Computer Graphics

Shading

Yu-Shuen Wang CS, NCTU

Illumination

- Local Illumination
- Phong shading model
  - ambient,
  - diffuse,
  - Specular
- Lighting Type
- BRDF

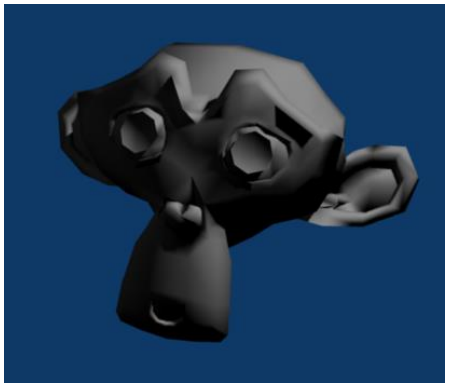
Objectives

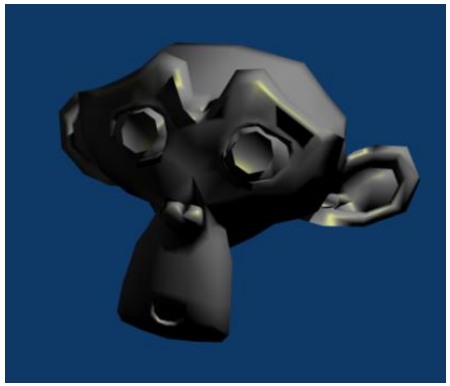
- Learn to shade objects so their images appear three-dimensional
- Introduce the types of light-material interactions
- Build a simple reflection model---the Phong model--- that can be used with real time graphics hardware

Why we need shading

- Suppose we build a model using many polygons and color it with `glColor`. We get something like

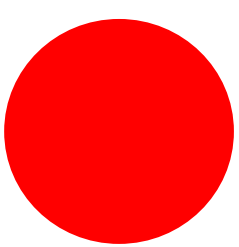
- But we want





Shading

- Why does the image of a real sphere look like

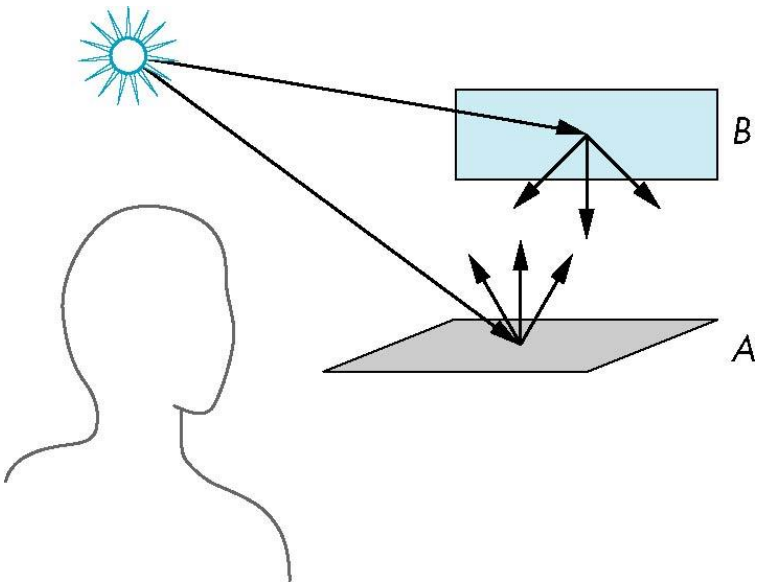


- Light-material interactions cause each point to have a different color or shade
- Need to consider
  - Light sources
  - Material properties
  - Location of viewer
  - Surface orientation

Scattering

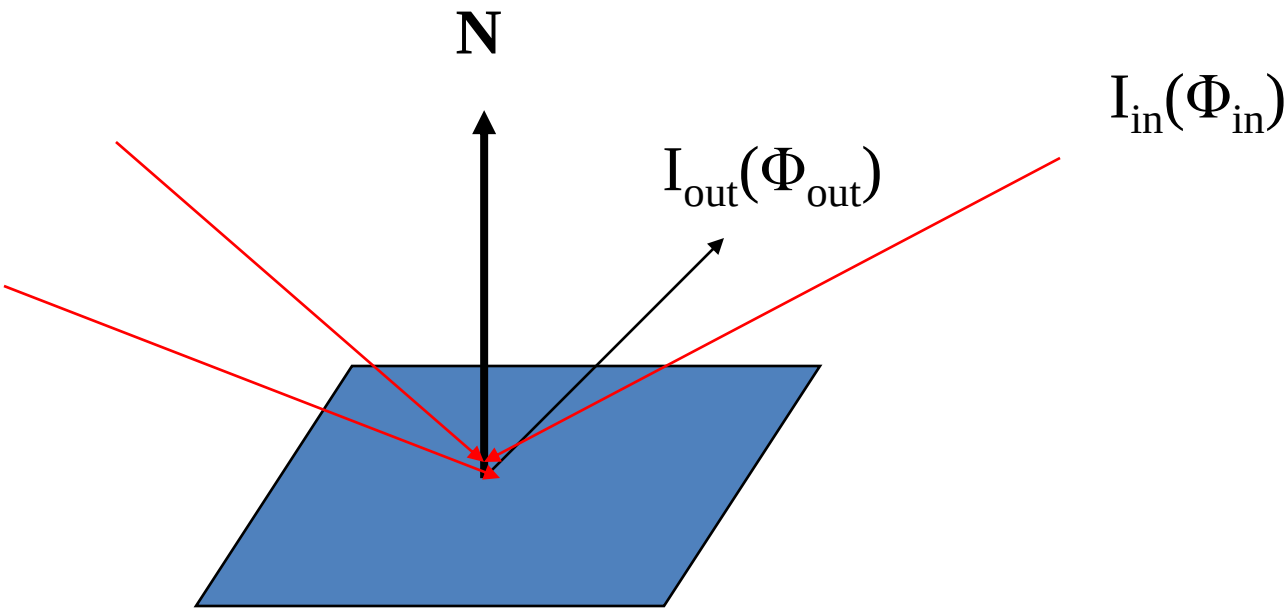
- Light strikes A
  - Some scattered
  - Some absorbed
- Some of scattered light strikes B
  - Some scattered
  - Some absorbed
- Some of this scattered

light strikes A and so on



Rendering Equation (Kajiya 1986)

- Consider a point on a surface



Rendering Equation

$$I_{out}(\Phi_{out}) = E(\Phi_{out}) + \int_{\Omega} R_{bd}(\Phi_{out}, \Phi_{in}) I_{in}(\Phi_{in}) \cos\theta \, d\omega$$

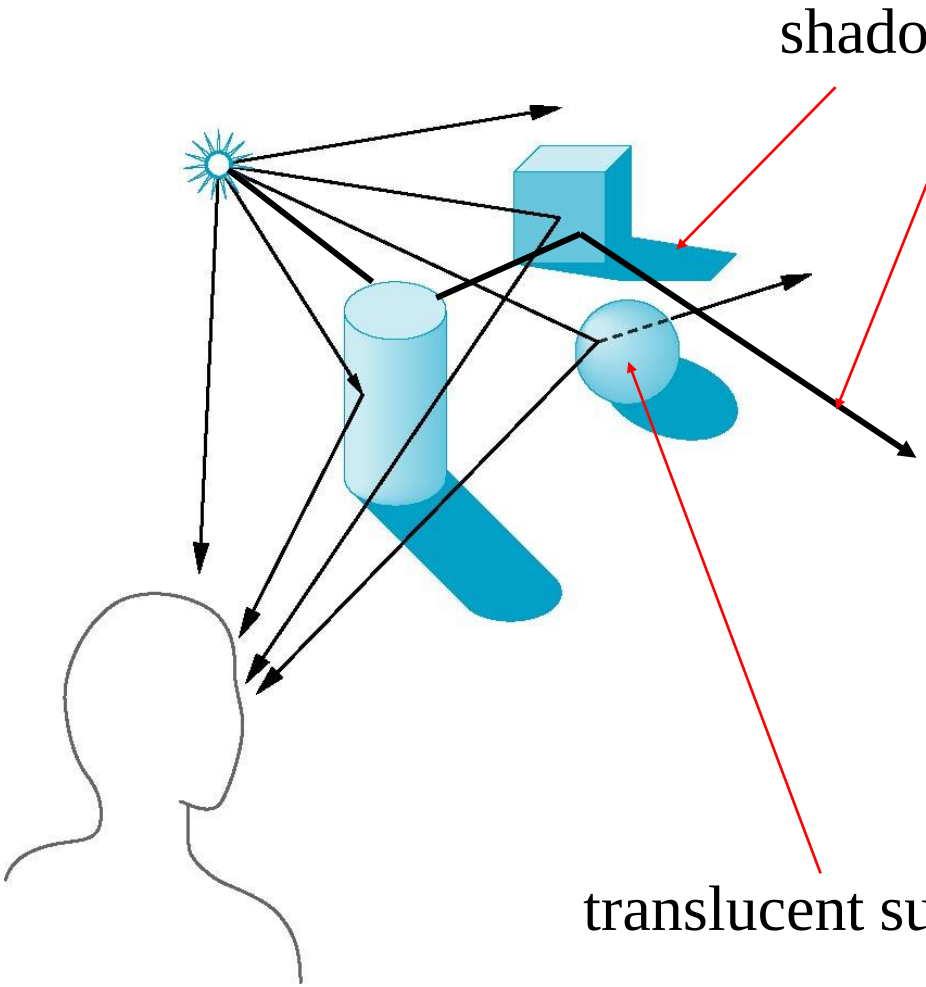
emission

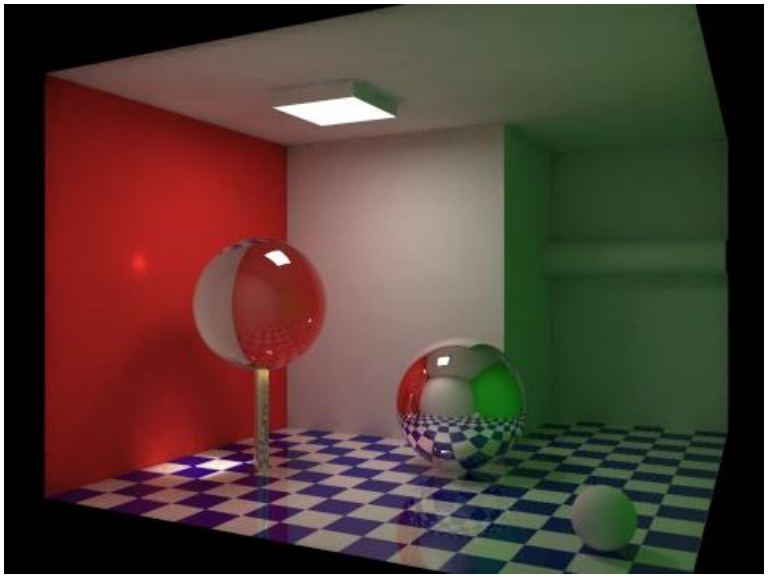
angle between ***Normal*** and ***Φin***

bidirectional reflection coefficient

Note that two angles are in 3D and wavelength is fixed

Global Effects





Rendering Equation

- The infinite scattering and absorption of light can be described by the *rendering equation*
  - Cannot be solved in general
  - Ray tracing is a special case for perfectly reflecting surfaces
- Rendering equation is global and includes
  - Shadows
  - Multiple scattering from object to object

Local vs Global Rendering

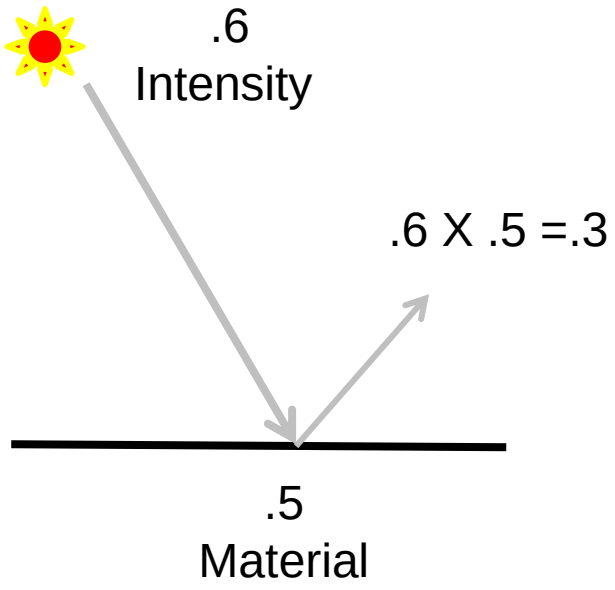
- Correct shading requires a global calculation involving all objects and light sources
  - Incompatible with pipeline model which shades each polygon independently (local rendering)
- However, in computer graphics, especially real time graphics, we are happy if things "look right"
  - Exist many techniques for approximating global effects

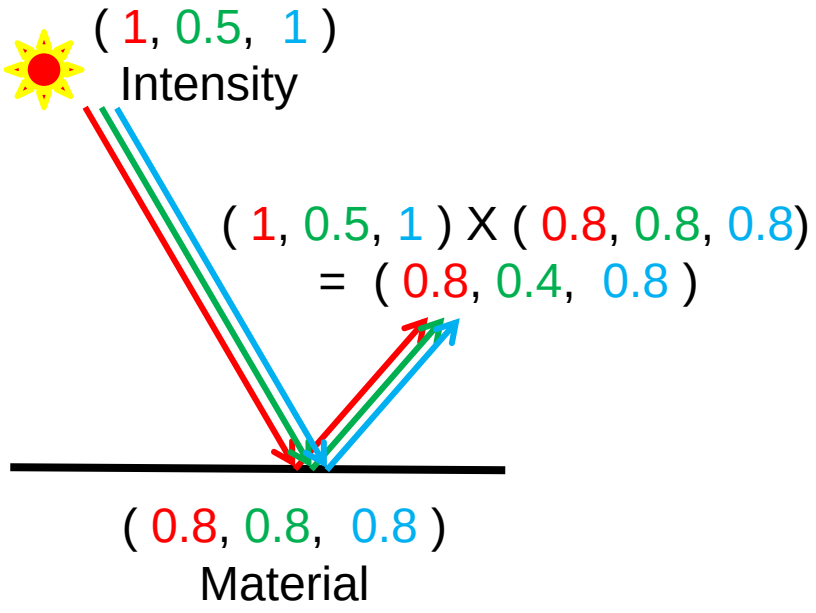
Light-Material Interaction

- Light that strikes an object is partially absorbed and partially scattered (reflected)
- The amount reflected determines the color and brightness of the object
  - A surface appears red under white light because the red component of the light is reflected and the rest is absorbed
- The reflected light is scattered in a manner that depends on the smoothness and orientation of the surface

