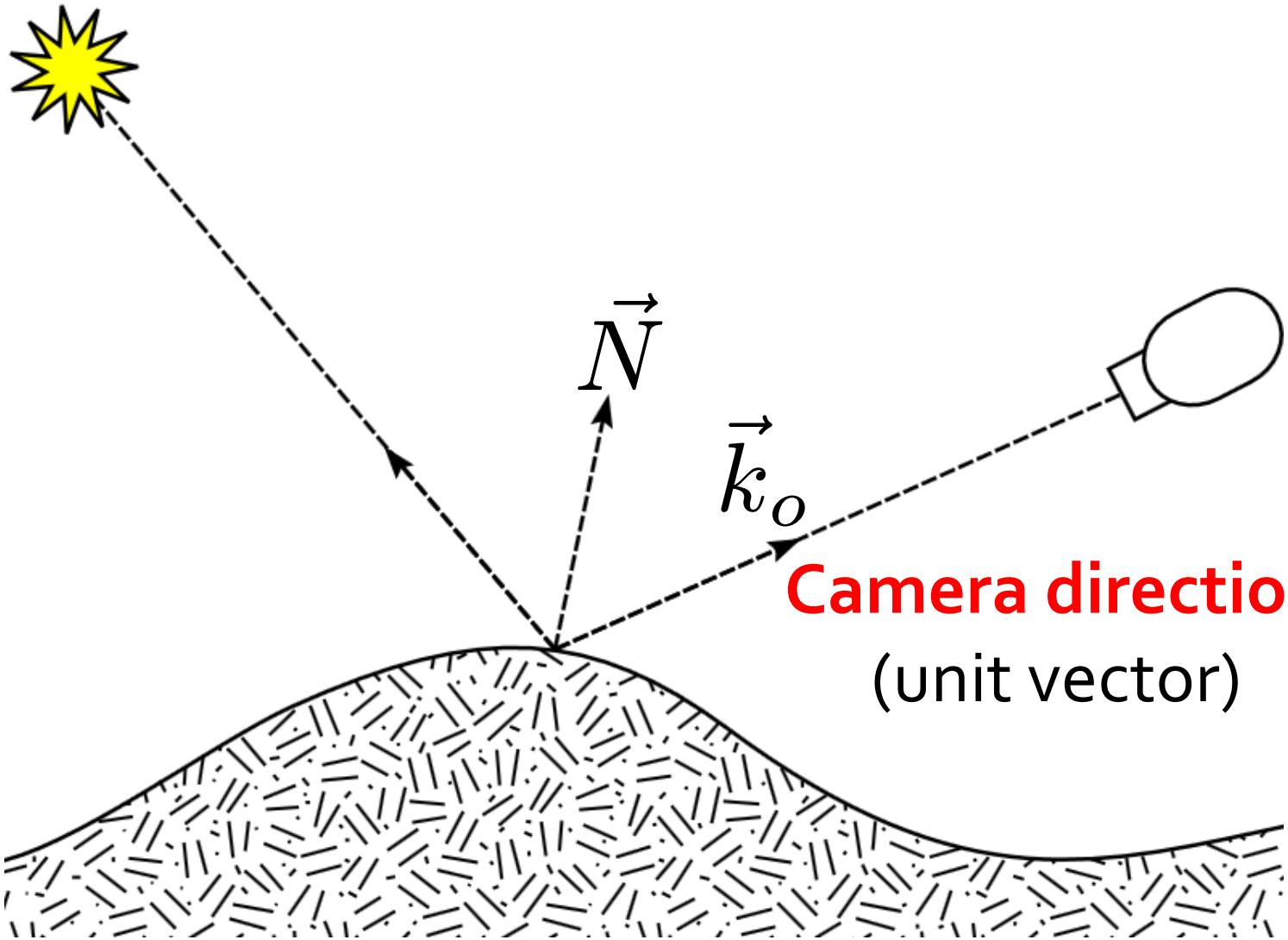
http://upload.wikimedia.org/wikipedia/commons/e/ed/BRDF_Diagram.svg

Basic Shading Model Ingredients



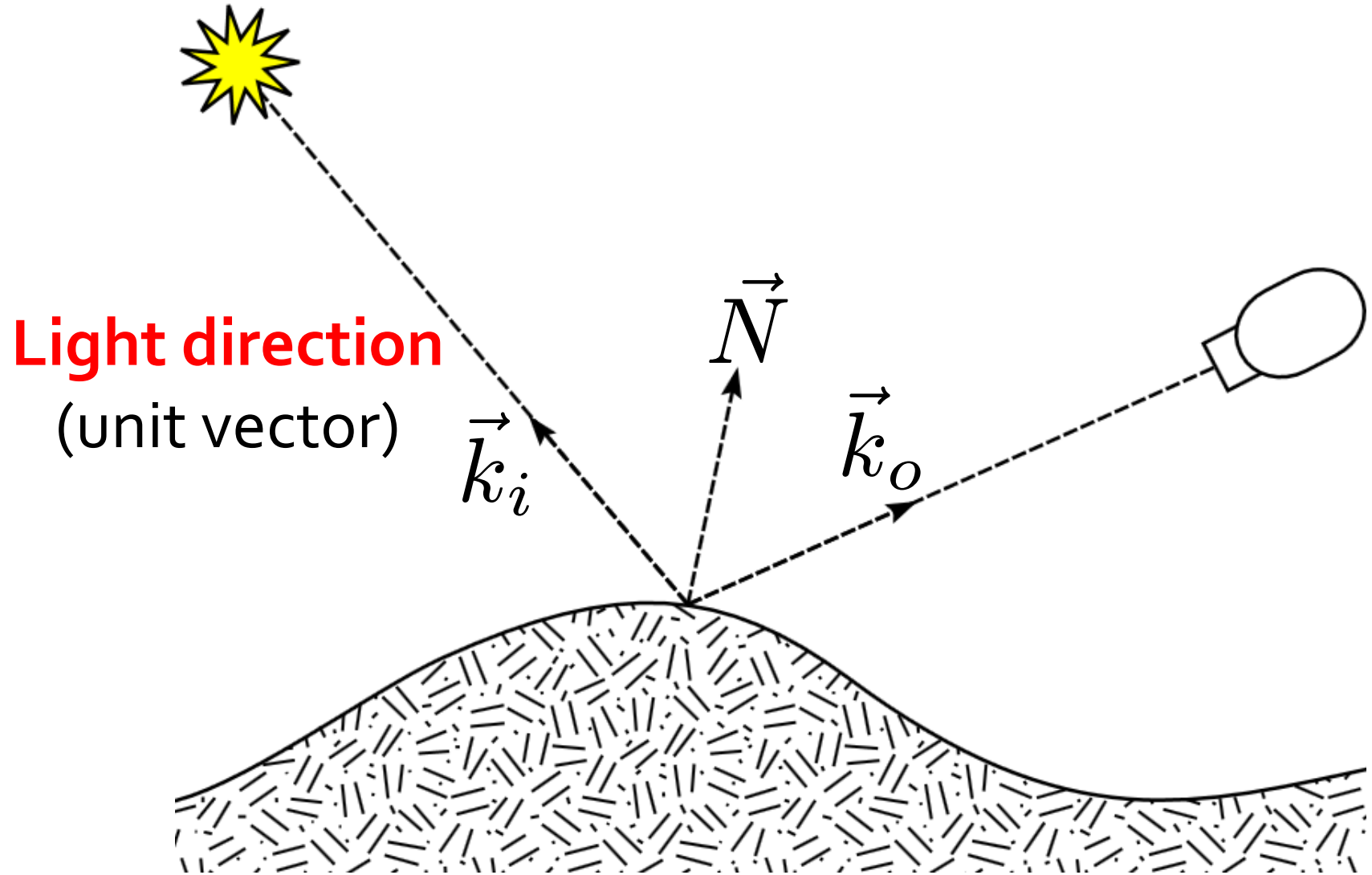
http://upload.wikimedia.org/wikipedia/commons/e/ed/BRDF_Diagram.svg

Basic Shading Model Ingredients



http://upload.wikimedia.org/wikipedia/commons/e/ed/BRDF_Diagram.svg

Basic Shading Model Ingredients



http://upload.wikimedia.org/wikipedia/commons/e/ed/BRDF_Diagram.svg

BRDFs

$$\rho(\vec{k}_i, \vec{k}_o; \vec{N})$$

Amount of light leaving the surface in each direction given input direction

Bidirectional Reflectance Distribution Function

Typical BRDF Properties

- Conserves energy

$$\int_{S^2} \rho(\vec{k}_i, \vec{k}_o)(\vec{N} \cdot \vec{k}_o)\ d\vec{k}_o \cdot \quad 1\ \forall \vec{k}_i$$

- Positive

$$\rho \geq 0$$

- Obeys Helmholtz reciprocity

$$\rho(\vec{k}_i, \vec{k}_o) = \rho(\vec{k}_o, \vec{k}_i)$$

Typical BRDF Properties

- Conserves energy

(don't worry)

$$\int_{S^2} \rho(\vec{k}_i, \vec{k}_o)(\vec{N} \cdot \vec{k}_o)\ d\vec{k}_o \cdot \ \ 1 \ \forall \vec{k}_i$$

- Positive

$$\rho \geq 0$$

- Obeys Helmholtz reciprocity

$$\rho(\vec{k}_i, \vec{k}_o) = \rho(\vec{k}_o, \vec{k}_i)$$

Measuring BRDFs

http://www.lcoptical.com/images/goniophotometer.jpg

Goniophotometer

Rendering Equation

$$L(\vec{k}_o) = \int_{S^2} \rho(\vec{k}_i, \vec{k}_o; \vec{N}) L_i(\vec{k}_i)(\vec{N} \cdot \vec{k}_i) \; d\vec{k}_i$$

Rendering Equation

$$L(\vec{k_o}) = \int_{S^2} \rho(\vec{k_i}, \vec{k_o}; \vec{N}) L_i(\vec{k_i})(\vec{N} \cdot \vec{k_i}) \, d\vec{k_i}$$

Outgoing light is sum of all incoming light reflected off the surface.

