# Graphics & Visualization

**Chapter 12**

# *ILLUMINATION MODELS & ALGORITHMS*

# Introduction

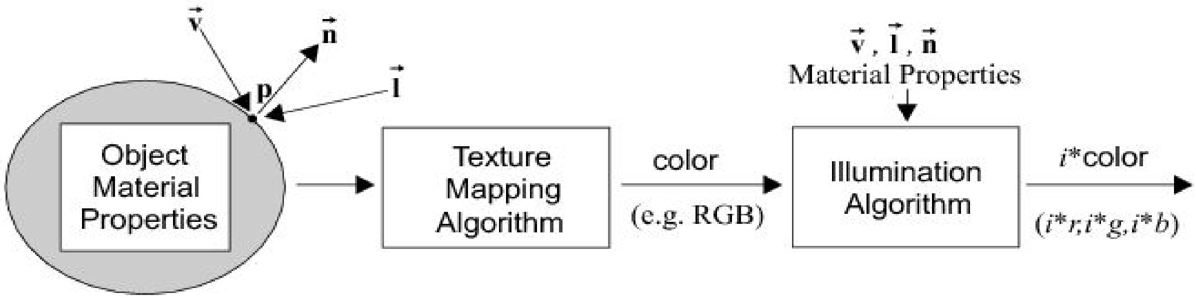
- Representation of illumination phenomena in CG :
  - Based on the laws of optics
- Laws that make the most difference in practice are implemented
- Computational cost must be considered

**Role of an *Illumination Model*:**

- Light illuminates a point **p** of an object (directly or via reflections) → changes object’s color at **p** according to:
  - Direction of the incident light
  - Direction of observation
  - Surface normal at **p**
  - Reflectivity of the material
  - …

# Introduction (2)

- Effects of illumination and texturing should not be confused:




- Two trends in Computer Graphics:
  - 1st uses practical illumination models:
    - Produces acceptable illumination effects
    - Low computational cost
    - Suitable for real-time applications
  - 2nd implements a large part of available illumination theory:
    - Produces the most convincing illumination effects
    - High computational cost
    - Suitable for demanding, non real-time applications

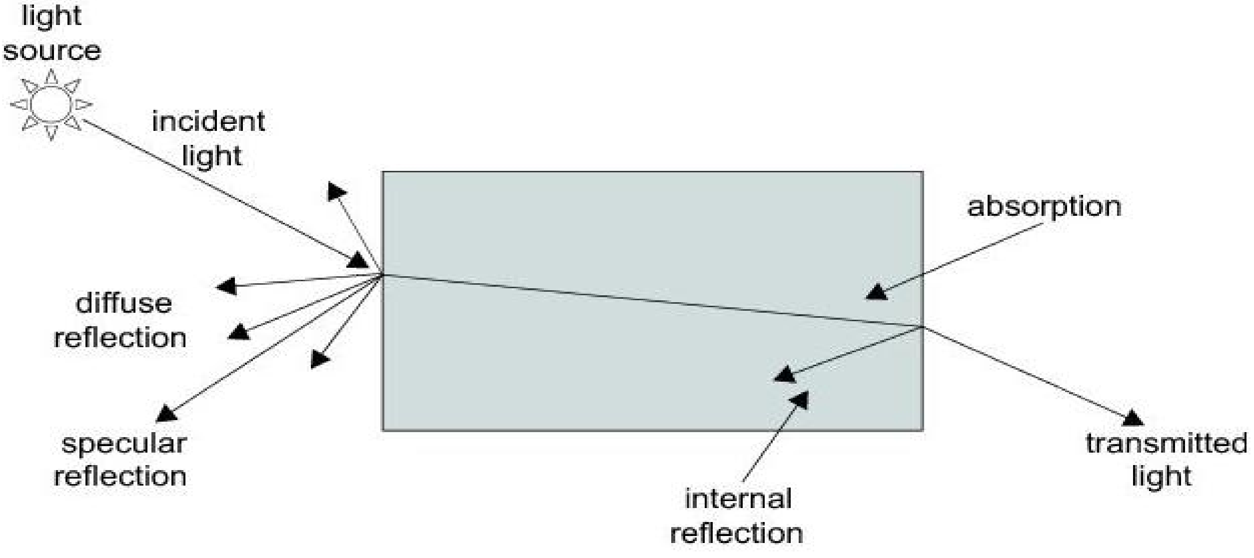
# Introduction (3)