Material in the world

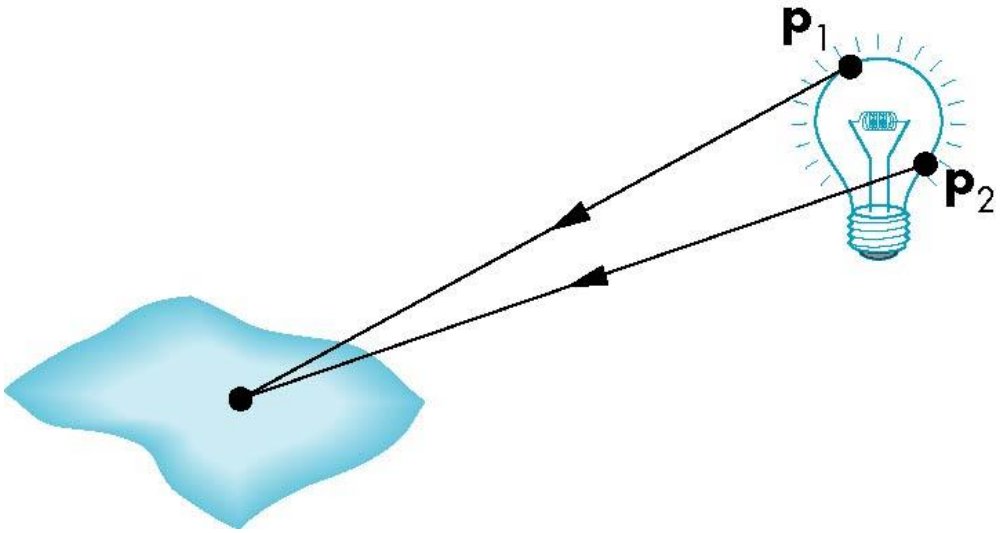
- Material Properties





Light Sources

General light sources are difficult to work with because we must integrate light coming from all points on the source

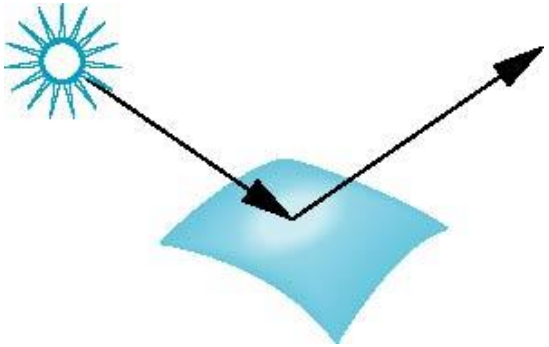


Simple Light Sources

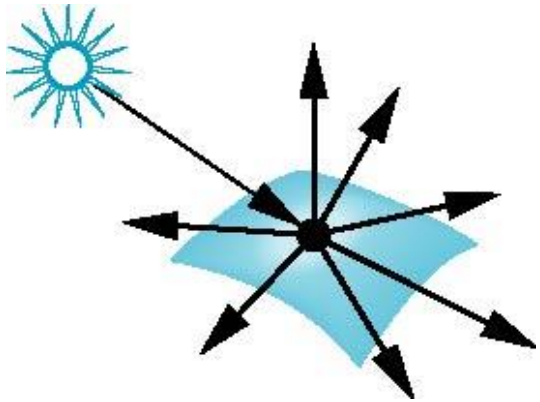
- Point source
  - Model with position and color
  - Distant source = infinite distance away (parallel)
- Spotlight
  - Restrict light from ideal point source
- Ambient light
  - Same amount of light everywhere in scene
  - Can model contribution of many sources and reflecting surfaces

Surface Types

- The smoother a surface, the more reflected light is concentrated in the direction a perfect mirror would reflect the light
- A very rough surface scatters light in all directions



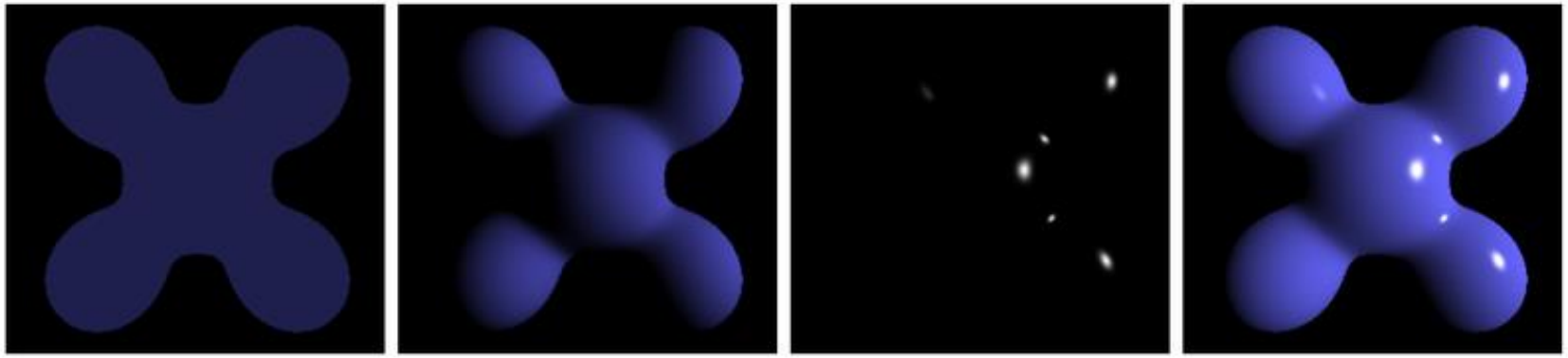
smooth surface



rough surface

Shading Model

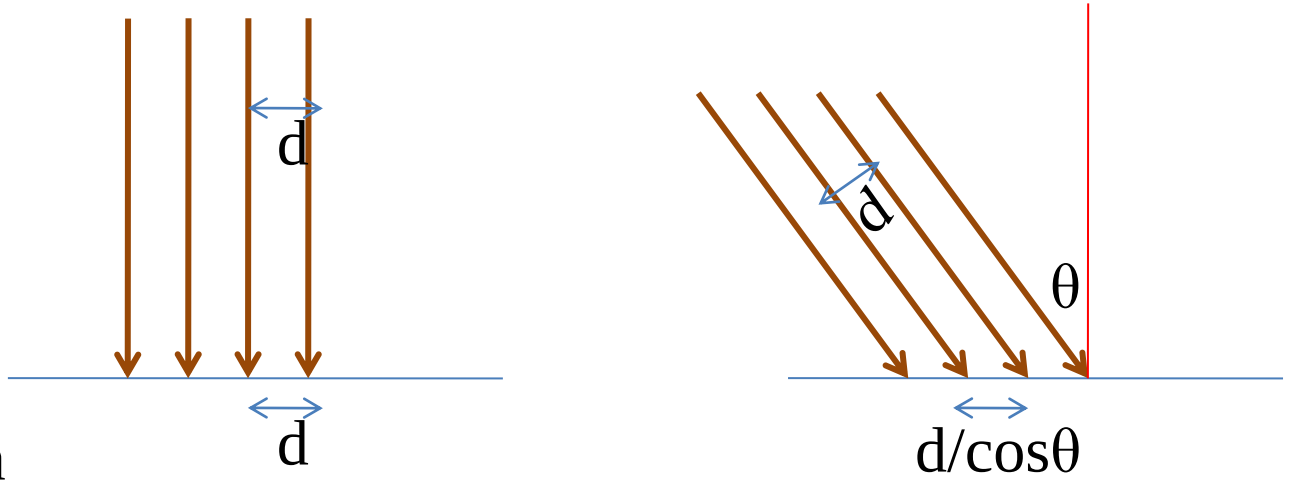
- A simple model that can be computed rapidly
- Has three components
  - Diffuse
  - Specular
  - Ambient



Lambertian Shading (1971)

- Perfectly diffuse reflector
- Light scattered equally in all directions
- Amount of light reflected is proportional to the vertical component of incoming light
  - reflected light ~$\cos \theta_i$
  - $\cos \theta_i = \mathbf{l} \cdot \mathbf{n}$ if vectors normalized
  - There are also three coefficients, k_r, k_b, k_g that show how much of each color component is reflected

Lambertian Surface

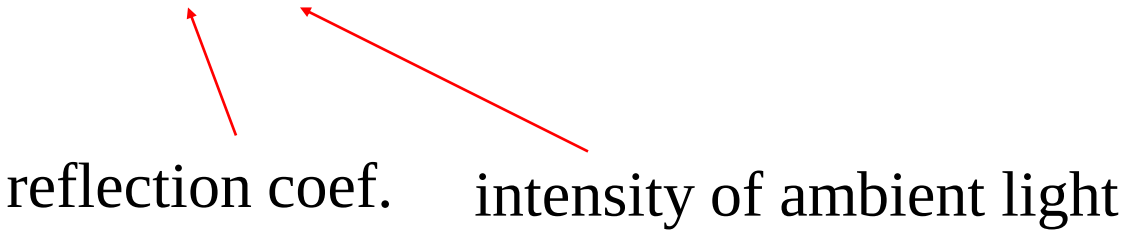


length	d	d/cosθ
intensity	1/d	cosθ / d

Reflected light ~cos θ

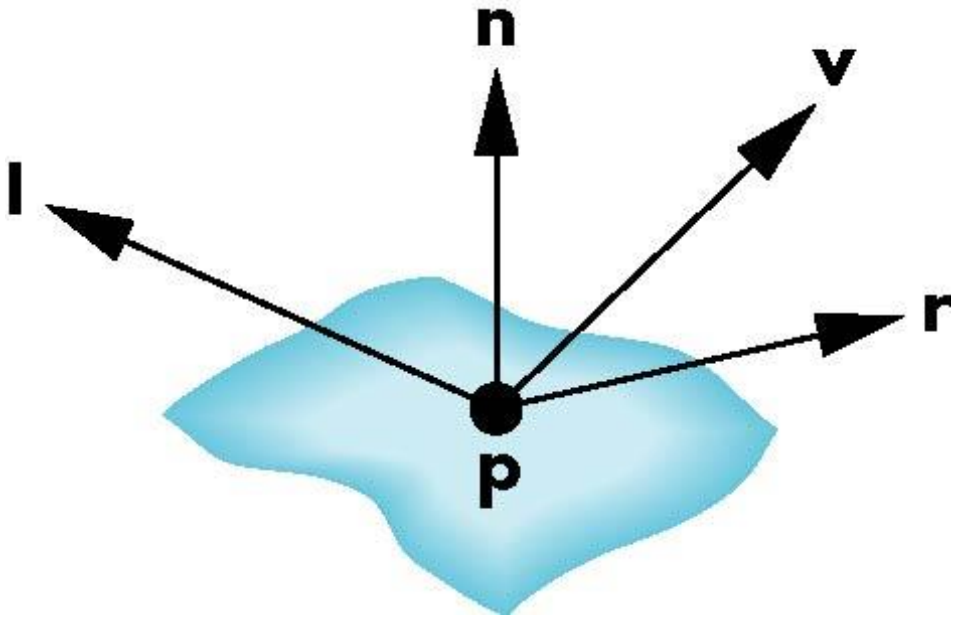
Ambient Light

- Ambient light is the result of multiple interactions between (large) light sources and the objects in the environment
- Amount and color depend on both the color of the light(s) and the material properties of the object
- Add $\mathbf{k_a}\ \mathbf{I_a}$ to ambient term



Phong shading (1973)

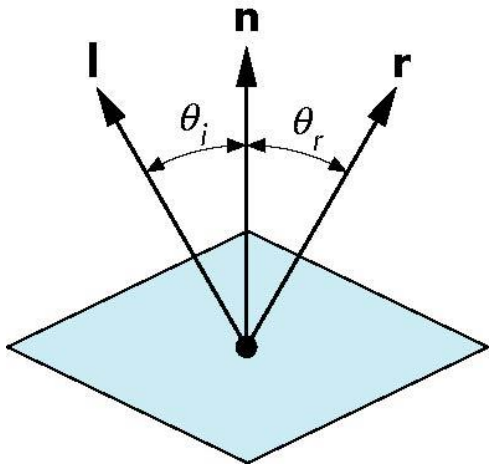
- Uses four vectors
  - To source
  - To viewer
  - Normal
  - Perfect reflector



Ideal Reflector

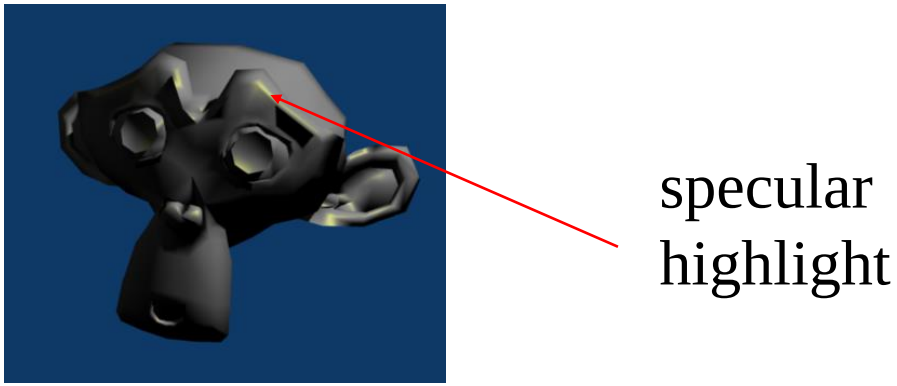
- Normal is determined by local orientation
- Angle of incidence = angle of relection
- The three vectors must be coplanar

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



Specular Surfaces

- Most surfaces are neither ideal diffuse nor perfectly specular (ideal reflectors)
- Smooth surfaces show specular highlights due to incoming light being reflected in directions concentrated close to the direction of a perfect reflection



Modeling Specular Relections

- Phong proposed using a term that dropped off as the angle between the viewer and the ideal reflection increased