Rendering Equation

$$L(\vec{k}_o) = \int_{S^2} \rho(\vec{k}_i, \vec{k}_o; \vec{N}) L_i(\vec{k}_i)(\vec{N} \cdot \vec{k}_i)\ d\vec{k}_i$$

Outgoing intensity

Rendering Equation

$$L(\vec{k}_o) = \int_{S^2} \rho(\vec{k}_i, \vec{k}_o; \vec{N}) L_i(\vec{k}_i)(\vec{N} \cdot \vec{k}_i) \, d\vec{k}_i$$

Outgoing intensity

Sum over light source directions

Rendering Equation

$$L(\vec{k}_o) = \int_{S^2} \rho(\vec{k}_i, \vec{k}_o; \vec{N}) L_i(\vec{k}_i)(\vec{N} \cdot \vec{k}_i)\ d\vec{k}_i$$

Outgoing intensity

BRDF

Sum over light source directions

Rendering Equation

$$L(\vec{k}_o) = \int_{S^2} \rho(\vec{k}_i, \vec{k}_o; \vec{N}) L_i(\vec{k}_i)(\vec{N} \cdot \vec{k}_i)\ d\vec{k}_i$$

Outgoing intensity

Sum over light source directions

BRDF

Incoming light

Rendering Equation

$$L(\vec{k}_o) = \int_{S^2} \rho(\vec{k}_i, \vec{k}_o; \vec{N}) L_i(\vec{k}_i)(\vec{N} \cdot \vec{k}_i)\, d\vec{k}_i$$

Outgoing intensity

Sum over light source directions

BRDF

Incoming light

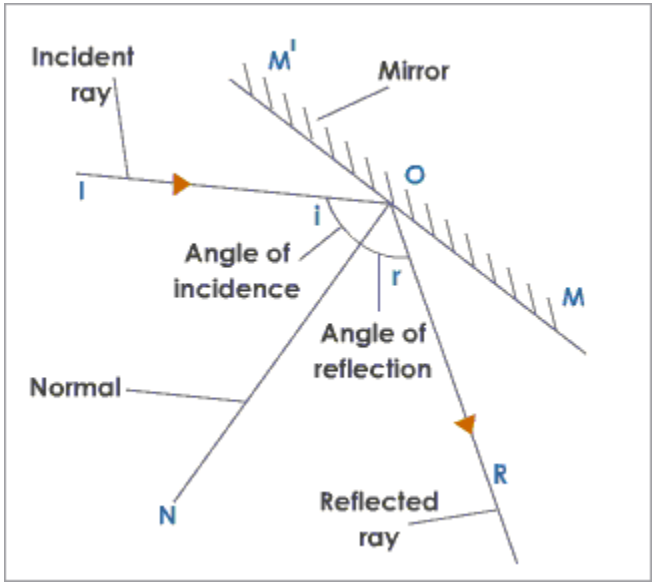
Normal component

For Simple/Fast Rendering

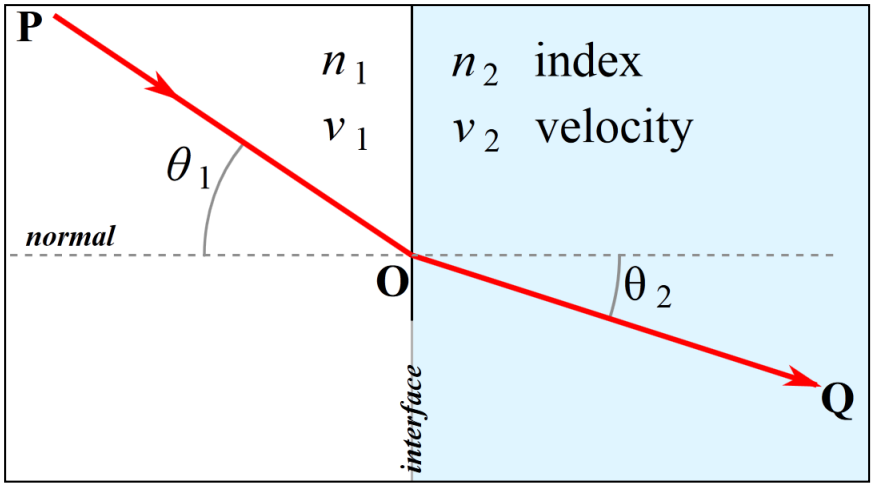
We rarely evaluate the BRDF or rendering integral but rather

know the form of *L* directly.

Two *L*'s We Already Covered

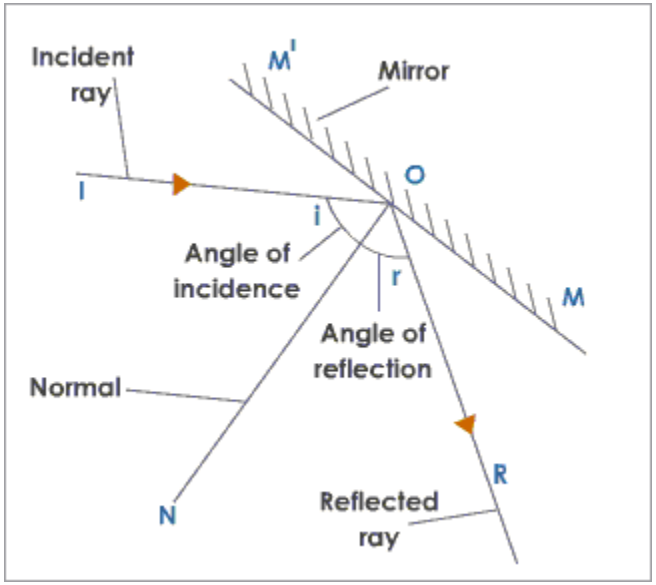


Reflection

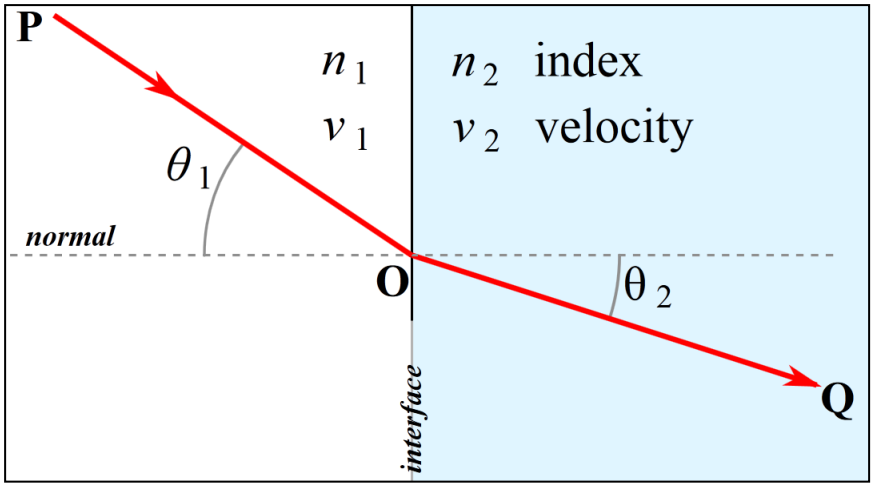


Refraction

Two *L*'s We Already Covered



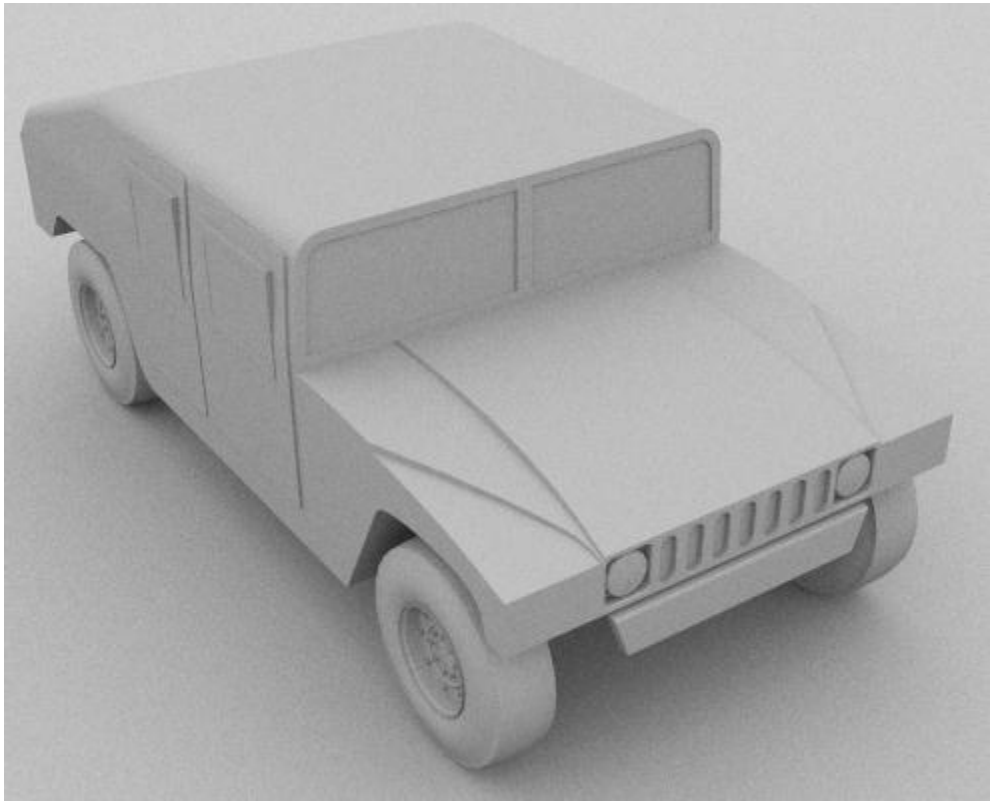
Reflection



Refraction

Challenge for home: What are the BRDFs?