- Essential difference between the two approaches:
  - The 2nd considers the interaction of light between objects or how objects are indirectly illuminated by light reflected from other objects
  - 1st approach constitutes the *local illumination models*
  - 2nd approach constitutes the *global illumination models*

- Distinction between illumination models and algorithms:
  - An illumination model encapsulates a set of physical illumination laws
  - An illumination algorithm implements an illumination model efficiently

# Physics of Light-Object Interaction I

- Incident Light= reflected light + scattered light + absorbed light + transmitted light
- Depending on the structure (roughness) of the object's surface portions of the incident light will be:
  - Reflected in the "mirror" of the incident direction (*specular reflection)*
  - Scattered in all directions (*diffuse reflection)*
  - Absorbed, increasing the object's temperature
  - Transmitted through the object (if transparent)



# Physics of Light-Object Interaction I (2)

- *Radiant energy (Q):* emitted from a light source or reflected from a surface and transferred through space as photons
  - Radiant energy is the total energy emitted as radiation of all wavelengths in a defined period of time and is measured in *joules*
- *Radiant power* (or *flux* $\Phi$): the rate at which radiant energy passes a spatial reference and is measured in *watts*:

$$\Phi = dQ / dt \qquad (12.1)$$

- Energy emitted or reflected from a point:
  - may be restricted to certain directions or
  - may be spreading equally in all directions
- *Radiant intensity* ($I_r$): radiant power per unit of solid angle $\omega_r$ in a certain direction:

$$I_r = d\Phi_r / d\omega_r \qquad (12.2)$$

  - *Intensity* is measured in *watts/steradian* (overloaded term)

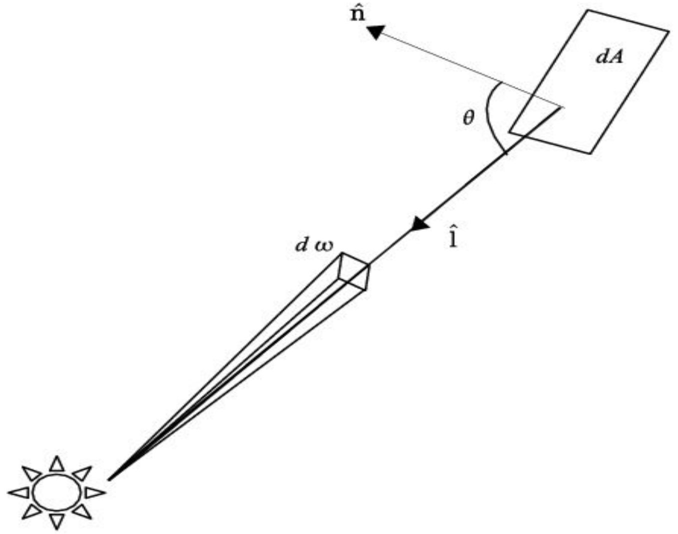
# Physics of Light-Object Interaction I (3)

- *Radiance (L):* Assume an infinitesimal surface *dA* with normal vector $\hat{\mathbf{n}}$ forming an angle $\theta$ with the direction of incident or outgoing illumination $\hat{\mathbf{l}}$. Radiance is defined as the radiant power per unit solid angle leaving or entering the infinitesimal area *dA* from a certain direction per unit projected surface area in that direction:

$$L = d\Phi / (d\omega dA \cos\theta) = d\Phi / (d\omega dA(\hat{\mathbf{n}} \cdot \hat{\mathbf{l}})) \qquad (12.3)$$