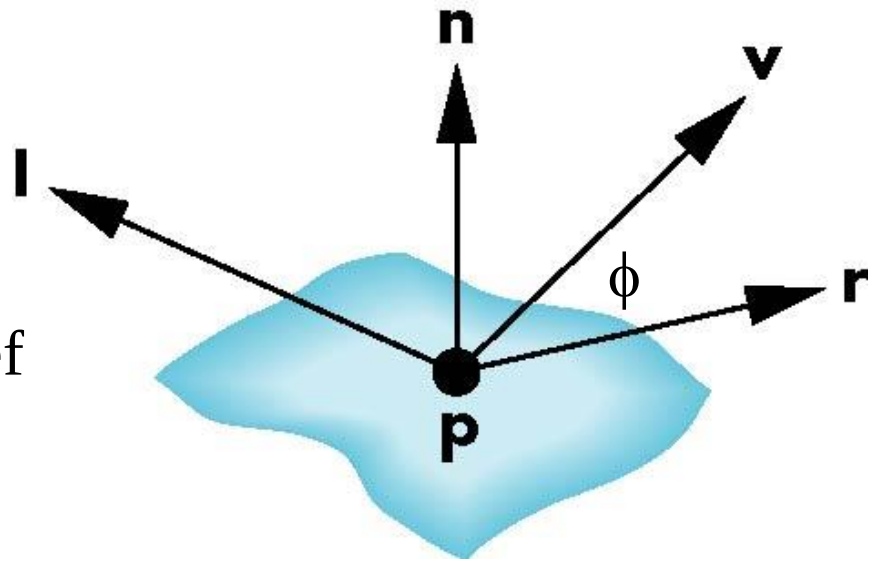
$$\mathbf{I}_r \sim k_s \mathbf{I} \cos^{\alpha}\phi$$

reflected intensity

reflect coef

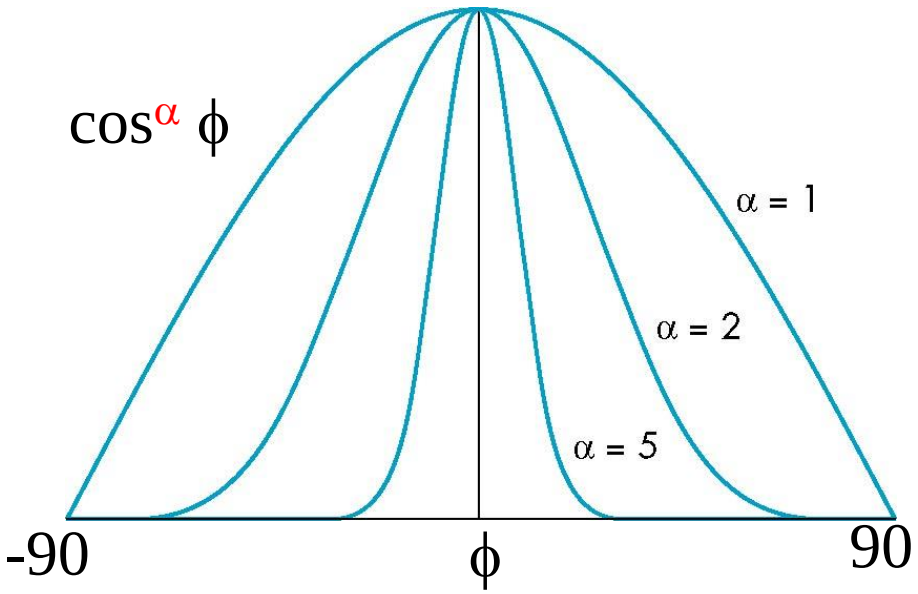
incoming intensity

shininess coef



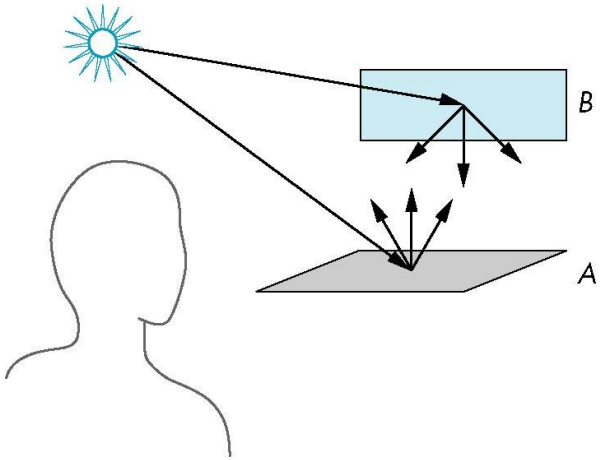
The Shininess Coefficient

- Values of α between 100 and 200 correspond to metals
- Values between 5 and 10 give surface that look like plastic



Distance Terms

- The light from a point source that reaches a surface is inversely proportional to the square of the distance between them
- We can add a factor of the form $1/(a + bd + cd^2)$ to the diffuse and specular terms
- The constant and linear terms soften the effect of the point source



Light Sources

- In the Phong Model, we add the results from each light source
- Each light source has separate ***diffuse***, ***specular***, and ***ambient*** terms to allow for maximum flexibility even though this form does not have a physical justification
- Separate red, green and blue components
- Hence, 9 coefficients for each point source
  - $I_{dr}, I_{dg}, I_{db}, \quad I_{sr}, I_{sg}, I_{sb}, \quad I_{ar}, I_{ag}, I_{ab}$

Material Properties

- Material properties match light source properties
  - Nine absorption coefficients
    - Diffuse: k_{dr}, k_{dg}, k_{db},
    - Speculr: k_{sr}, k_{sg}, k_{sb},
    - Ambient: k_{ar}, k_{ag}, k_{ab}
  - Shininess coefficient α

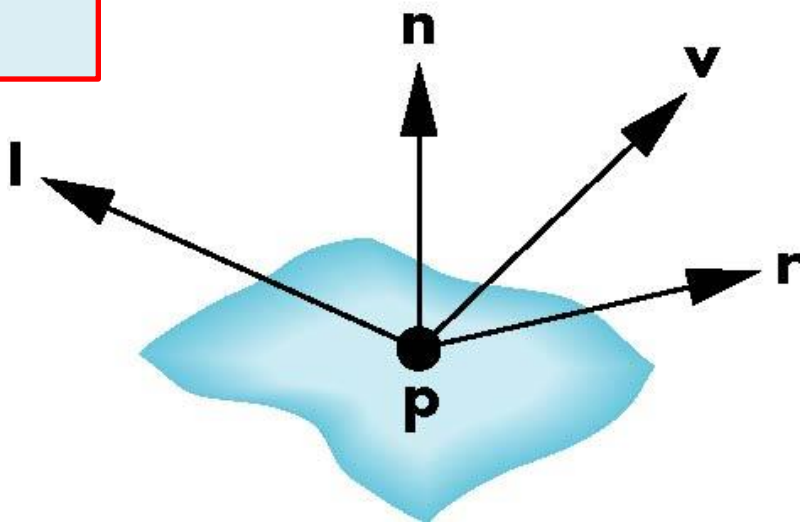
Adding up the Components

For each light source and each color component, the Phong model can be written (without the distance terms) as

$$I = k_d I_d \, \mathbf{l} \cdot \mathbf{n} + k_s I_s (\mathbf{v} \cdot \mathbf{r})^{\alpha} + k_a I_a$$

diffuse | specular | ambient

For each color component we add contributions from all sources



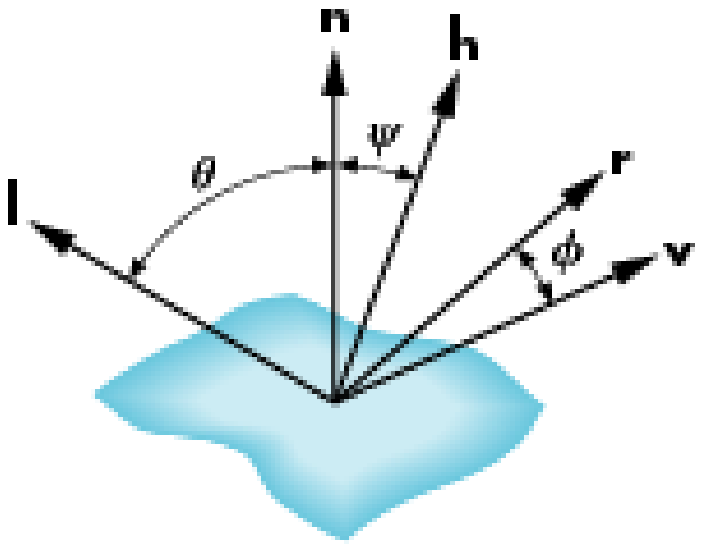
Modified Phong Model (Blinn-Phong)

- The specular term in the Phong model is problematic because it requires the calculation of a new reflection vector and view vector for each vertex
- Blinn suggested an approximation using the halfway vector that is more efficient

The Halfway Vector

- ***h*** is normalized vector halfway between ***l*** and ***v***

$$\mathbf{h} = (\mathbf{l} + \mathbf{v}) / |\mathbf{l} + \mathbf{v}|$$



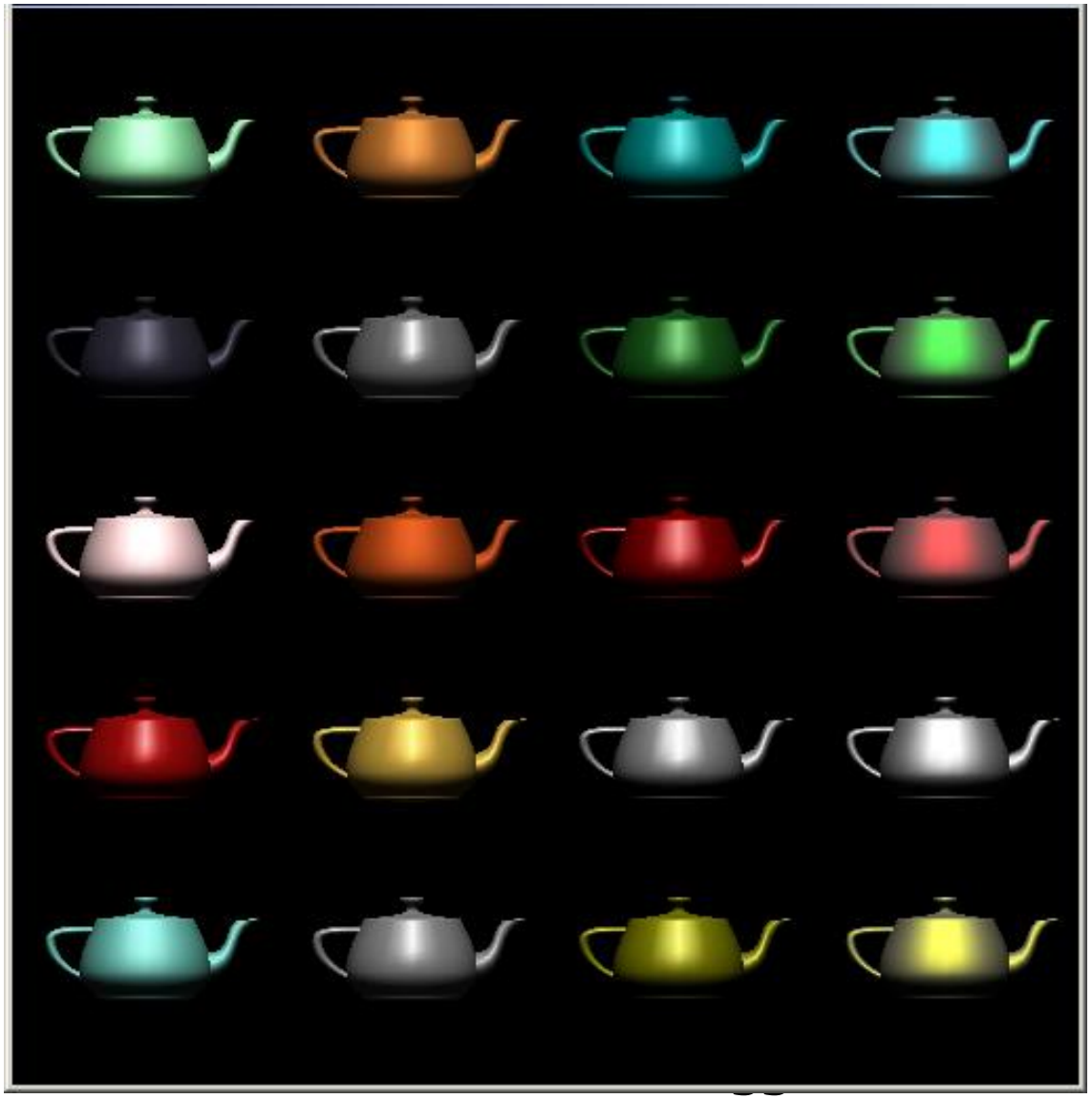
$$2\psi = \phi$$

Using the halfway vector

- Replace $(\mathbf{v} \cdot \mathbf{r})^{\alpha}$ by $(\mathbf{n} \cdot \mathbf{h})^{\beta}$
- β is chosen to match shineness
- Note that halway angle is half of angle between $\mathbf{r}$ and $\mathbf{v}$ if vectors are coplanar
- Resulting model is known as the modified Phong or Blinn lighting model
  - Specified in OpenGL standard

Example

Only differences in these teapots are the parameters in the modified Phong model



Computation of Vectors

- $\mathbf{l}$ and $\mathbf{v}$ are specified by the application
- Can compute $\mathbf{r}$ from $\mathbf{l}$ and $\mathbf{n}$
- Determining $\mathbf{n}$ is problematic
- For simple surfaces: it can be determined, but how we determine $\mathbf{n}$ differs the appearance of a surface
- OpenGL leaves determination of normal to application
  - Exception for GLU quadrics and Bezier surfaces (Chapter 11)

Local Illumination options

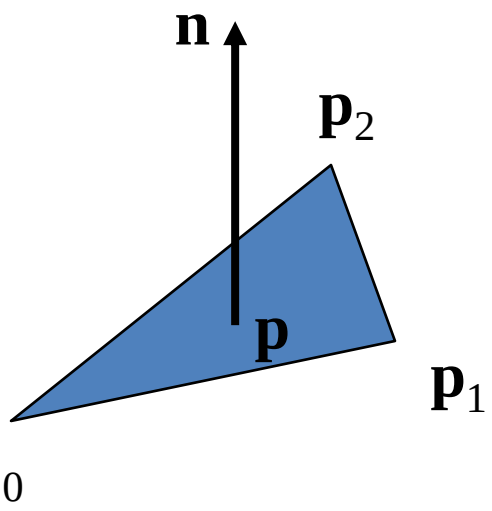
- Discuss polygonal shading
  - Flat
  - Smooth
  - Gouraud
  - Phong
- Lighting