Diffuse Surfaces



$$L = c_r c_l \max(\vec{N} \cdot \vec{k}_i, 0)$$

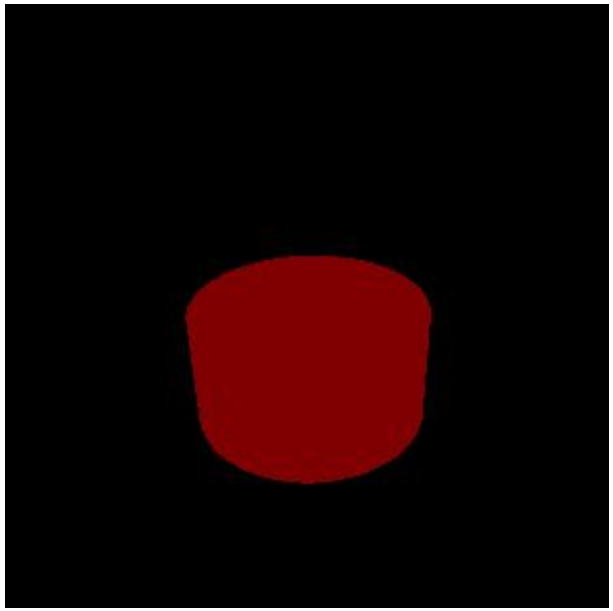
Light and surface color

Cosine term

http://joshwills.com/rendering/path_hummer.png

Lambertian ("Cosine") shading

Ambient Shading



Hack!

$$L = c_r c_a$$

Surface color

Ambient "light" color

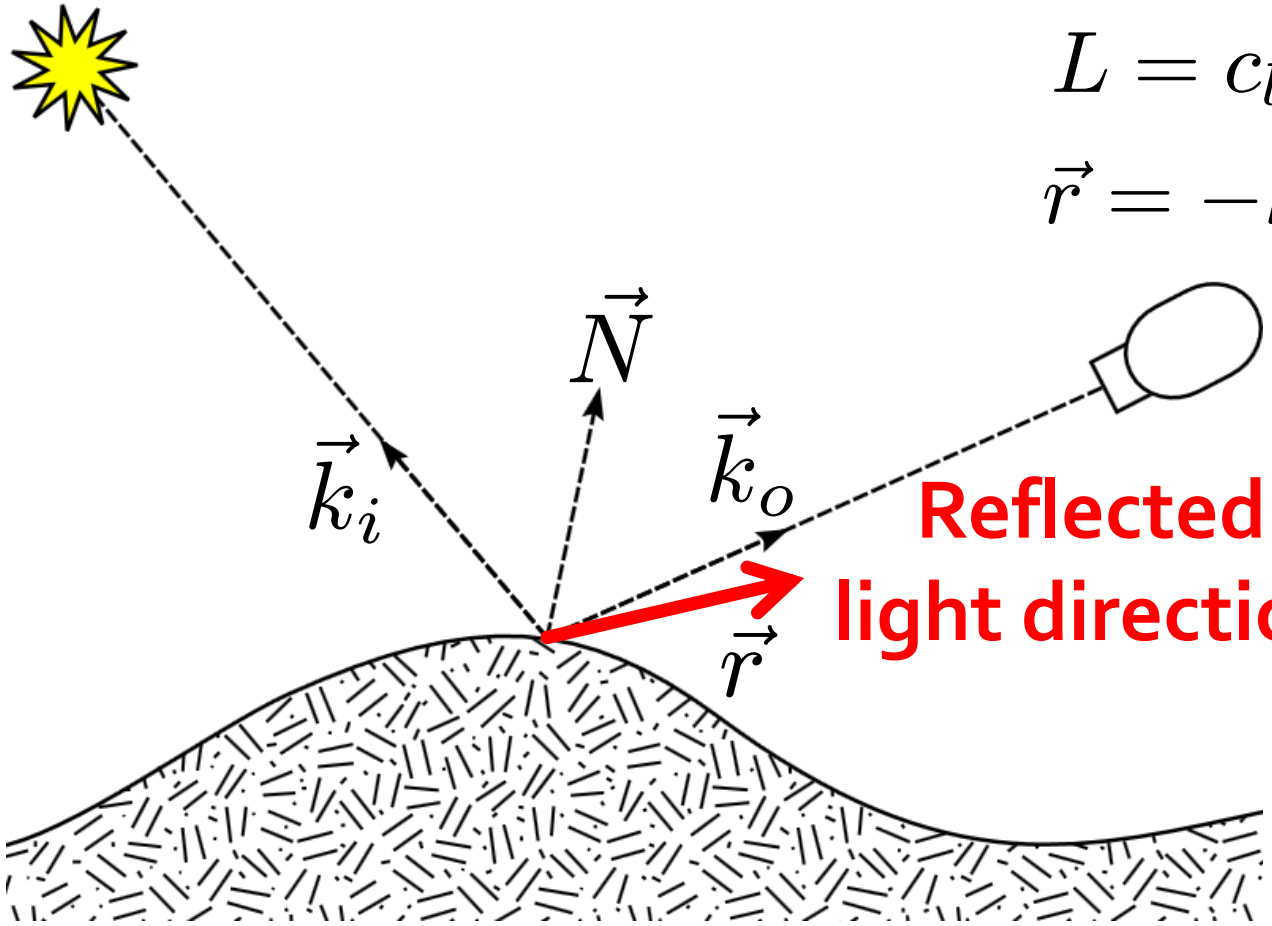
http://www.cs.utexas.edu/~fussell/courses/cs384g/projects/raytracing/samples/cyl_ambient.jpg