# Physics of Light-Object Interaction I (4)

- Radiance:




- Radiance is inversely proportional to the square of the distance from the light source and is measured in *watts/(steradians · m*$^2$*)*
- *Albedo (ρ):* Ratio of scattered to incident electromagnetic radiation across the spectrum
  - Albedo defines the color of a material without the effect of illumination

# Physics of Light-Object Interaction I (5)

- *Irradiance ($E_i$):* Incident flux per unit area in the vicinity of a point
  - Irradiance can be visualized as the power per unit area incident from all directions within a hemisphere onto an elementary surface located at the center of the base of that hemisphere:

$$E_i = d\Phi_i / dA \qquad (12.4)$$

  - Measured in *watts/m²*
- *Radiosity (B):* flux per unit area exiting a surface:

$$E_r = B = d\Phi_r / dA \qquad (12.5)$$

  - Also measured in *watts/m²*
- *Incident intensity ($I_i$):* incident flux on a point per unit solid angle:

$$I_i = d\Phi_i / d\omega_i \qquad (12.6)$$

- Relation between incident intensity & irradiance by combining (12.4) & (12.6): $E_i = I_i d\omega_i / dA.$ (12.7)

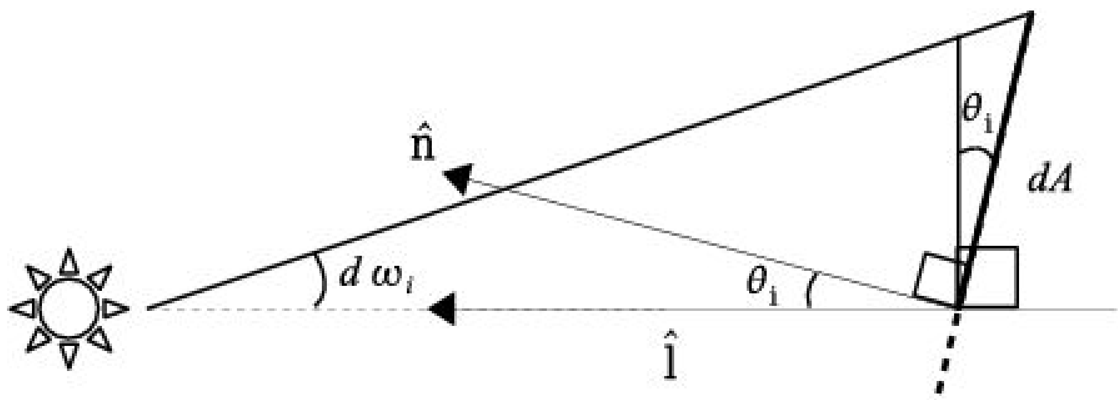
# Physics of Light-Object Interaction I (6)

- From the definition of solid angle:

$$d\omega_i = \frac{dA\cos\theta_i}{d^2}$$

where $dA \cdot \cos\theta_i$: projection of the elementary surface $dA$ onto a plane normal to the direction of illumination

$d$: distance from the light source to the elementary surface



# Physics of Light-Object Interaction I (7)

- Photometry law (from (12.7) and solid angle definition):

$$E_i = I_i \frac{\cos\theta_i}{d^2} = I_i \frac{(\hat{\mathbf{n}} \cdot \hat{\mathbf{l}})}{d^2} \qquad (12.8)$$

- In Computer Graphics we are interested in the relationship between the incident light from a certain direction onto a surface and:
  - The reflected light in another direction
  - The transmitted light through the object
- Relationship is captured by the *bidirectional reflectance distribution function (BRDF)*