Normal on a Triangle

plane $\mathbf{n} \cdot (\mathbf{p} - \mathbf{p}_0) = 0$

$\mathbf{n} = (\mathbf{p}_1 - \mathbf{p}_0) \times (\mathbf{p}_2 - \mathbf{p}_1)$

normalize $\mathbf{n} \leftarrow \mathbf{n} / |\mathbf{n}|$

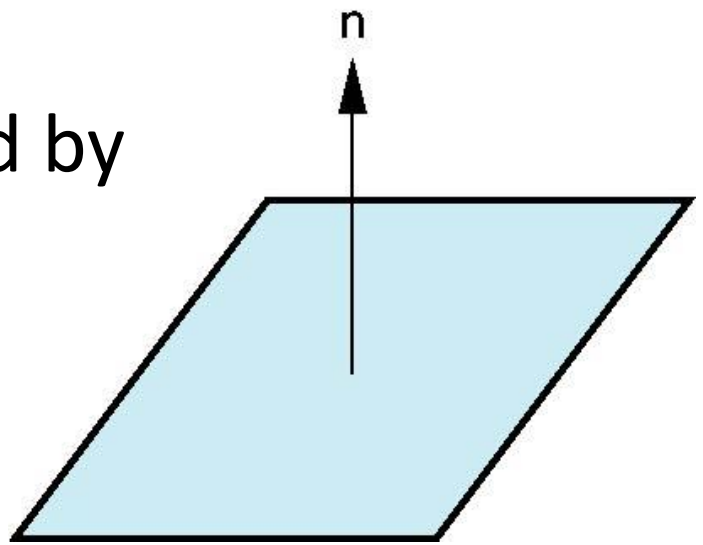


Note that right-hand rule determines outward face

Plane Normals

- Equation of plane: $ax+by+cz+d = 0$
- From Chapter 4 we know that plane is determined by three points p_0, p_2, p_3 or normal $\mathbf{n}$ and p_0
- Normal can be obtained by

$$\mathbf{n} = (p_1-p_0) \times (p_2-p_1)$$

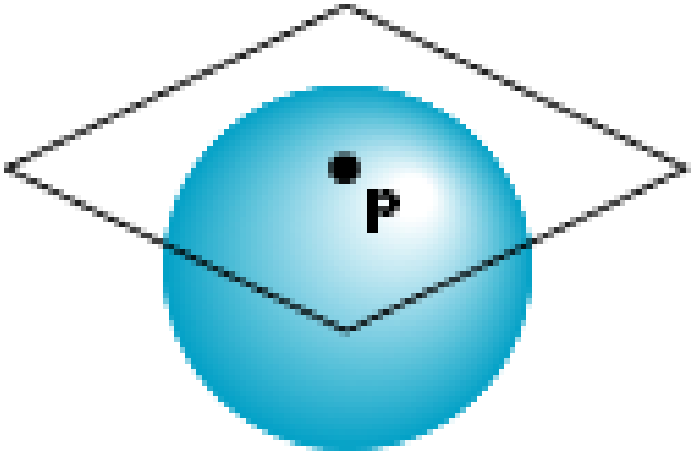


General Case

- We can compute parametric normals for other simple cases
  - Quadrics
  - Parameteric polynomial surfaces

Normal on a Sphere

- Implicit function $f(x, y, z)=0$
- Sphere $f(\mathbf{p})=\mathbf{p}\cdot\mathbf{p}-1$
- $n = \mathbf{p}$

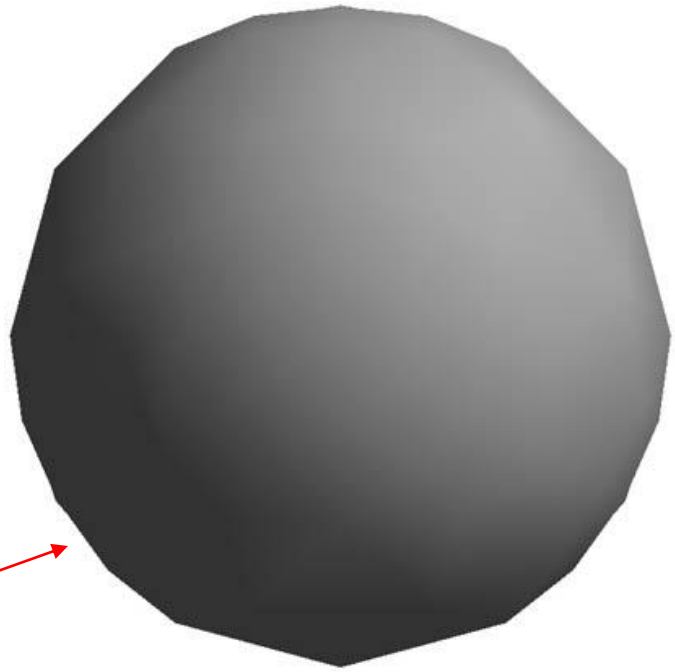


Polygon Normals

- Polygons have a single normal
  - Shades at the vertices as computed by the Phong model can be almost same
  - Identical for a distant viewer (default) or if there is no specular component
- Consider model of sphere
- Get different normals at each vertex although this concept is not quite correct mathematically

Smooth Shading

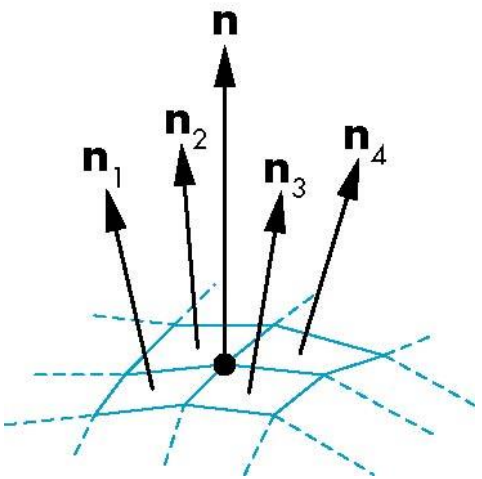
- We can set a new normal at each vertex
- Easy for sphere model
  - If centered at origin $\mathbf{n} = \mathbf{p}$
- Now smooth shading works
- Note *silhouette edge*



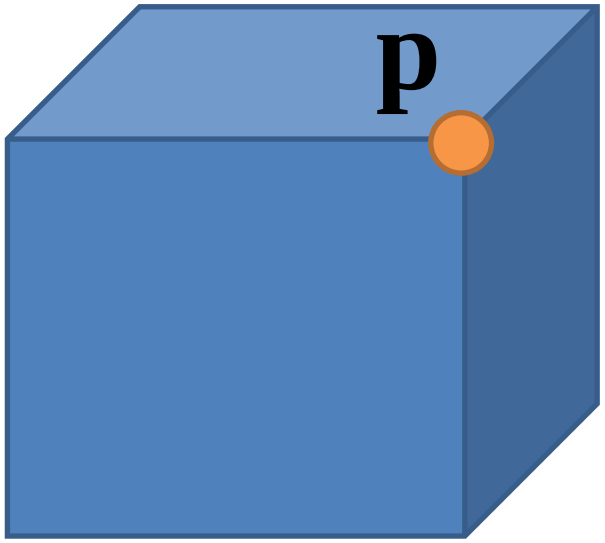
Mesh Shading

- The previous example is not general because we knew the normal at each vertex analytically
- For polygonal models, Gouraud proposed we use the average of the normals around a mesh vertex

$$\mathbf{n} = (\mathbf{n}_1+\mathbf{n}_2+\mathbf{n}_3+\mathbf{n}_4)/\,|\mathbf{n}_1+\mathbf{n}_2+\mathbf{n}_3+\mathbf{n}_4|$$



Normal on a cube



Gouraud and Phong Shading

- Gouraud Shading
  - Find average normal at each vertex (vertex normals)
  - Apply modified Phong model at each vertex
  - Interpolate vertex shades across each polygon
- Phong shading
  - Find vertex normals
  - Interpolate vertex normals across edges
  - Interpolate edge normals across polygon
  - Apply modified Phong model at each fragment

Comparison

- If the polygon mesh approximates surfaces with a high curvatures, Phong shading may look smooth while Gouraud shading may show edges
- Both need data structures to represent meshes so we can obtain vertex normals

Objectives

- Introduce the OpenGL shading functions
- Discuss polygonal shading
  - Flat
  - Smooth
  - Gouraud

Steps in OpenGL shading

1. Enable shading and select model
2. Specify normals
3. Specify material properties
4. Specify lights

Normals

- In OpenGL the normal vector is part of the state
- Set by **`glNormal*()`**
  - **`glNormal3f(x, y, z);`**
  - **`glNormal3fv(p);`**
- Usually we want to set the normal to have unit length so cosine calculations are correct
  - Length can be affected by transformations
  - Note that scaling does not preserved length
  - **`glEnable(GL_NORMALIZE)`** allows for autonormalization at a performance penalty

Enabling Shading

- Shading calculations are enabled by
  - **`glEnable(GL_LIGHTING)`**
  - Once lighting is enabled, glColor() ignored
- Must enable each light source individually
  - **`glEnable(GL_LIGHTi)`** i=0,1…..
- Can choose light model parameters
  - **`glLightModeli(parameter, GL_TRUE)`**
    - **`GL_LIGHT_MODEL_LOCAL_VIEWER`** does not simplify distant viewer assumption in calculation
    - **`GL_LIGHT_MODEL_TWO_SIDED`** shades both sides of polygons independently

Defining a Point Light Source

- For each light source, we can set an RGBA for the diffuse, specular, and ambient components, and for the position

```
GLfloat diffuse0[]={1.0, 0.0, 0.0, 1.0};
GLfloat ambient0[]={1.0, 0.0, 0.0, 1.0};
GLfloat specular0[]={1.0, 0.0, 0.0, 1.0};
Glfloat light0_pos[]={1.0, 2.0, 3,0, 1.0};

glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
glLightv(GL_LIGHT0, GL_POSITION, light0_pos);
glLightv(GL_LIGHT0, GL_AMBIENT, ambient0);
glLightv(GL_LIGHT0, GL_DIFFUSE, diffuse0);
glLightv(GL_LIGHT0, GL_SPECULAR, specular0);
```

Distance

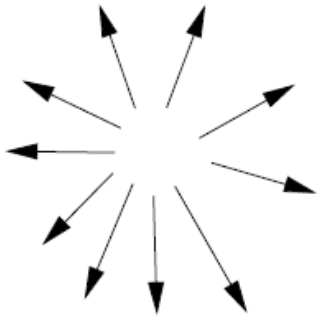
- The coefficients in the distance terms are by default k_c=1.0 (constant terms), k_l=k_q=0.0 (linear and quadratic terms). Change by

$$\text{attenuation factor} = \frac{1}{k_c + k_l d + k_q d^2}$$

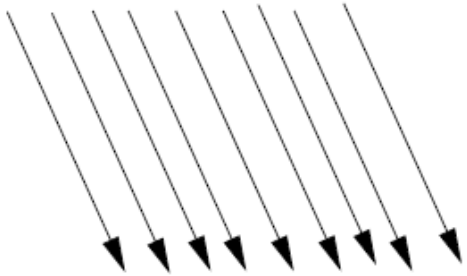
```
glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, 2.0);
glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, 1.0);
glLightf(GL_LIGHT0, GL_QUADRATIC_ATTENUATION, 0.5);
```

Direction

- The source colors are specified in RGBA
- The position is given in homogeneous coordinates
  - If w =1.0, we are specifying a finite location
  - If w =0.0, we are specifying a parallel source with the given direction vector



point



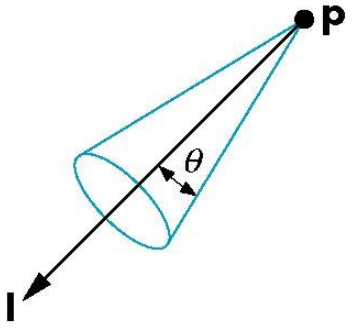
parallel

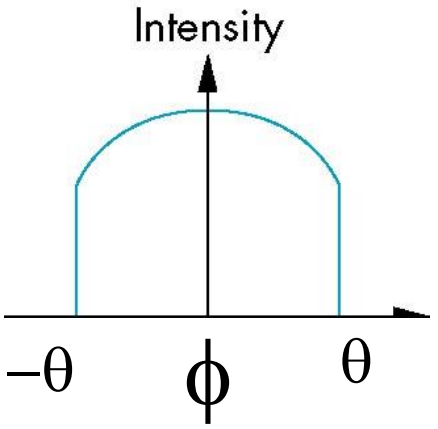
GL_POSITION

GL_SPOT_DIRECTION

Spotlights

- Use **`glLightv`** to set
  - Direction **`GL_SPOT_DIRECTION`**
  - Cutoff **`GL_SPOT_CUTOFF`**
  - Attenuation **`GL_SPOT_EXPONENT`**
    - Proportional to $\cos^{\alpha}\phi$





Global Ambient Light

- Ambient light depends on color of light sources
- OpenGL also allows a global ambient term that is often helpful for testing
  - **`glLightModelfv(GL_LIGHT_MODEL_AMBIENT, global_ambient)`**

Moving Light Sources

- Light sources are geometric objects whose positions or directions are affected by the model-view matrix
- Depending on where we place the position (direction) setting function, we can
  - Move the light source(s) with the object(s)
  - Fix the object(s) and move the light source(s)
  - Fix the light source(s) and move the object(s)
  - Move the light source(s) and object(s) independently