Constant regardless of light position

Phong Specular Term

$$L = c_l \max(0, \vec{k}_o \cdot \vec{r})^p$$

$$\vec{r} = -\vec{k}_i + 2(\vec{k}_i \cdot \vec{N})\vec{N}$$

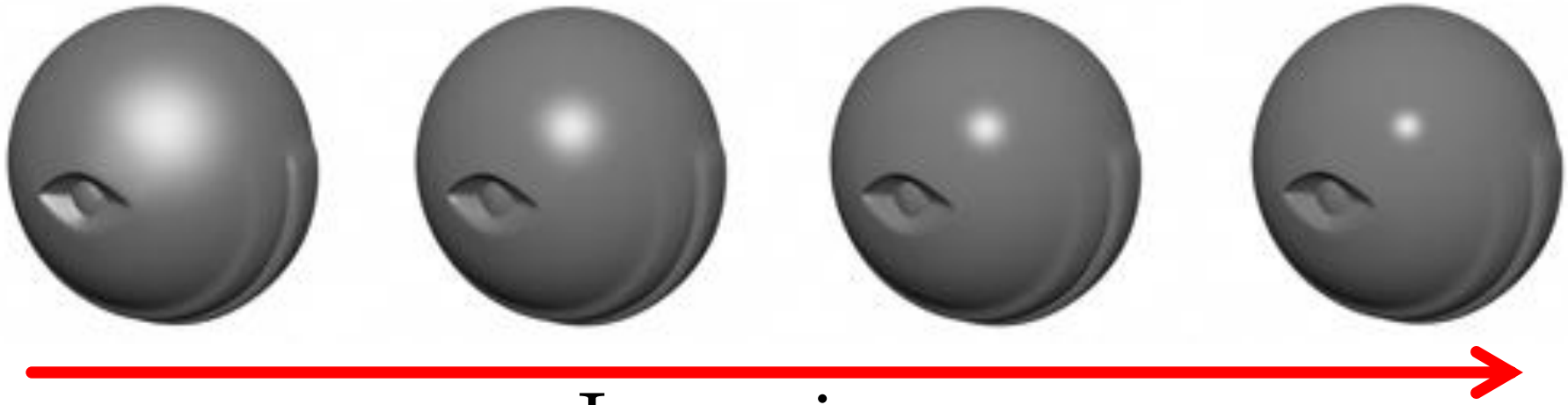


Adds glossy shine

Phong Specular Term

$$L = c_l \max(0, \vec{k}_o \cdot \vec{r})^p$$

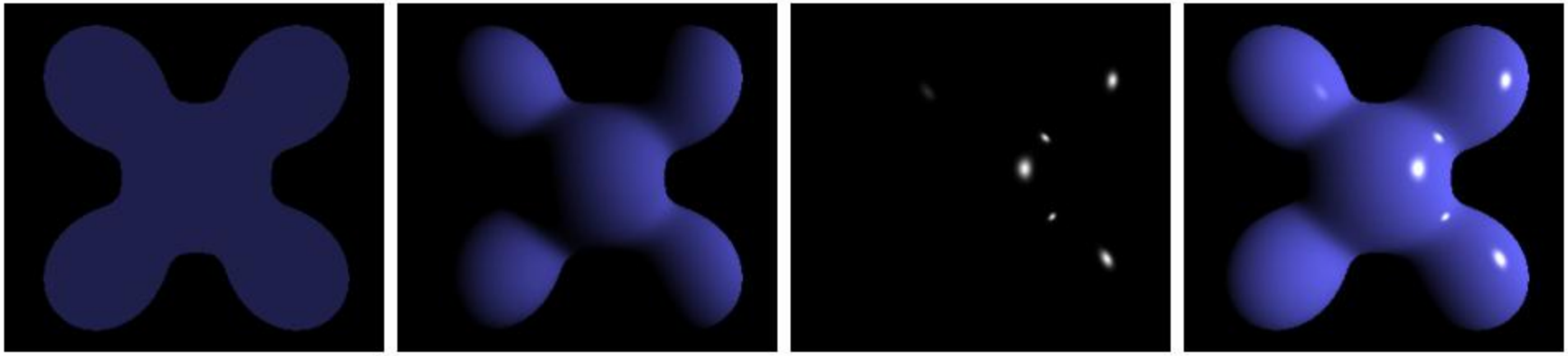
$$\vec{r} = -\vec{k}_i + 2(\vec{k}_i \cdot \vec{N})\vec{N}$$



http://http.developer.nvidia.com/CgTutorial/elementLinks/fig5_12.jpg

Adds glossy shine

Phong Reflection Model



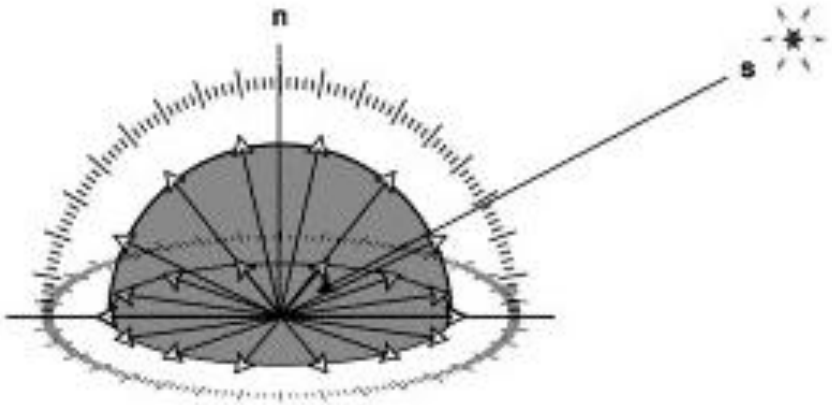
http://upload.wikimedia.org/wikipedia/commons/6/6b/Phong_components_version_4.png

Ambient-diffuse-specular (ADS)

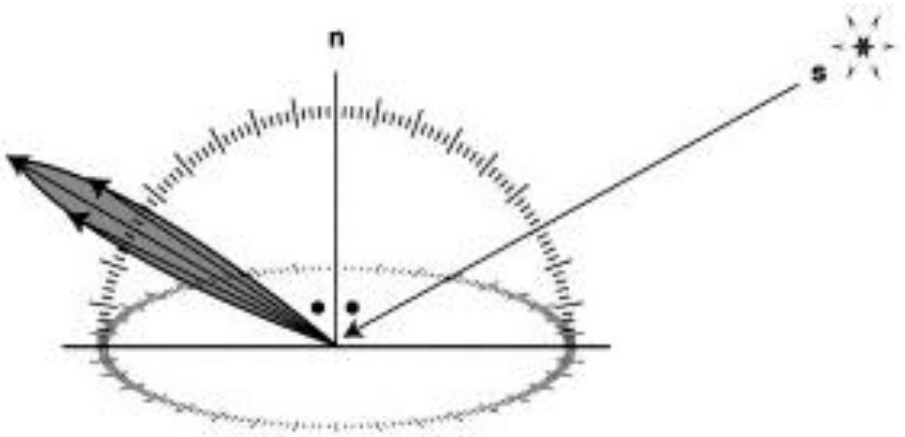
Specifying a Material

Can specify different **ambient**, **diffuse**, and **specular** reflected colors, plus Phong **exponent** *p*.

Visualizing a BRDF

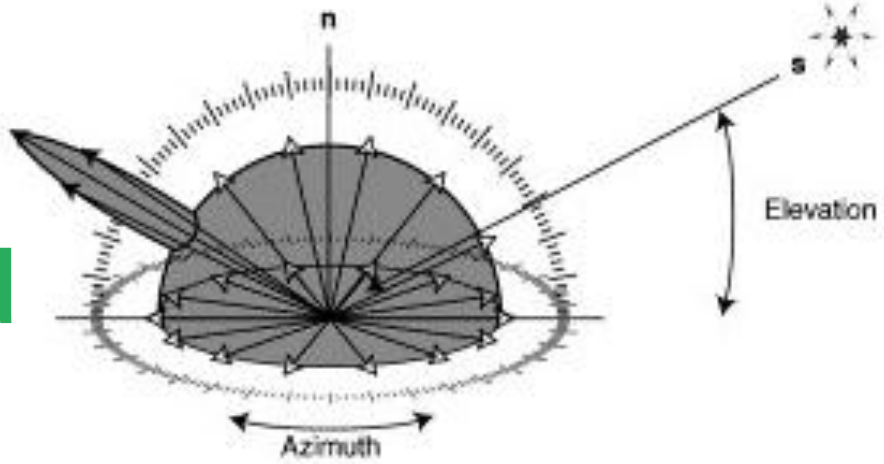


Lambertian



Specular

Combined

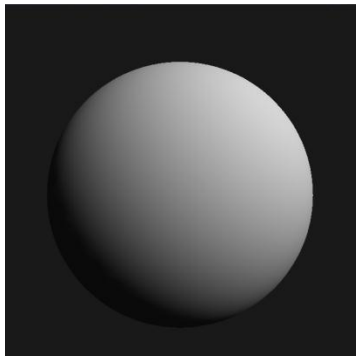


http://www.sciencedirect.com/science/article/pii/S0001691805000909

Lighting in OpenGL

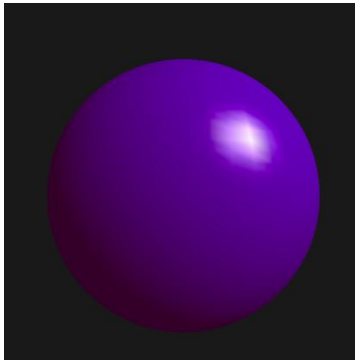
```
glEnable(GL_LIGHTING); //important!
GLfloat light0_position[] = { 1, 1, 1, 0 };
GLfloat light0_diffuse[] = { 1,1, 1, 1 };
GLfloat light0_specular[] = { 1, 1, 1, 1 };

glLightfv( GL_LIGHT0, GL_POSITION, light0_position);
glLightfv( GL_LIGHT0, GL_DIFFUSE, light0_diffuse);
//if you set these to 0's you won't see diffuse color
//regardless of your material
glLightfv( GL_LIGHT0, GL_SPECULAR, light0_specular);
//if you set these to 0's you won't see specular
//highlights regardless of your material
glEnable(GL_LIGHT0);
```



http://graphics.stanford.edu/courses/cs148-11-fall/lectures/rendering.pdf

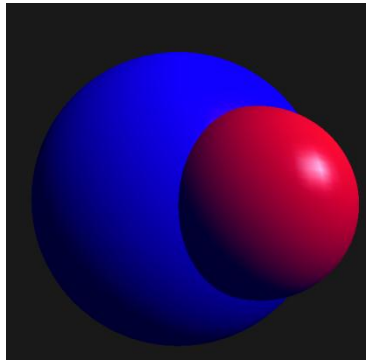
Material Properties in OpenGL



```
GLfloat mat_ambient[] = {1, .1, .8, 1 };
GLfloat mat_diffuse[] = { .2f, .2f, .6f, 1 };
GLfloat mat_specular[] = { .8f, .8f, .8f, 1 };
GLfloat mat_shininess[] = { 128 }; //range[0,128]

glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
```