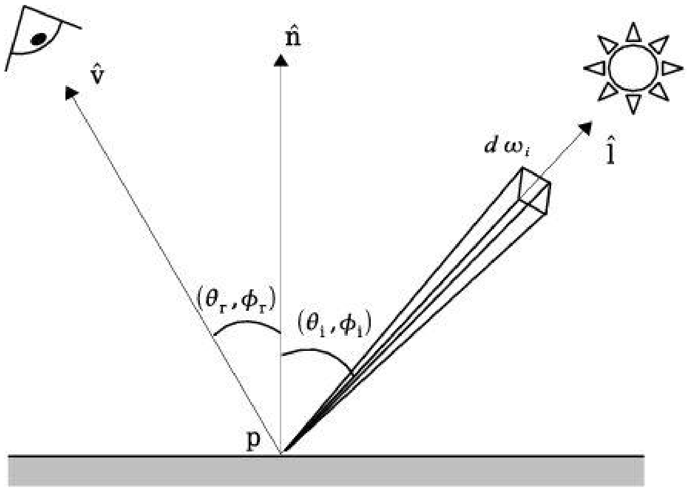
# Physics of Light-Object Interaction I (8)

- BRDF depends on:
  - Lighting and observation directions
  - Wavelength
  - Shadow casting
  - The optical properties of the object
  - Reflectivity
  - Absorption
  - Emission
- BRDF can only be approximated
- BRDF associates the outgoing radiance $dL_r$ in direction $(\theta_r, \varphi_r)$ to the irradiance $dE_i$ from the incident direction $(\theta_i, \varphi_i)$:

$$BRDF = \frac{dL_r}{dE_i} \qquad (12.9)$$

# Physics of Light-Object Interaction I (9)

- BRDF




- BRDF expresses how objects look differently when seen from different angles or when illuminated from different directions, e.g.:

Light source opposite observer

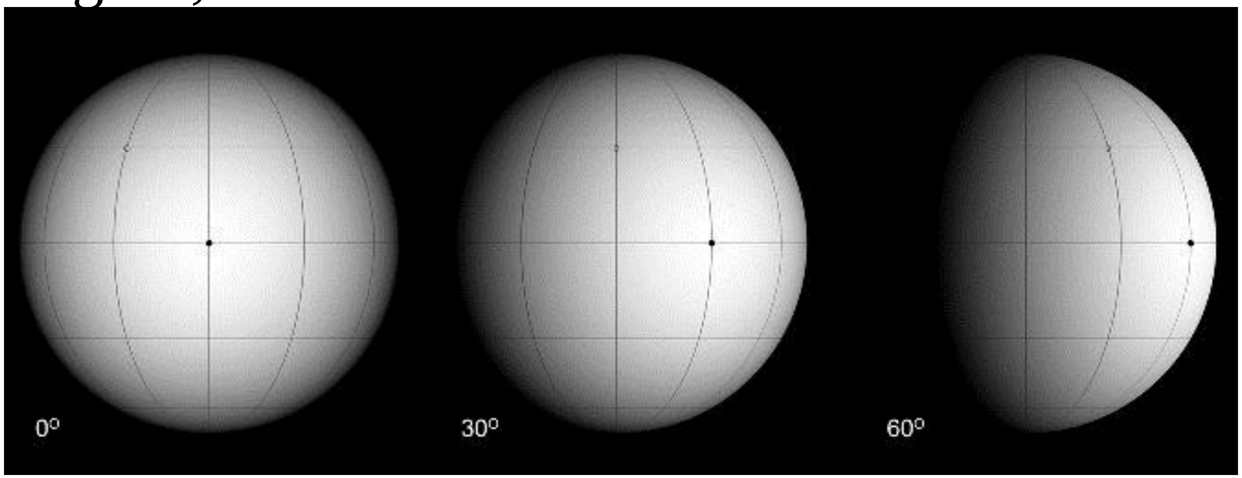
Light source behind observer

# The Lambert Illumination Model*

- Simplest illumination model for body reflection
- Assumes that incident light at vicinity of a point **p** on a surface is equally diffused in all directions on the incident hemisphere
- BRDF of the body surface is constant in all directions and invariant with respect to wavelength and polarization
- A perfectly diffuse surface is called *Lambertian*
- *Diffuse illumination* mostly accounts for reflected light due to body reflectance
  - (In contrast, *specular illumination* corresponds to light reflected off the surface)
- *Lambert's cosine law:* The total radiant power observed from a Lambertian surface is directly proportional to the cosine of the angle $\theta_r$ between the observer's line of sight and the surface normal.