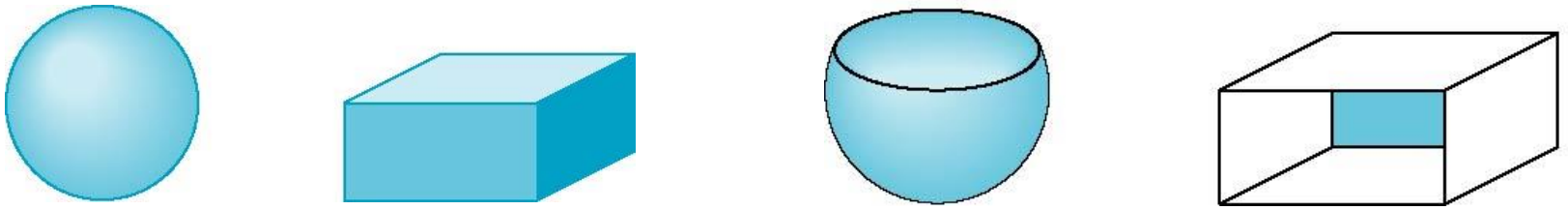
Material Properties

- Material properties are also part of the OpenGL state and match the terms in the modified Phong model
- Set by **`glMaterialv()`**

```
GLfloat ambient[] = {0.2, 0.2, 0.2, 1.0};
GLfloat diffuse[] = {1.0, 0.8, 0.0, 1.0};
GLfloat specular[] = {1.0, 1.0, 1.0, 1.0};
GLfloat shine = 100.0
glMaterialf(GL_FRONT, GL_AMBIENT, ambient);
glMaterialf(GL_FRONT, GL_DIFFUSE, diffuse);
glMaterialf(GL_FRONT, GL_SPECULAR, specular);
glMaterialf(GL_FRONT, GL_SHININESS, shine);
```

Front and Back Faces

- The default is shade only front faces which works correctly for convex objects

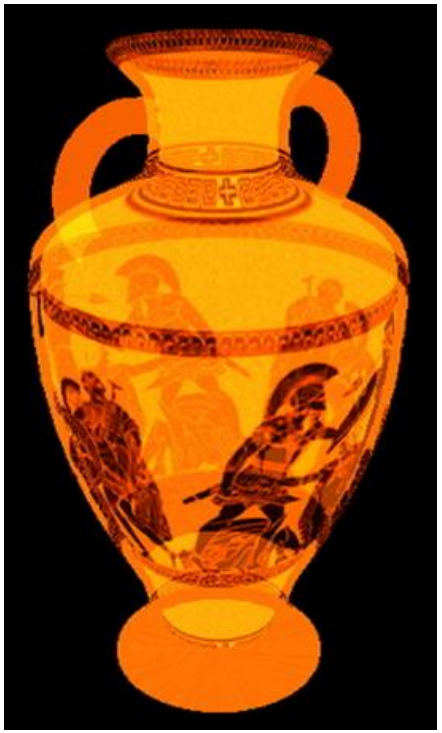


back faces not visible

back faces visible

Emissive Term

- We can simulate a light source in OpenGL by giving a material an emissive component
- This component is unaffected by any sources or transformations



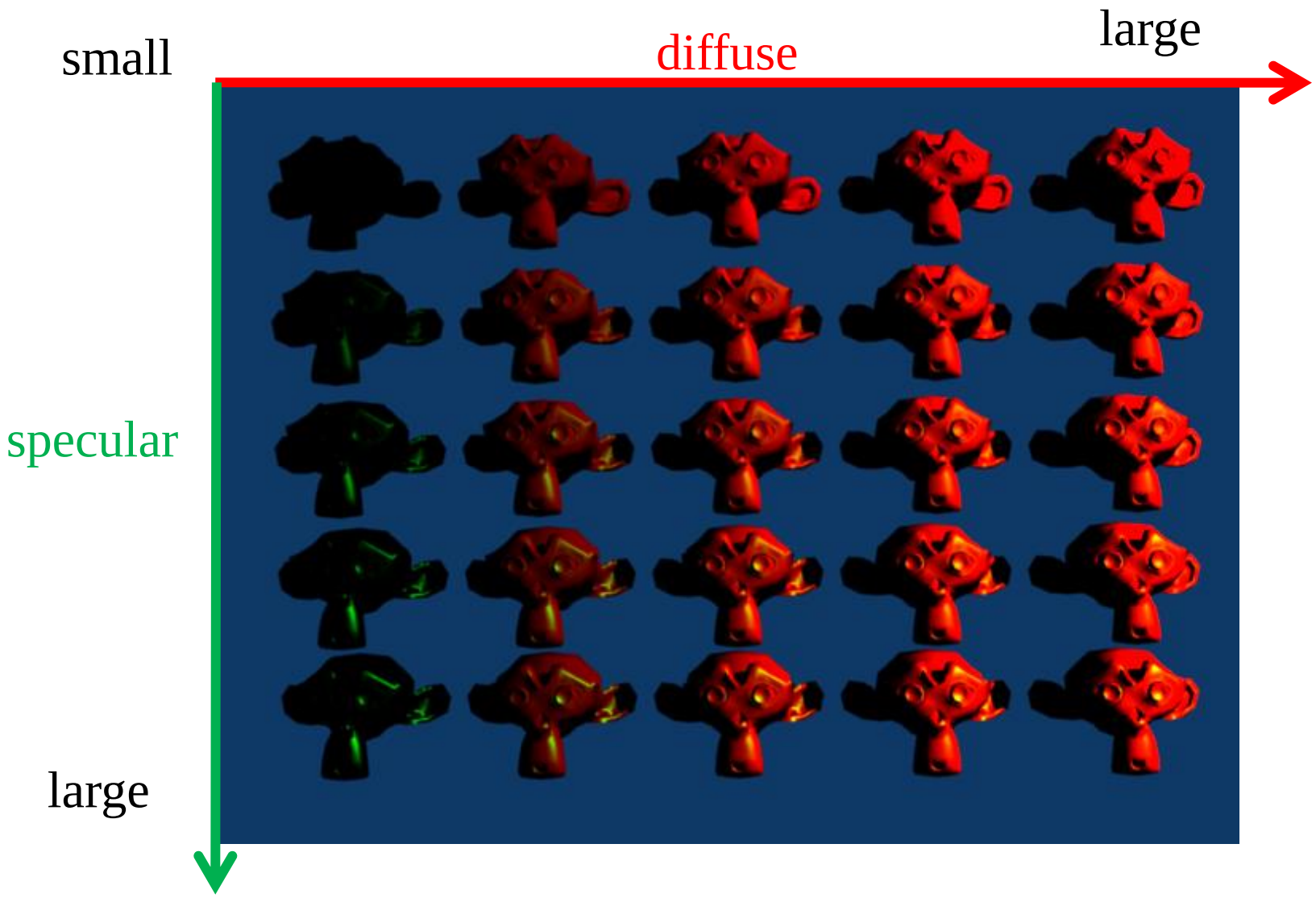
```
GLfloat emission[] = 0.0, 0.3, 0.3, 1.0);
glMaterialf(GL_FRONT, GL_EMISSION, emission);
```

Transparency

- Material properties are specified as RGBA values
- The A value can be used to make the surface translucent
- The default is that all surfaces are opaque regardless of A
- Later we will enable blending and use this feature

Polygonal Shading

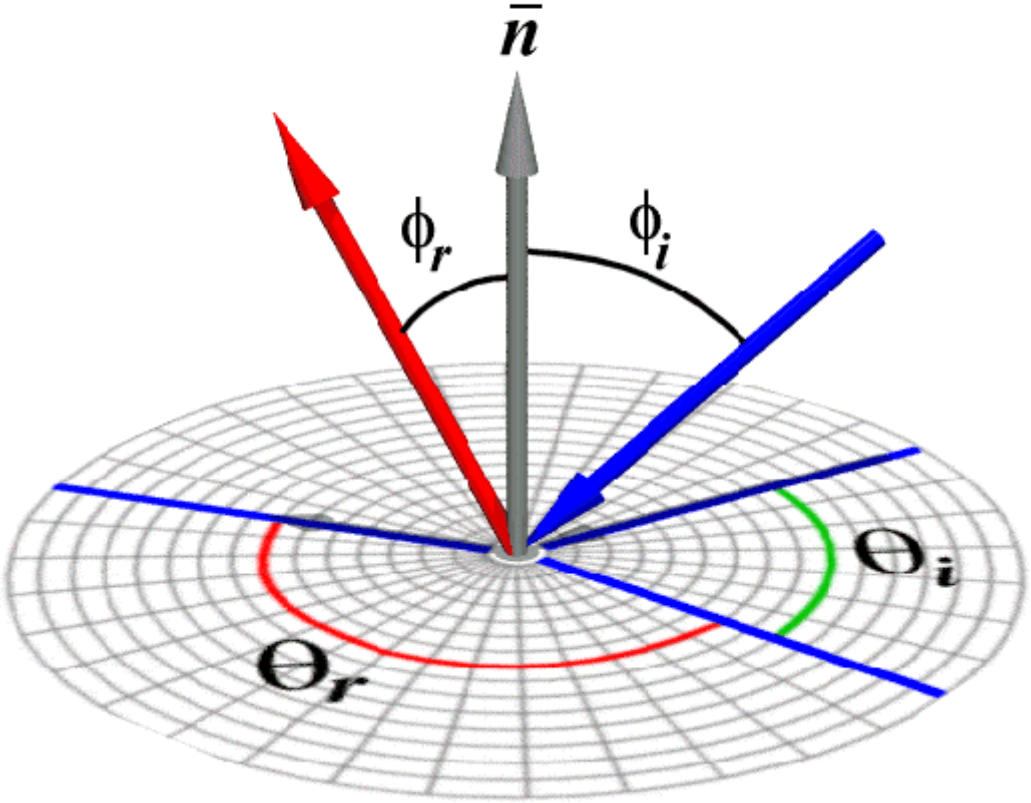
- Shading calculations are done for each vertex
  - Vertex colors become vertex shades
- By default, vertex shades are interpolated across the polygon
  - `glShadeModel(GL_SMOOTH);`
- If we use `glShadeModel(GL_FLAT);` the color at the first vertex will determine the shade of the whole polygon


small
diffuse
large
specular
large

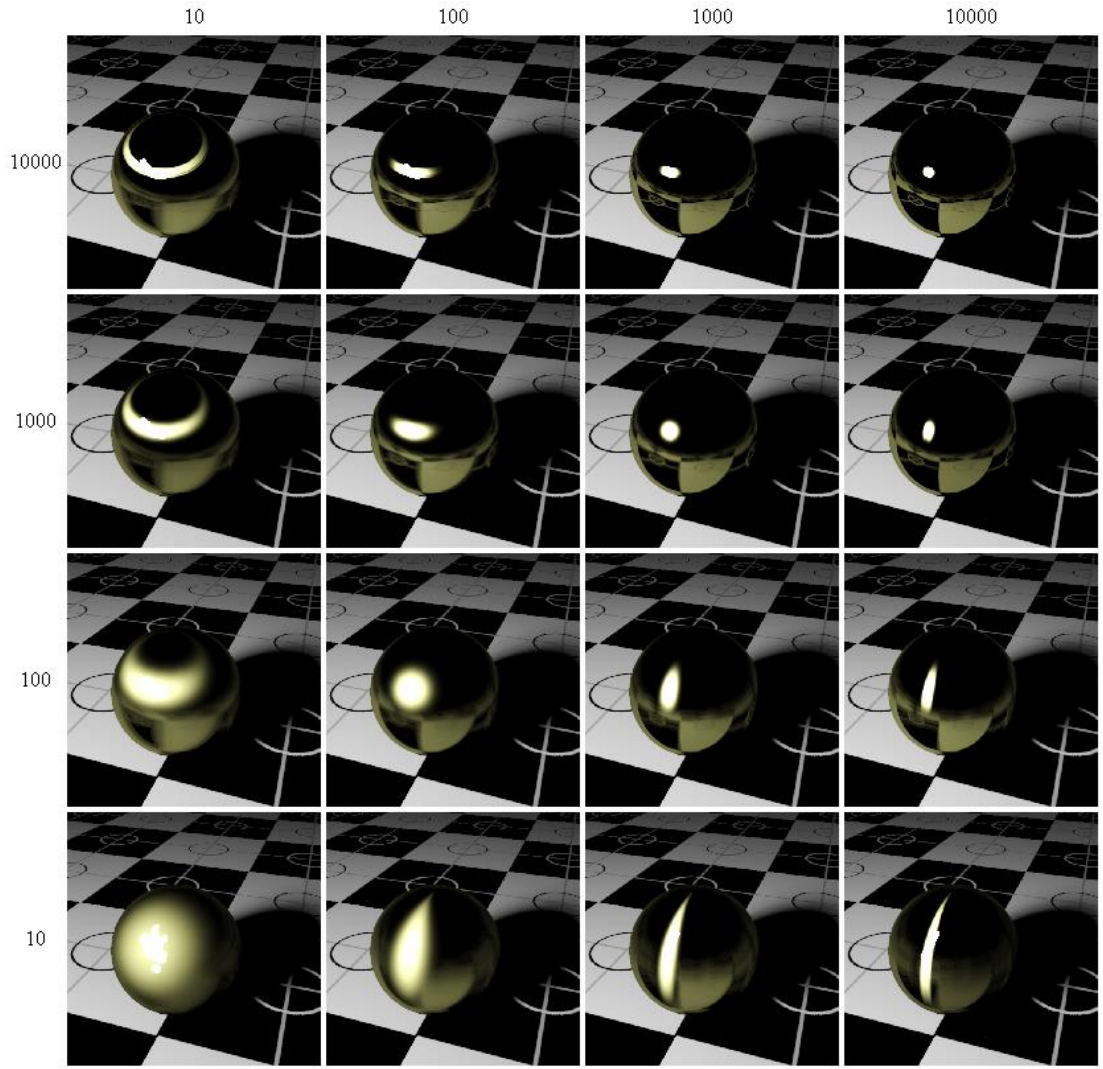
BRDF

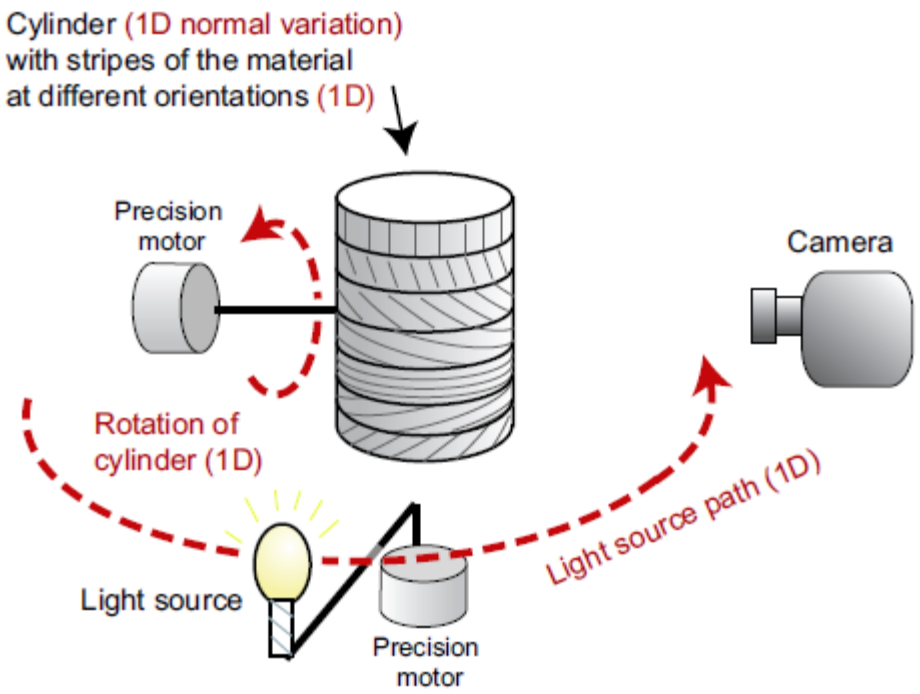
Bidirectional reflectance distribution function