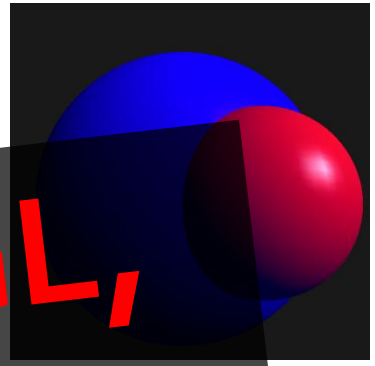
Material Properties in OpenGL



```
glEnable(GL_COLOR_MATERIAL);
…
//when you want to draw something with a new
//diffuse AND ambient color…
glColorMaterial(GL_FRONT, GL_AMBIENT_AND_DIFFUSE);
glColor4f(0,0,1,1);
drawSphere();
//now we just change the diffuse to red
glColorMaterial(GL_FRONT, GL_DIFFUSE);
glColor4f(1,0,0,1);
drawSphere();
```

http://graphics.stanford.edu/courses/cs148-11-fall/lectures/rendering.pdf

Material Properties in OpenGL

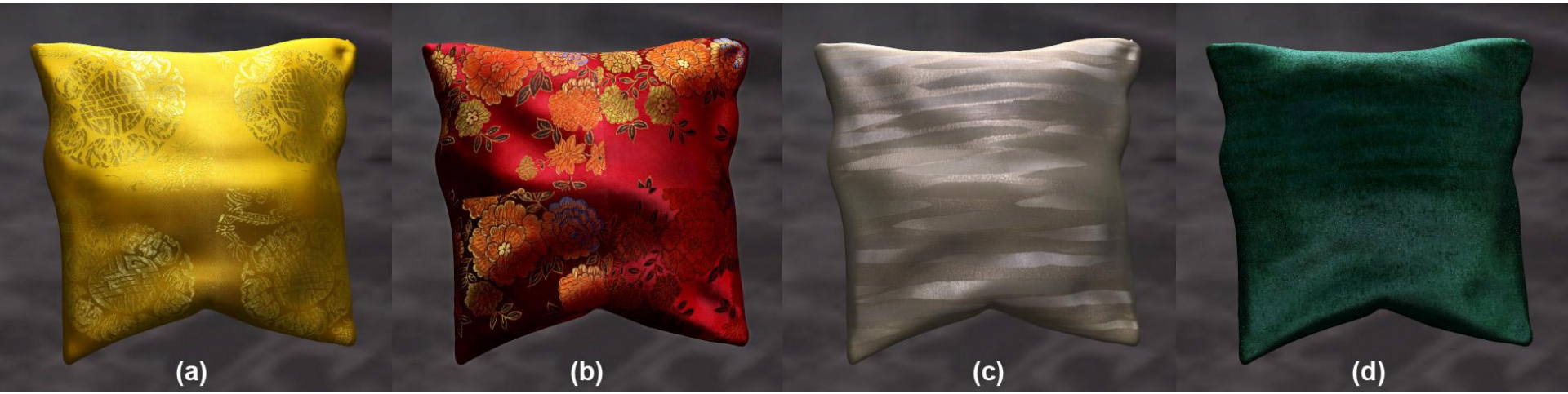


```
glEnable(GL_COLOR_MATERIAL);
…
//when you want to draw something with a new
//diffuse AND ambient color:
glColorMaterial(GL_FRONT, GL_AMBIENT_AND_DIFFUSE);
glColor4f(0,0,1,1);
drawSphere();
//now we just change the diffuse to red
glColorMaterial(GL_FRONT, GL_DIFFUSE);
glColor4f(1,0,0,1);
drawSphere();
```

In "modern" OpenGL, you should write shaders for this!

http://graphics.stanford.edu/courses/cs148-11-fall/lectures/rendering.pdf

Extensions of BRDFs



http://research.microsoft.com/en-us/um/people/johnsny/images/aniso_pillows.bmp

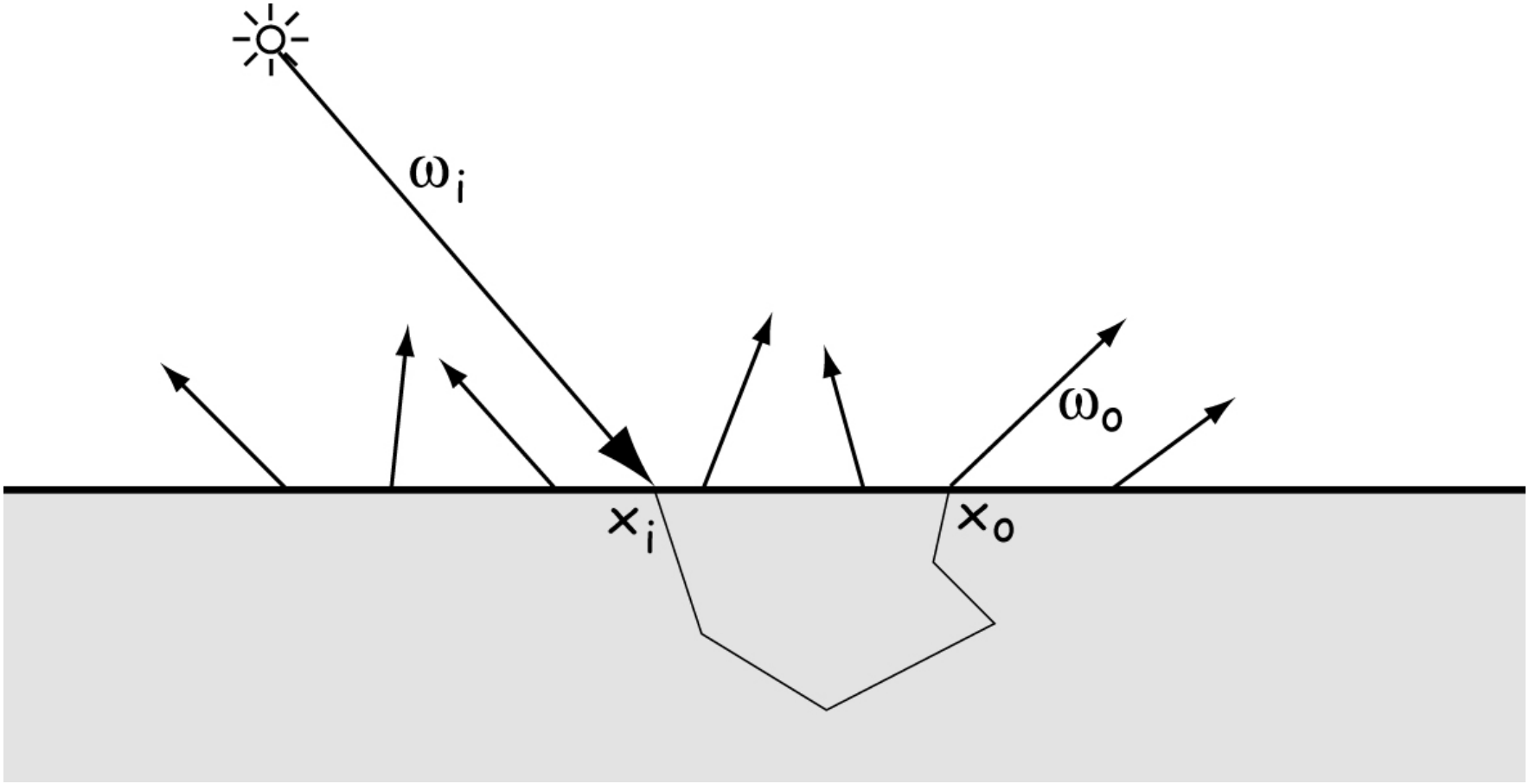
Spatially-varying BRDF (SVBRDF)

Extensions of BRDFs

http://graphics.ucsd.edu/~henrik/images/imgs/diana_bssrdf.jpg

B Subsurface Scattering RDF (BSSRDF)

Extensions of BRDFs



http://graphics.stanford.edu/courses/cs148-11-fall/lectures/rendering.pdf

B Subsurface Scattering RDF (BSSRDF)

Extensions of BRDFs

http://graphics.ucsd.edu/~henrik/images/imgs/face_brdf.jpg

B Subsurface Scattering RDF (BSSRDF)

Extensions of BRDFs

http://graphics.ucsd.edu/~henrik/images/imgs/milk.jpg

B Subsurface Scattering RDF (BSSRDF)

Extensions of BRDFs

http://graphics.ucsd.edu/~henrik/images/imgs/milk.jpg

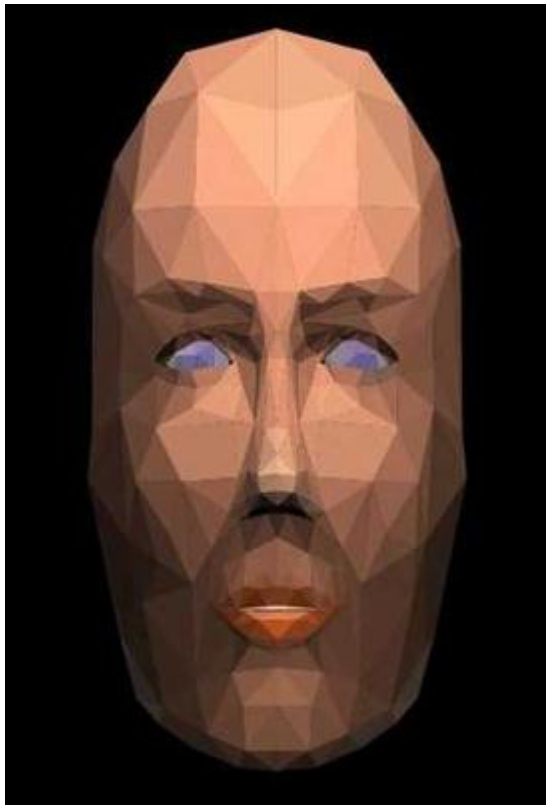
B Subsurface Scattering RDF (BSSRDF)