$\rho(\theta_r, \phi_r, \theta_i, \phi_i)$



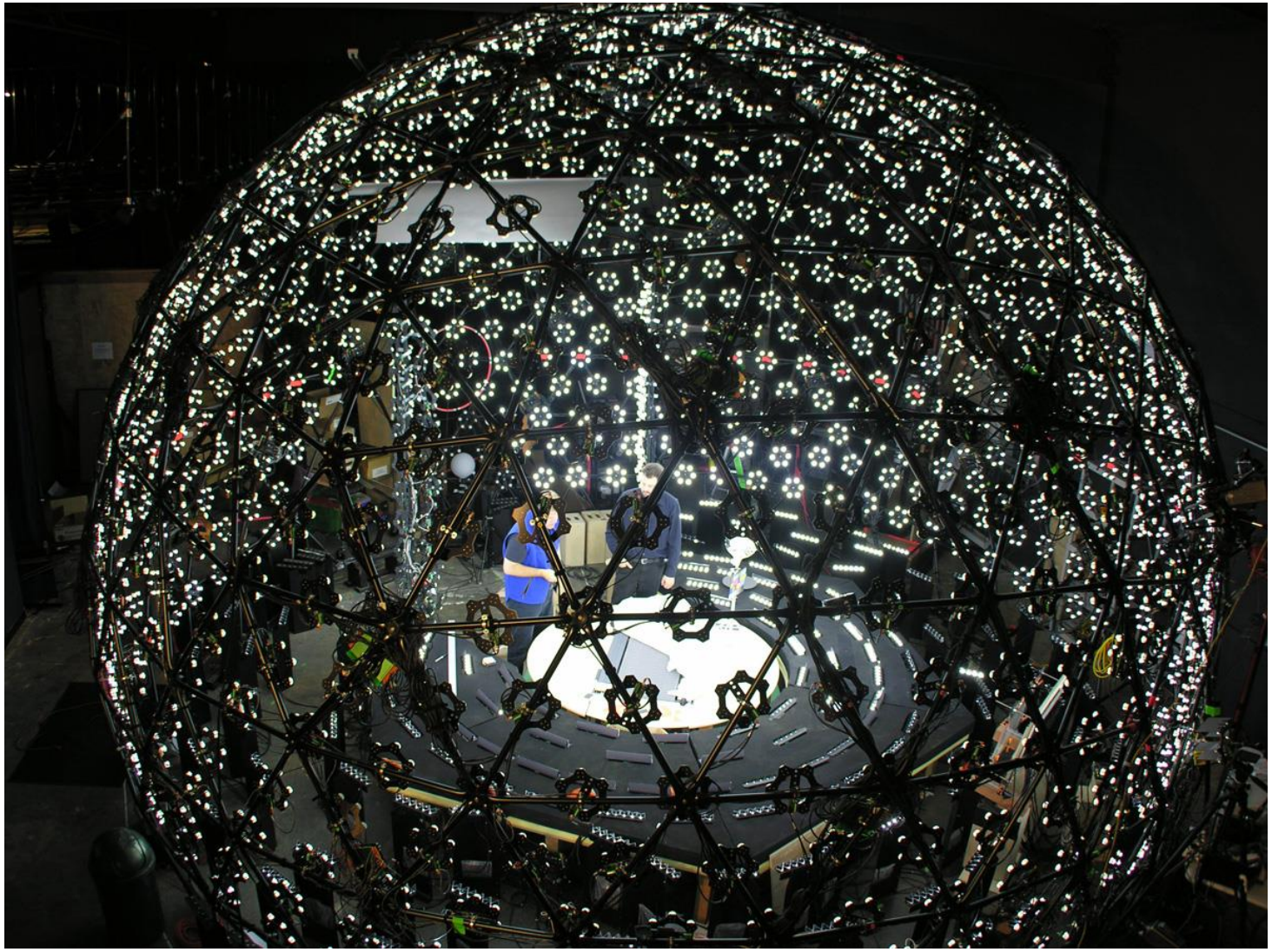
An Anisotropic Phong BRDF Model



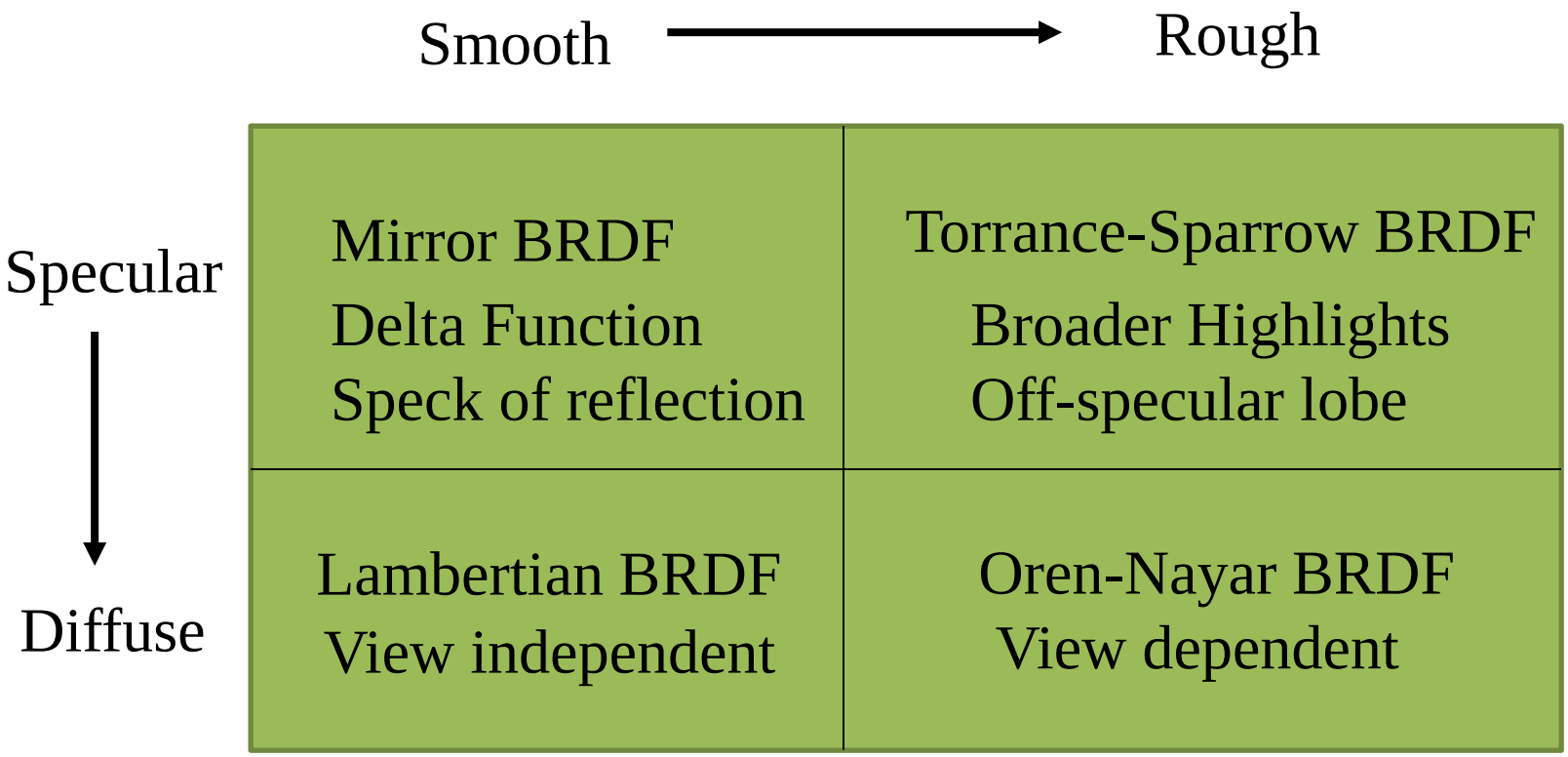

Cylinder (1D normal variation)
with stripes of the material
at different orientations (1D)
Precision
motor
Camera
Rotation of
cylinder (1D)
Light source path (1D)
Light source
Precision
motor


1-Felt
2-Polyester
3-Terrycloth
4-Rough Plastic
5-Leather
6-Sandpaper
7-Velvet
8-Pebbles
9-Frosted Glass
10-Plaster_a
11-Plaster_b
12-Rough Paper
13-Artificial Grass
14-Roof Shingle
15-Foil
16-Cork
17-Rough Tile
18-Rug_a
19-Rug_b
20-Styrofoam
21-Sponge
22-Lambswool
23-Lettuce Leaf
24-Rabbit Fur
25-Quarry Tile
26-Loofa
27-Insulation
28-Crumpled Paper
29-(2 zoomed)
30-(11 zoomed)
31-(12 zoomed)
32-(14 zoomed)
33-Slate_a
34-Slate_b
35-Spheres
36-Limestone
37-Brick_a
38-Ribbed Paper
39-Human Skin
40-Straw
41-Brick_b
42-Corduroy
43-Salt Crystals
44-Linen
45-Concrete_a
46-Cotton
47-Stones
48-Brown Bread
49-Concrete_b
50-Concrete_c
51-Corn Husk
52-White Bread
53-Plant
54-Wood_a
55-Orange Peel
56-Wood_b
57-Peacock Feather
58-Tree Bark
59-Cracker_a
60-Cracker_b
61-Moss

Light stage 6

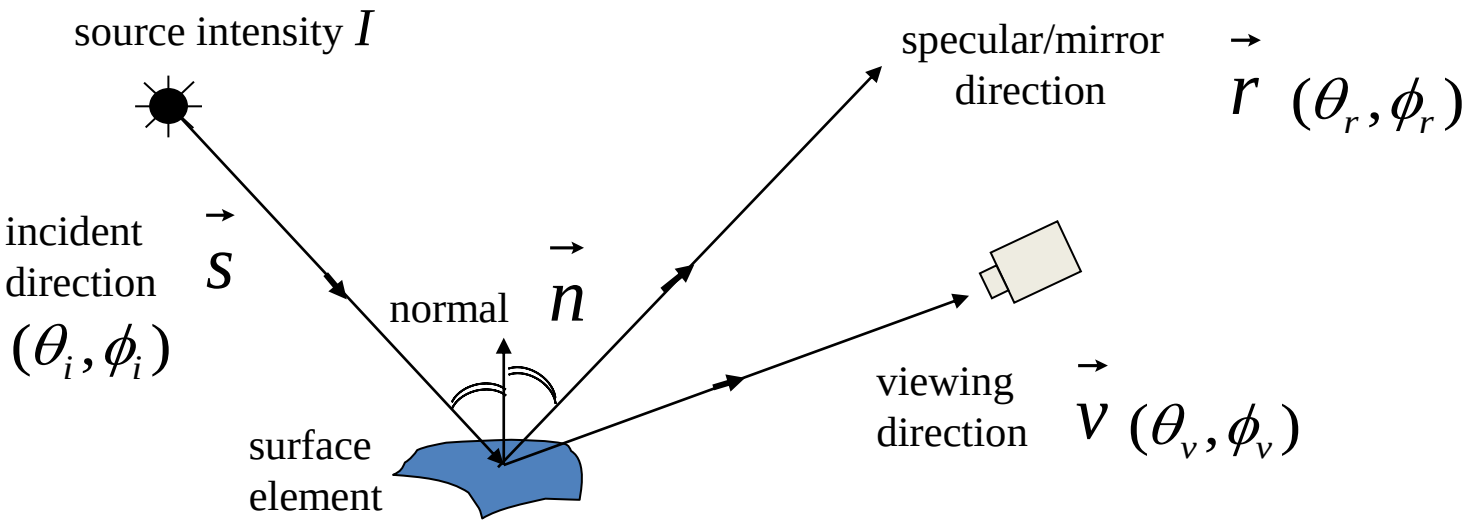


Summary of Surfaces and BRDFs



Many surfaces may be rough and show both diffuse and surface reflection.

Specular Reflection and Mirror BRDF



- **Very smooth surface.**
- All incident light energy reflected in a SINGLE direction.
- (only when $\vec{v} = \vec{r}$)
- Mirror BRDF is simply a double-delta function :

specular albedo

$$f(\theta_i, \phi_i; \theta_v, \phi_v) = \rho_s\ \delta(\theta_i - \theta_v)\ \delta(\phi_i - \phi_v)$$

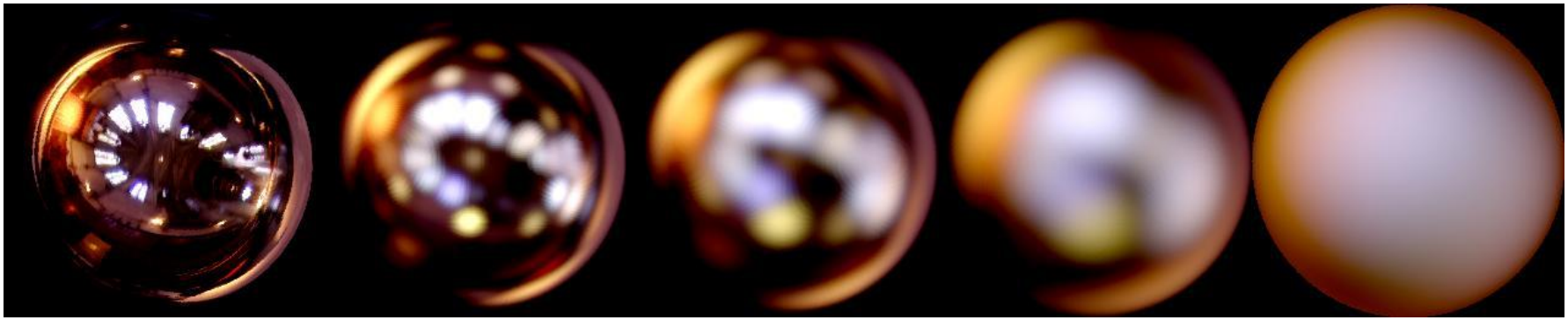
- Surface Radiance : $L = I\ \rho_s\ \delta(\theta_i - \theta_v)\ \delta(\phi_i - \phi_v)$

Glossy Surfaces

- Delta Function is a harsh BRDF model
  (valid only for highly polished mirrors and metals).
- Many glossy surfaces show broader highlights

- Surfaces are not perfectly smooth –
  - they show micro-surface geometry (roughness).
- Example Models : Phong model
  Torrance Sparrow model

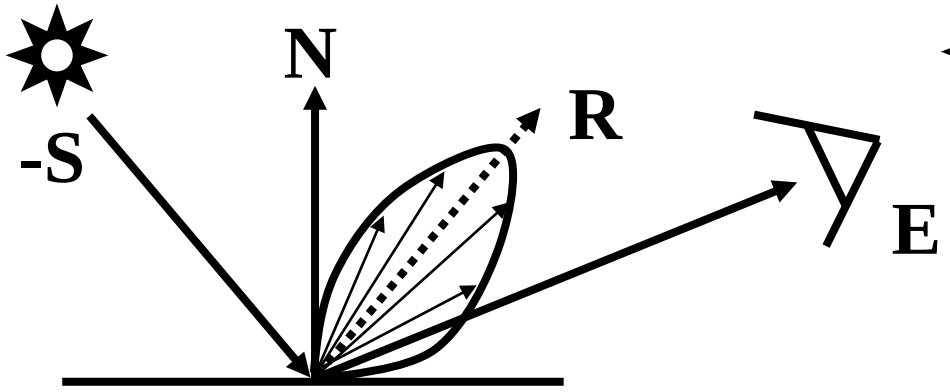
Blurred Highlights and Surface Roughness



Roughness

Phong Model: An Empirical Approximation

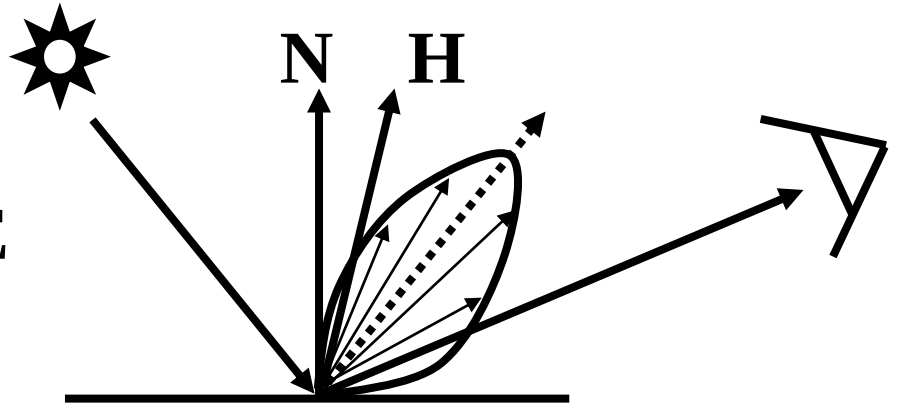
- How to model the angular falloff of highlights:



$$L = I\ \rho_s\ (R \cdot E)^{n_{shiny}}$$

$$R = -(N \cdot S)N + S$$

Phong Model



$$L = I\ \rho_s\ (N \cdot H)^{n_{shiny}}$$

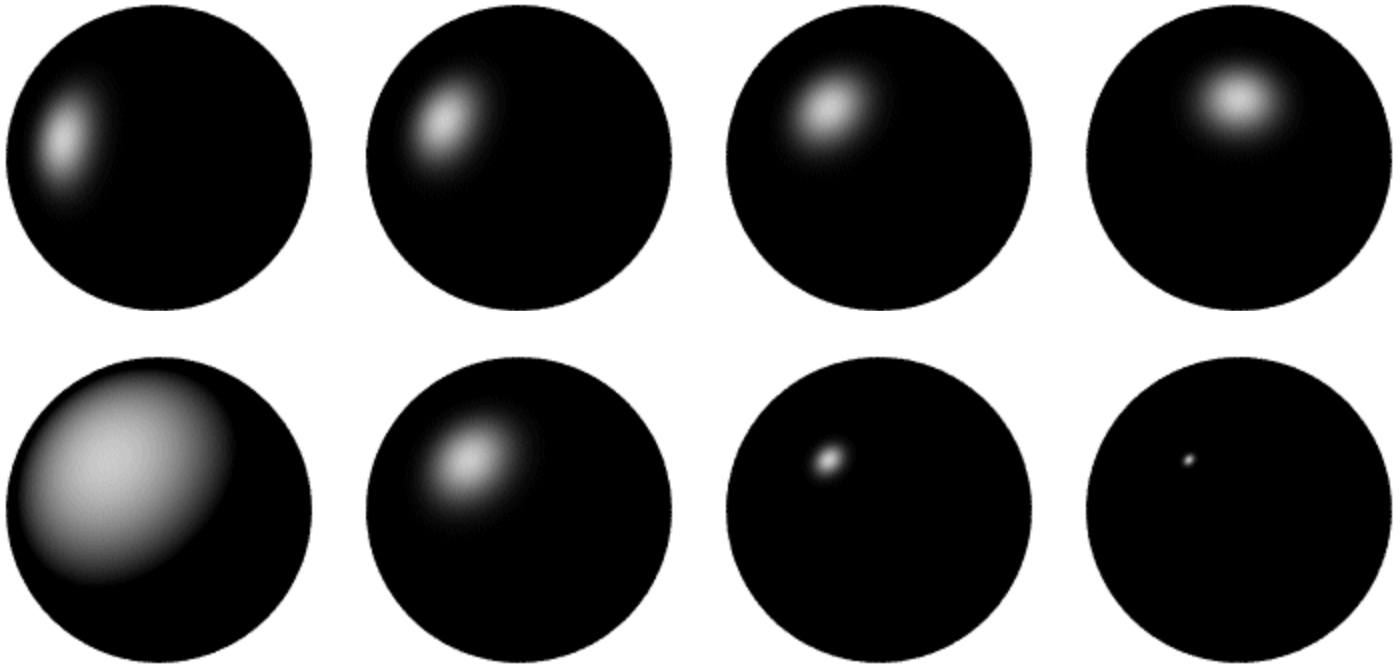
$$H = (E - S)/2$$

Blinn-Phong Model

- Sort of works, easy to compute
- But not physically based (no energy conservation and reciprocity).
- Very commonly used in computer graphics.

Phong Examples

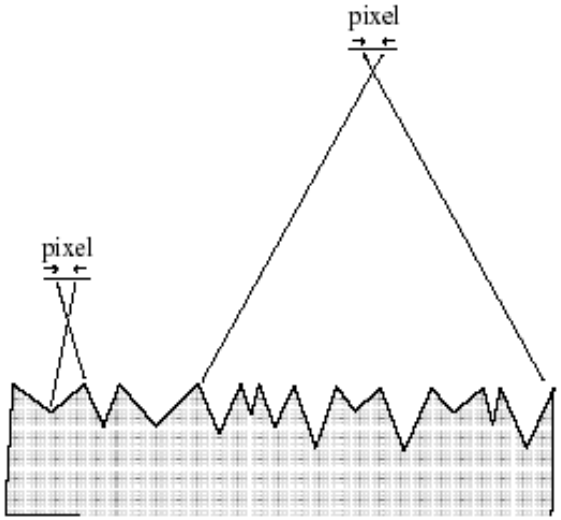
- These spheres illustrate the Phong model as *lighting direction* and n_{shiny} are varied:

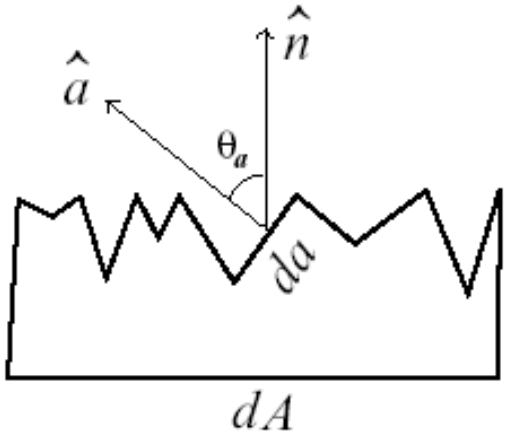


Torrance-Sparrow Model – Main Points

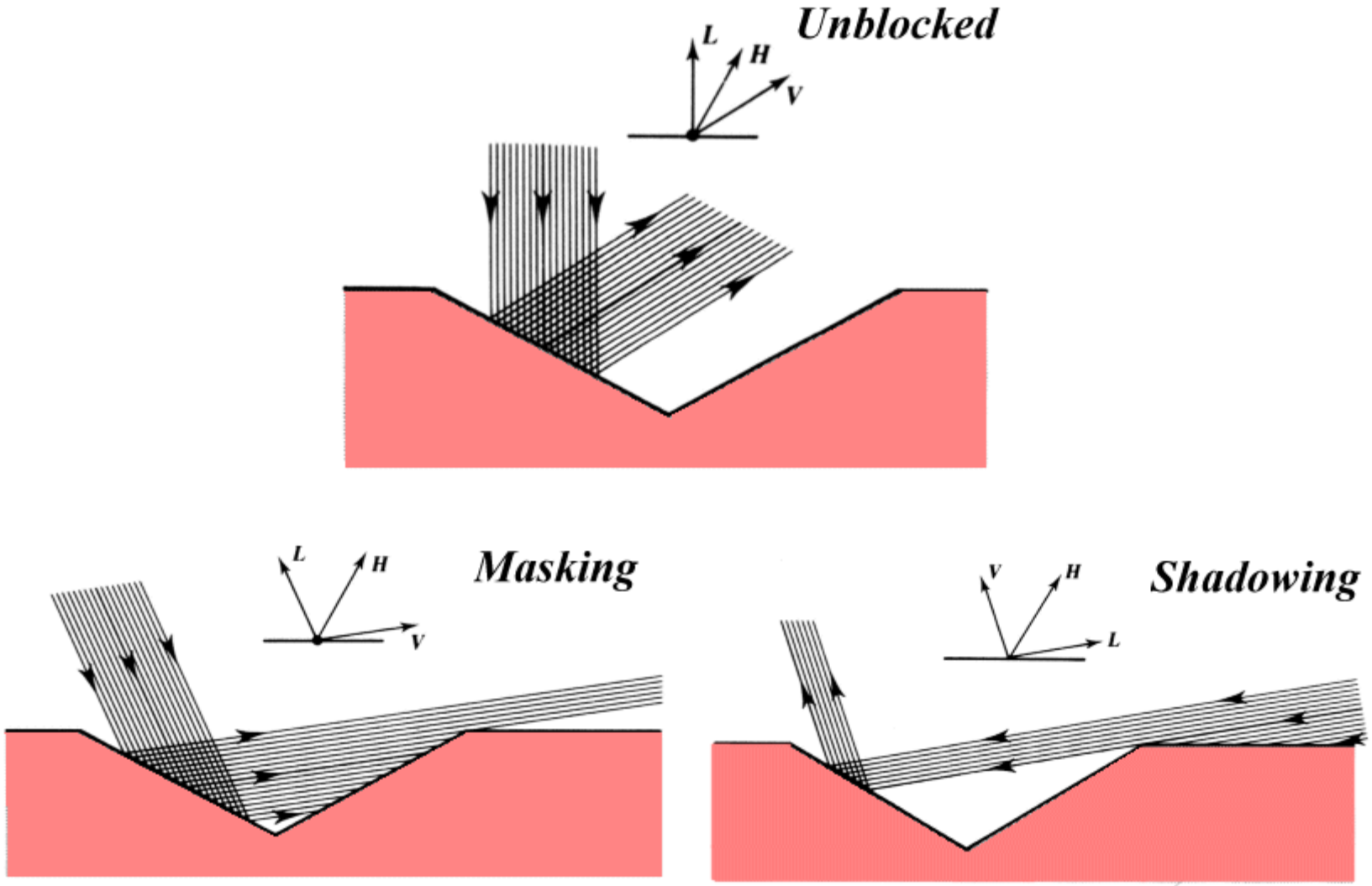
- Physically Based Model for Surface Reflection.
- Based on Geometric Optics.
- Explains off-specular lobe (wider highlights).
- Works for only rough surfaces.
- For very smooth surfaces, electromagnetic nature of light must be used

Modeling Rough Surfaces - Microfacets





- Roughness simulated by Symmetric V-groves at Microscopic level.
- Distribution on the slopes of the V-grove faces are modeled.
- Each microfacet assumed to behave like a perfect mirror.