Extensions of BRDFs

http://graphics.ucsd.edu/~henrik/images/imgs/milk.jpg

B Subsurface Scattering RDF (BSSRDF)

Algorithms for Shading



Algorithm:

1. Calculate color *c* of triangle.
2. Fill whole triangle with *c*

http://www.svgopen.org/2005/papers/Converting3DFaceToSVG/Figure_10.png

Flat shading: One color per triangle

Algorithms for Shading

Algorithm:

1. Calculate color c_i of each vertex.
2. Interpolate pixel colors using barycentric weights.

http://www.svgopen.org/2005/papers/Converting3DFaceToSVG/Figure_10.png

Gouraud shading: One color per vertex

Algorithms for Shading

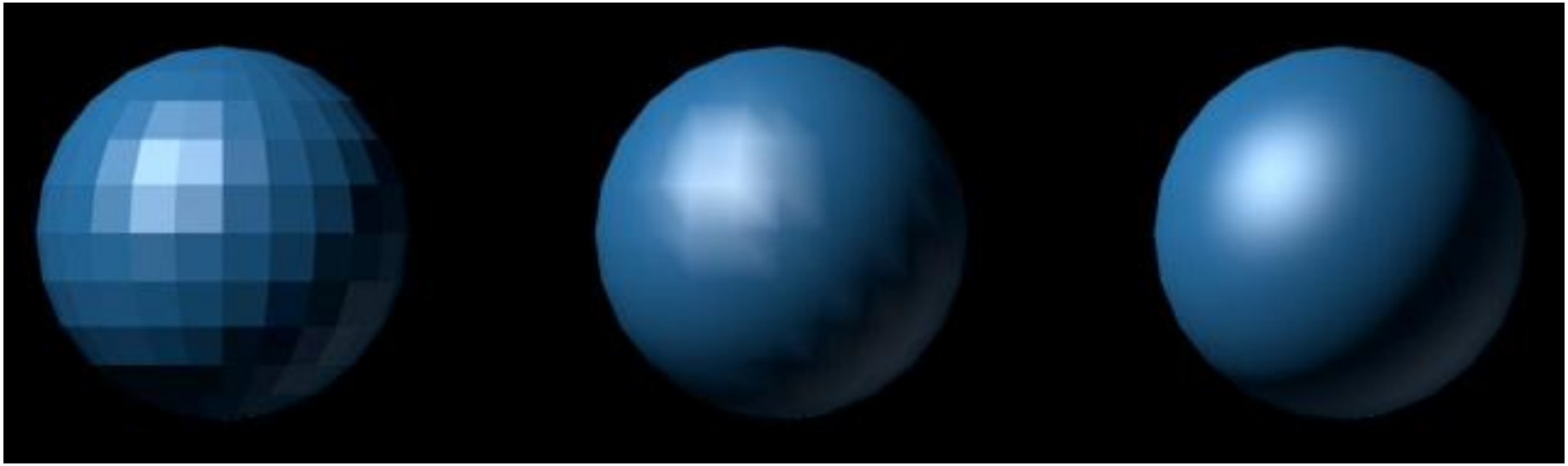
Algorithm:

1. Calculate normal N_i of each vertex.
2. Interpolate normal using barycentric weights, normalize, and use it for shading.

http://www.cs.cmu.edu/~ph/nyit/facet_gouraud_phong.jpeg

Phong shading: One color per fragment

Comparison

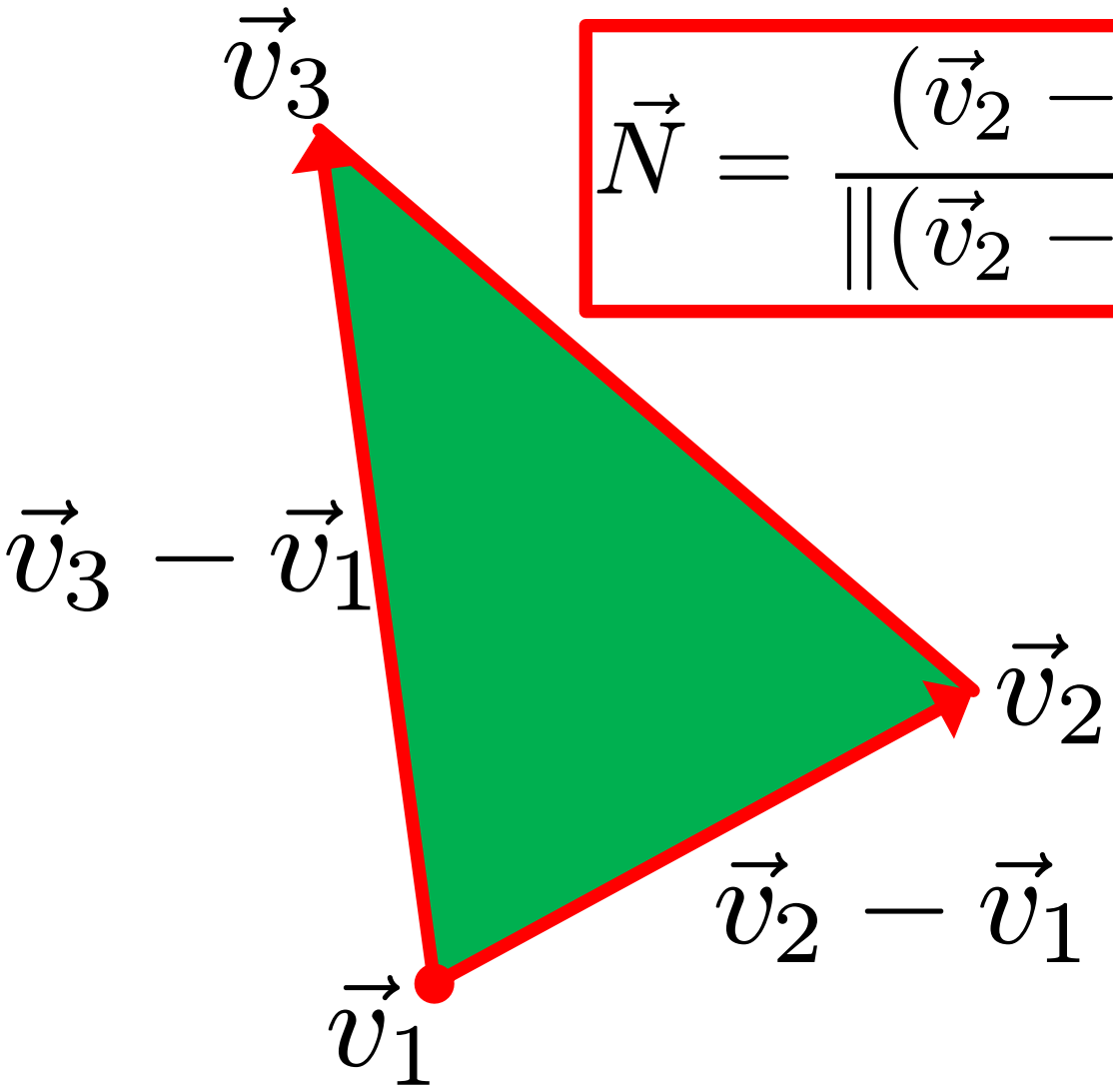


Which is which?