Unblocked
L
H
V
Masking
L
H
V
Shadowing
V
H
L

Torrance-Sparrow BRDF – Different Factors

Fresnel term:
allows for wavelength dependency

Geometric Attenuation:
reduces the output based on the amount of shadowing or masking that occurs.

$$f = \frac{F(\theta_i)G(\omega_i, \omega_r)D(\theta_h)}{4\cos(\theta_i)\cos(\theta_r)}$$

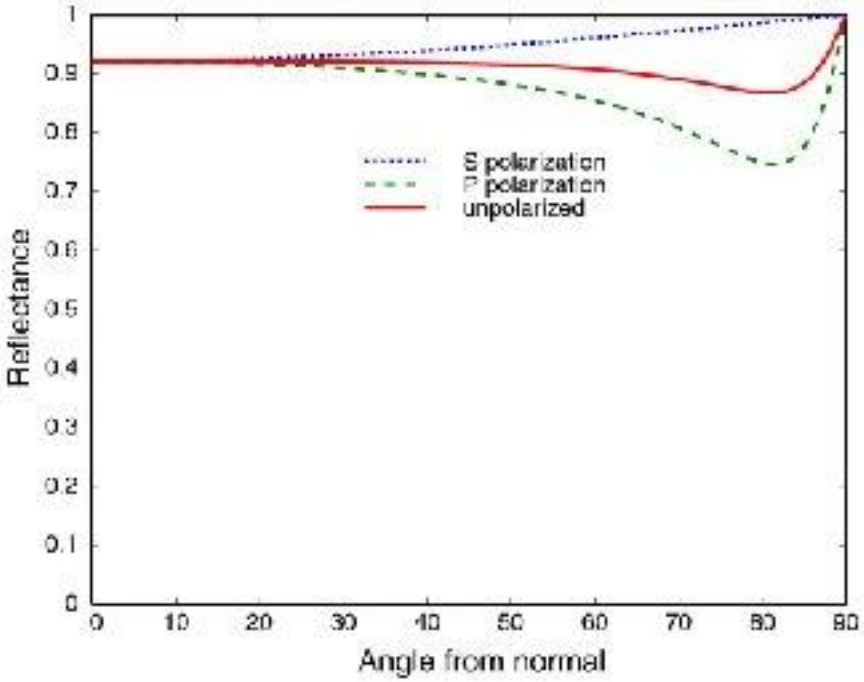
Distribution:
distribution function determines the percentage of microfacets oriented to reflect in the viewer direction.

How much of the macroscopic surface is visible to the light source

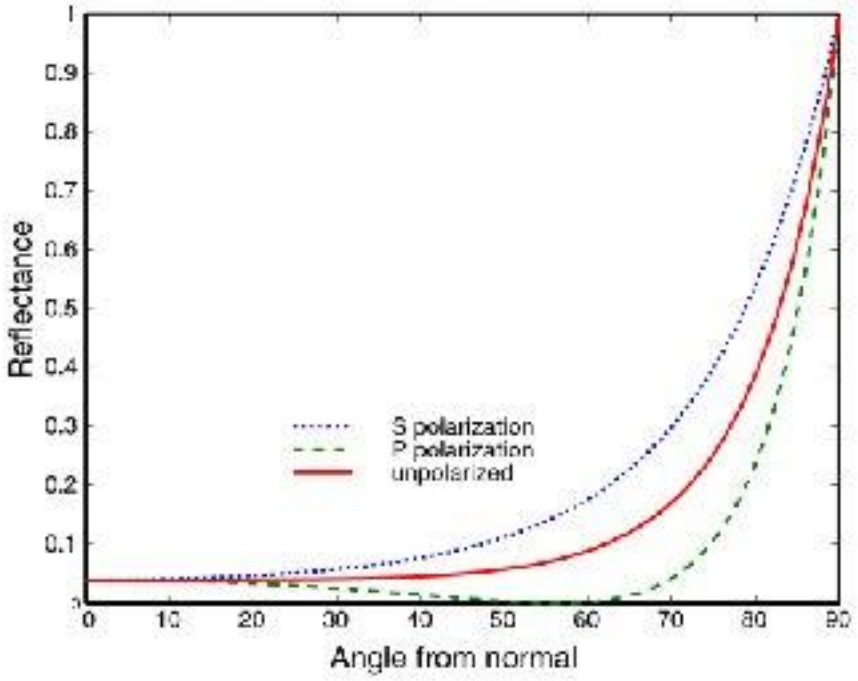
How much of the macroscopic surface is visible to the viewer

Fresnel Reflectance

Metal (Aluminum)



Dielectric (N=1.5)

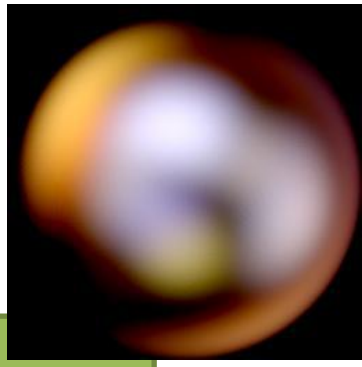


Gold F(0)=0.82
Silver F(0)=0.95

Glass n=1.5 F(0)=0.04
Diamond n=2.4 F(0)=0.15

Schlick Approximation $F(\theta) = F(0) + (1 - F(0))(1 - \cos\theta)^5$

Summary of Surfaces and BRDFs



Smooth → Rough

Specular ↓ Diffuse

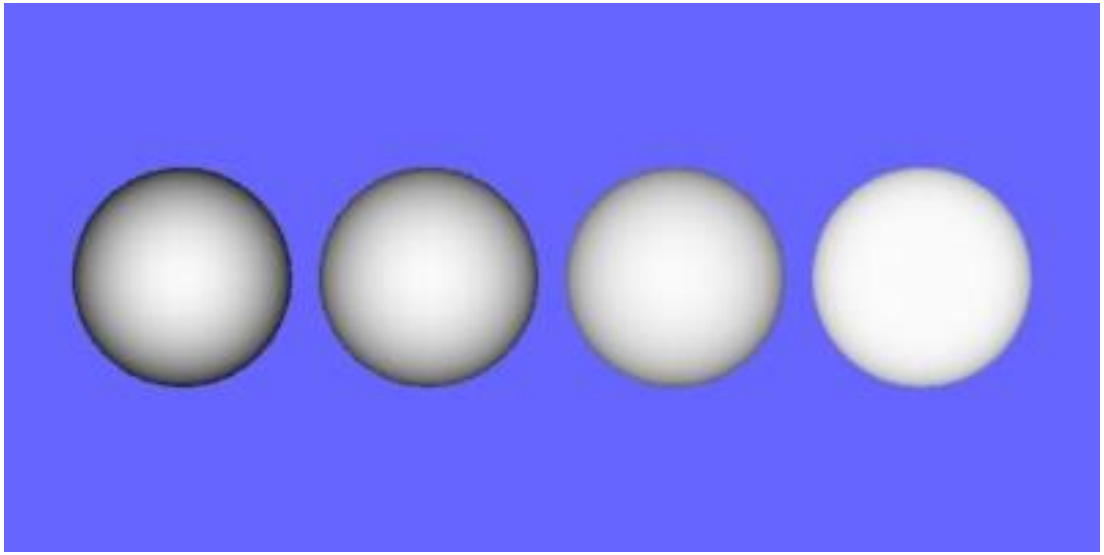
Mirror BRDF Delta Function Speck of reflection	Torrance-Sparrow BRDF Broader Highlights Off-specular lobe
Lambertian BRDF View independent	Oren-Nayar BRDF View dependent

Many surfaces may be rough and show both diffuse and surface reflection.

Rendered Sphere with Lambertian BRDF

- Edges are dark (N.S = 0) when lit head-on
- See shading effects clearly.

Surface Roughness Causes Flat Appearance



Increasing surface roughness ⟶

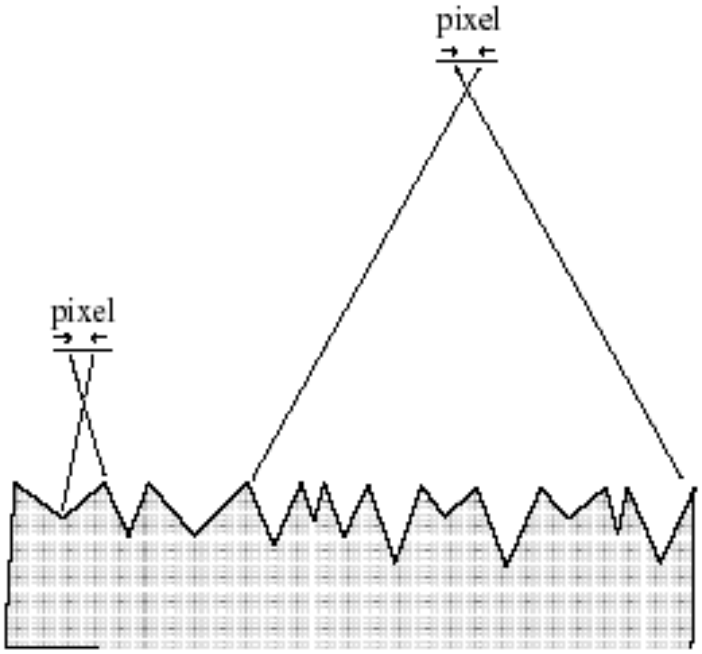
Lambertian model

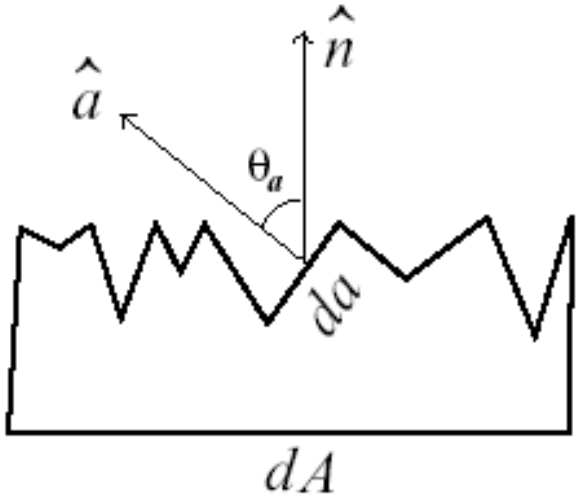
Valid for only SMOOTH MATTE surfaces.
Bad for ROUGH MATTE surfaces.

Oren-Nayar Model – Main Points

- Physical Based Model for Diffuse Reflection.
- Based on Geometric Optics.
- Explains view dependent appearance in Matte Surfaces
- Take into account partial interreflections.
- Roughness represented like in Torrance-Sparrow Model
- Lambertian model is simply an extreme case with roughness equal to zero.

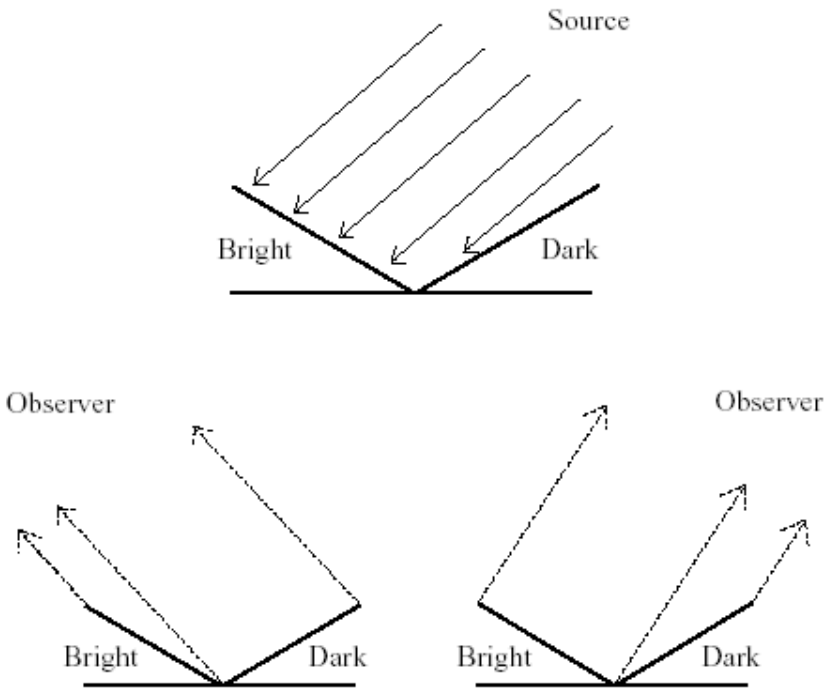
Modeling Rough Surfaces - Microfacets





- Roughness simulated by Symmetric V-groves at Microscopic level.
- Distribution on the slopes of the V-grove faces are modeled.
- Each microfacet assumed to behave like a **perfect lambertian surface.**

View Dependence of Matte Surfaces - Key Observation



• Overall brightness increases as the angle between the source and viewing direction decreases.

• Pixels have finite areas. As the viewing direction changes, different mixes between dark and bright are added up to give pixel brightness.

Torrance-Sparrow BRDF – Different Factors

Fresnel term: allows for wavelength dependency

Geometric Attenuation: reduces the output based on the amount of shadowing or masking that occurs.

$$f = \frac{F(\theta_i)G(\omega_i, \omega_r)D(\theta_h)}{4\cos(\theta_i)\cos(\theta_r)}$$

Distribution: distribution function determines the percentage of microfacets oriented to reflect in the viewer direction.

How much of the macroscopic surface is visible to the light source

How much of the macroscopic surface is visible to the viewer

Oren-Nayar Model – Different Factors

Fresnel term: allows for wavelength dependency

Geometric Attenuation: reduces the output based on the amount of shadowing or masking that occurs.

$$f = \frac{F(\theta_i)G(\omega_i, \omega_r)D(\theta_h)}{4\cos(\theta_i)\cos(\theta_r)}$$

How much of the macroscopic surface is visible to the light source

How much of the macroscopic surface is visible to the viewer

Distribution: distribution function determines the percentage of microfacets oriented to reflect in the viewer direction.

Oren-Nayar Model – Different Factors

Fresnel term: allows for wavelength dependency

Geometric Attenuation: reduces the output based on the amount of shadowing or masking that occurs.

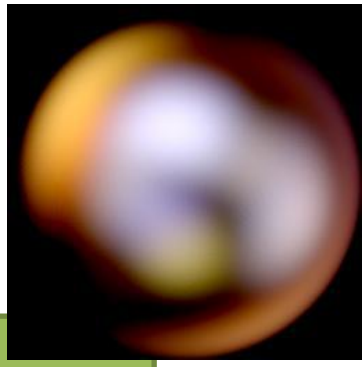
$$f = \frac{F(\theta_i)G(\omega_i, \omega_r)D(\theta_h)}{4\cos(\theta_i)\cos(\theta_r)}$$

Distribution: distribution function determines the percentage of the surface area where the facets visible to the light source

How much of the macroscopic surface is visible to the light source

How much of the macroscopic surface is visible to the viewer

Summary of Surfaces and BRDFs



Smooth → Rough

Specular ↓ Diffuse

Mirror BRDF Delta Function Speck of reflection	Torrance-Sparrow BRDF Broader Highlights Off-specular lobe
Lambertian BRDF View independent	Oren-Nayar BRDF View dependent

Many surfaces may be rough and show both diffuse and surface reflection.

Anisotropic Reflection