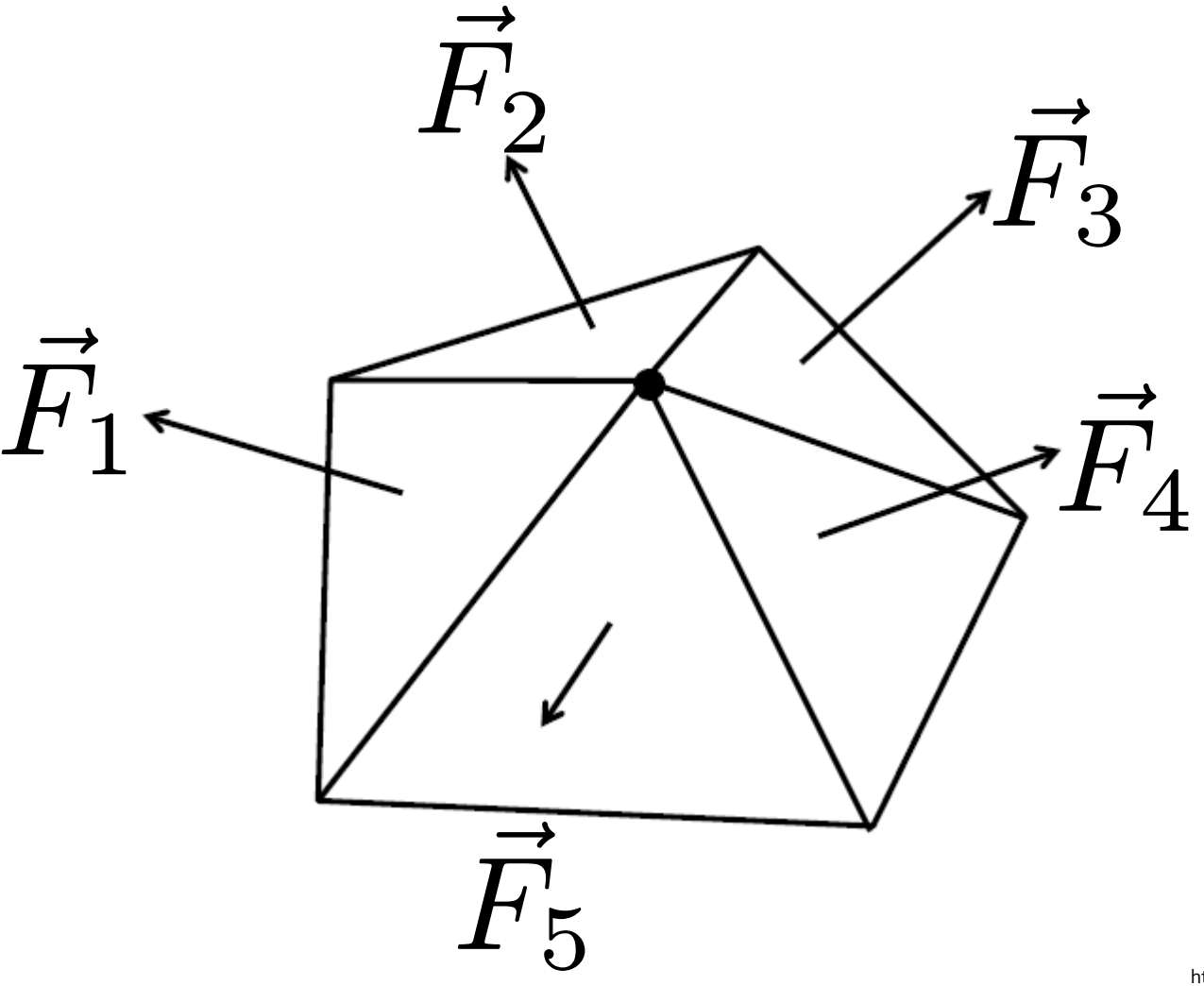
http://www.inf.pucrs.br/~pinho/CG/Aulas/Iluminacao/Gouraud_Phong_01.jpg

Recall: Computing Face Normals

$$\vec{N} = \frac{(\vec{v}_2 - \vec{v}_1) \times (\vec{v}_3 - \vec{v}_1)}{\|(\vec{v}_2 - \vec{v}_1) \times (\vec{v}_3 - \vec{v}_1)\|}$$

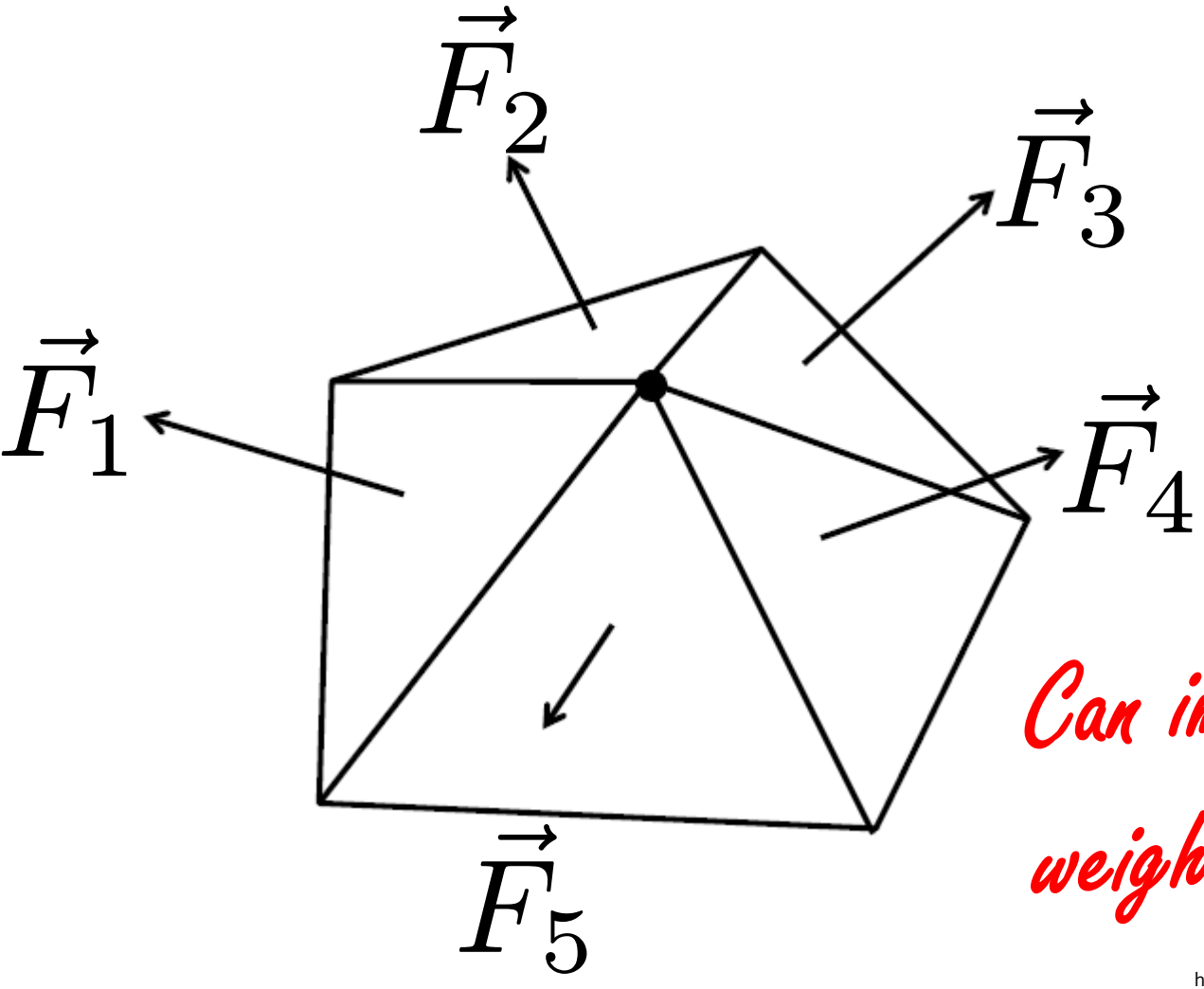


Computing Vertex Normals



$$\frac{1}{N}\sum_{i=1}^{N}\vec{F}_i$$

Computing Vertex Normals



$$\frac{1}{N}\sum_{i=1}^{N}\vec{F}_i$$

Can improve using area weights, other tricks!

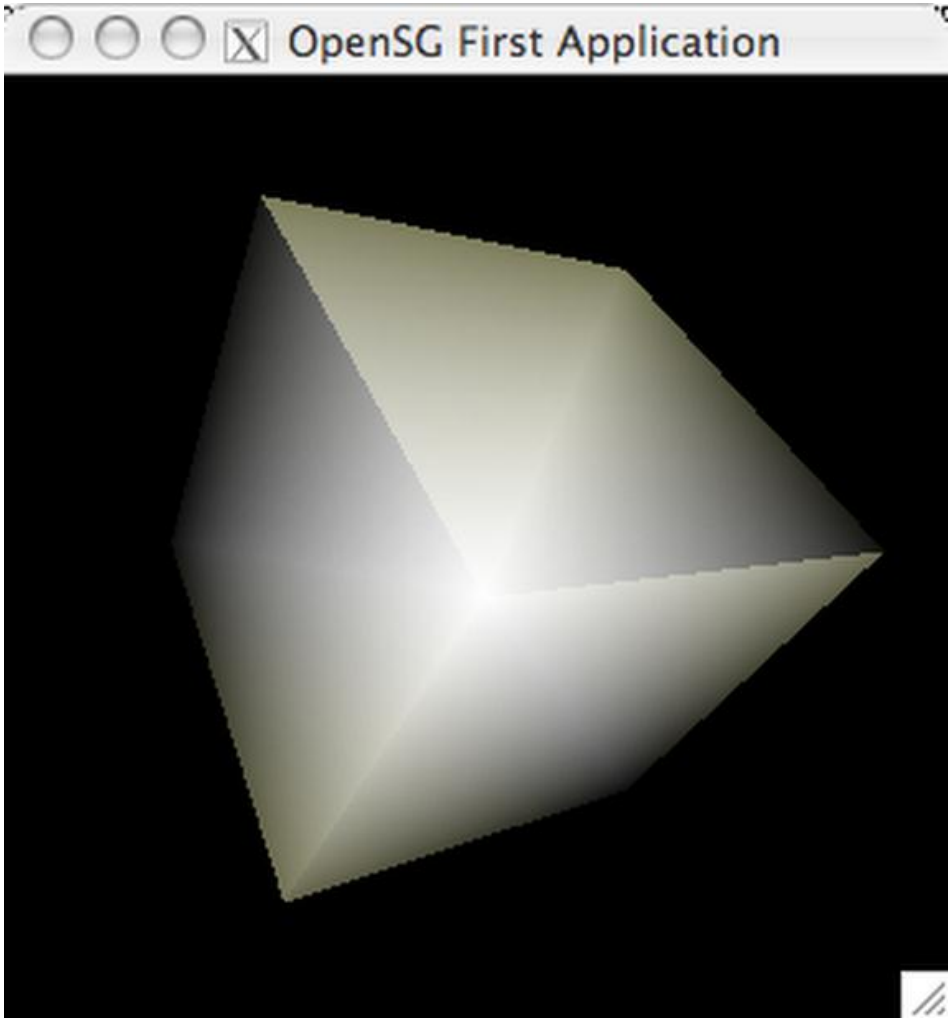
Vertex Normals in OpenGL

Takes care of conversion between object and light/camera coordinates for normal!

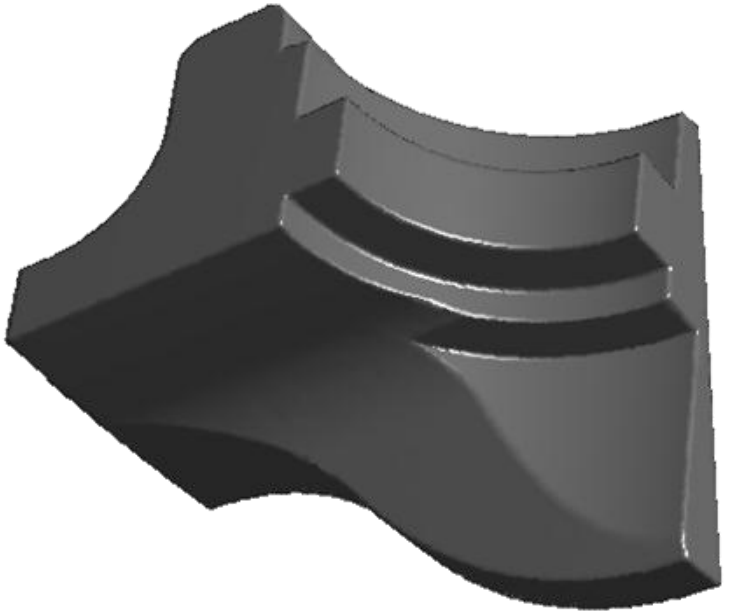
```
glBegin(GL_TRIANGLES);
glNormal(1.f,1.f,1.f);
glVertex3f(0.f,0.f,1.f);
glNormal(1.f,1.f,1.f);
glVertex3f(1.f,0.f,1.f);
glNormal(1.f,1.f,1.f);
glVertex3f(1.f,1.f,1.f);
glEnd();
```

http://graphics.stanford.edu/courses/cs148-11-fall/lectures/rendering.pdf

To Think About…



Do vertex normals on a cube make sense? When should normals be smoothed?



http://www.opensg.org/wiki/Tutorial/OpenSG1/Geometry
http://www.ceremade.dauphine.fr/~peyre/numerical-tour/tours/fastmarching_4_mesh/

Nonphysical/Artistic Shading

Can choose artistic effects to accentuate surface properties rather than simulate light transport.

Nonphotorealistic

[non-foh-toh-ree-*uh*-**lis**-tik]:

Rendering techniques that do not attempt to mimic light transport.

http://en.wikipedia.org/wiki/Fragment_%28computer_graphics%29

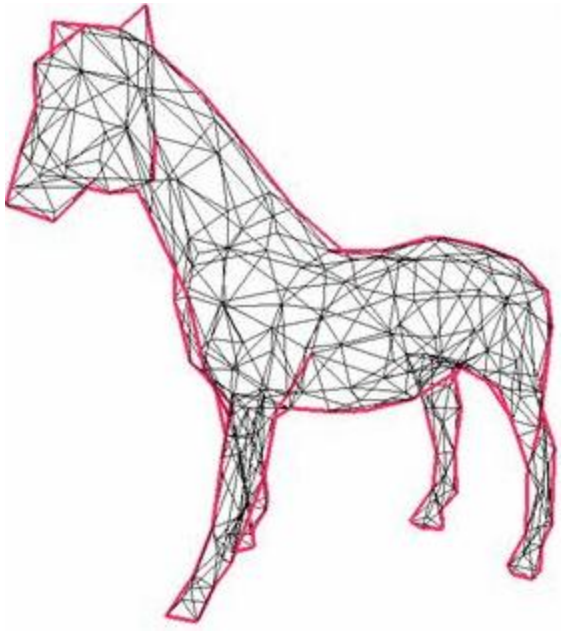
Silhouette Edge Condition

$$(\vec{k}_o \cdot \vec{N}_0)(\vec{k}_o \cdot \vec{N}_1) \cdot \quad 0$$

Face normals on either side of edge change direction relative to eye.

Silhouette Edge Condition

$$(\vec{k}_o \cdot \vec{N}_0)(\vec{k}_o \cdot \vec{N}_1) \cdot \quad 0$$



http://www.cc.gatech.edu/grads/c/catyarr/Silhouette/applet/

Crease Edge Condition

$$\vec{N}_0 \cdot \vec{N}_1 \cdot \quad \text{threshold}$$

Faces adjacent to the edge have normals form a large angle.

Crease Edge Condition

$$\vec{N}_0 \cdot \vec{N}_1 \cdot \quad \text{threshold}$$

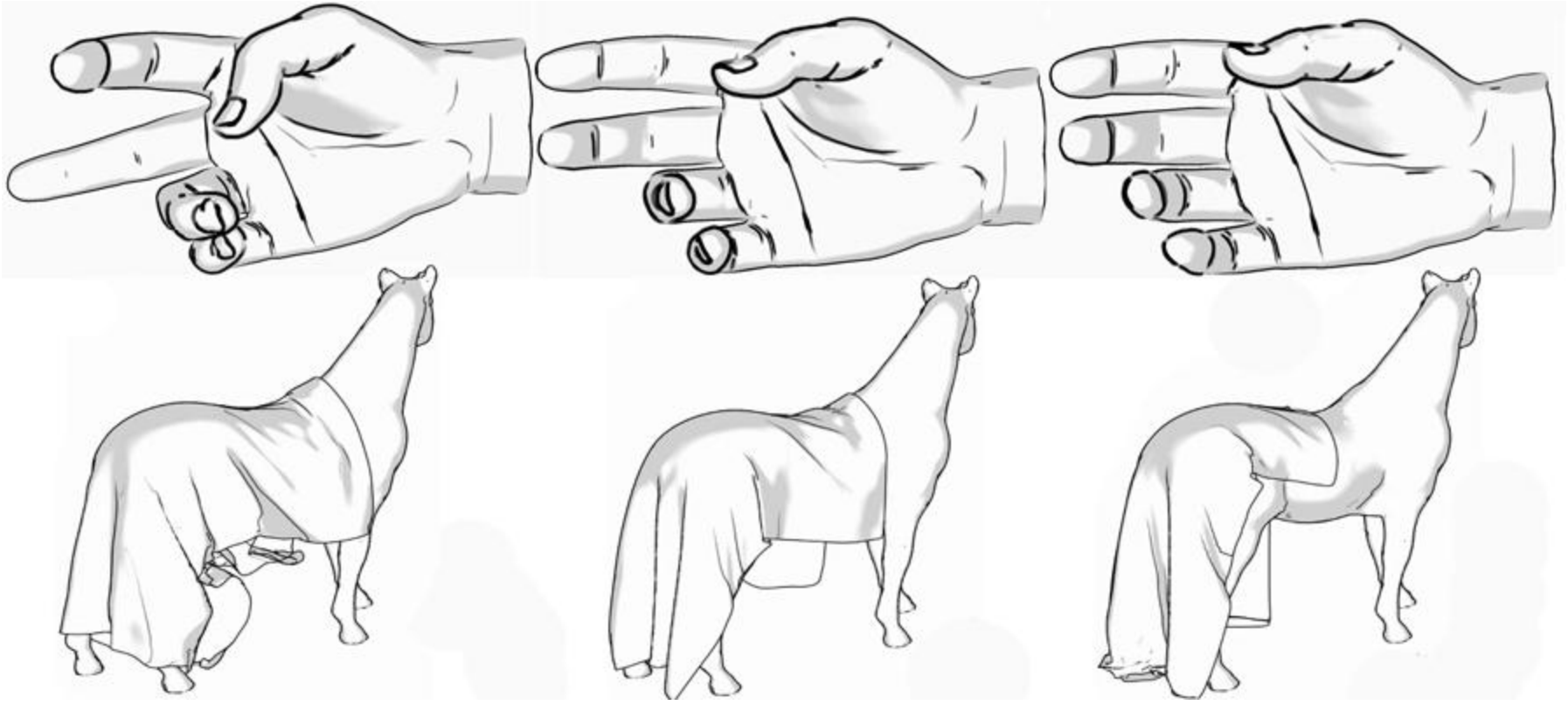
http://prideout.net/blog/p54/Toon2.png

Research: Suggestive Contours

Use curvature to choose better contour lines.

http://www.cs.rutgers.edu/~decarlo/pubs/sg03.pdf

Research: Line Drawing



http://people.cs.umass.edu/~kalo/papers/ddcurvature/

Research: Temporal Coherence

http://www.youtube.com/watch?v=JYo7drrB8WM

http://www.ann.jussieu.fr/~frey/papers/scivi/Cook%20R.L.,%20A%20reflectance%20model%20for%20computer%20graphics.pdf

Shading, Texturing, and Materials

CS 148, Summer 2012

Introduction to Computer Graphics and Imaging