# The Lambert Illumination Model (2)

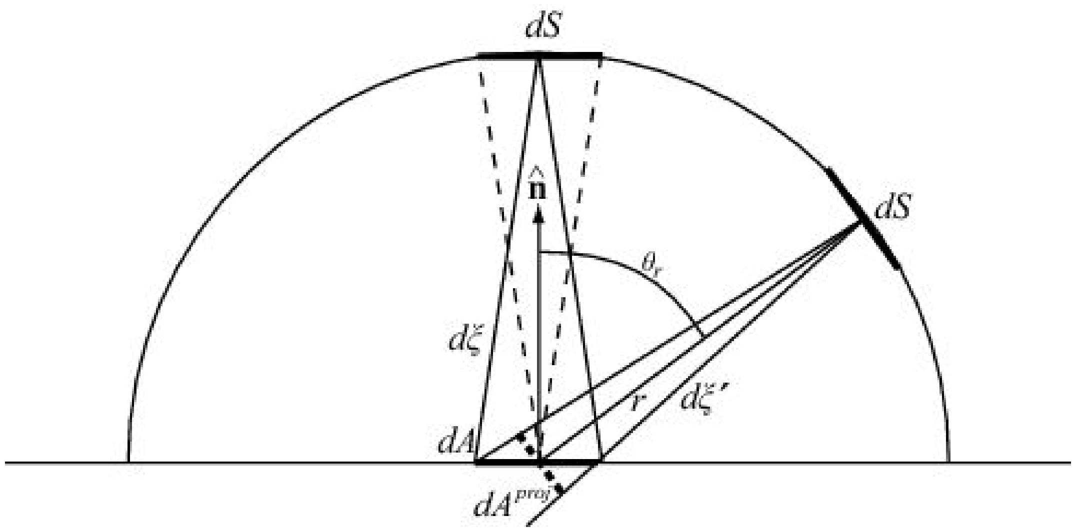
- Consequence of Lambert's law: when an elementary surface $dA$ is viewed from an arbitrary direction within the hemisphere $\Omega$ surrounding $dA$, it exhibits the same radiance




- Explanation: As the radiant power $d\Phi_r$ observed at a direction $(\theta_r,\phi_r)$ diminishes according to Lambert's cosine law, so does the solid angle $d\xi$ subtended by the surface patch $dA$ and viewed from a distant patch $dS$ around the observer location → equal decrease of both terms, which eventually cancel out

# The Lambert Illumination Model (3)

- Imagine that receiving patch *dS* were positioned directly above *dA*, perpendicular to the normal vector of *dA*




- Since $\theta_r=0$, from the definition of radiance (12.3) → the observed radiance is:

$$L_0 = \frac{d\Phi_0}{dSd\xi}$$

# The Lambert Illumination Model (4)

- Let us position *dS* at a different viewing angle, away from the normal direction of *dA*, always perpendicular to the corresponding viewing direction vector (see previous figure)
- According to Lambert's cosine law, the new radiance at this arbitrary outbound direction is:

$$L = \frac{d(\Phi_0 \cos\theta_r)}{dS d\xi'} = \frac{\cos\theta_r d\Phi_0}{dS d\xi'} \qquad (12.11)$$

- As *dA* is very small, the new solid angle $d\xi'$ is proportional to the projection of *dA* on the light transfer direction $(d\xi' = dA\cos\theta_r/r^2)$, and therefore:

$$d\xi' = \cos\theta_r d\xi \qquad (12.12)$$

# The Lambert Illumination Model (5)

- Replacing the new solid angle in (12.11) yields:

$$L = \frac{\cos\theta_r d\Phi_0}{dSd\xi'} = \frac{\cos\theta_r d\Phi_0}{\cos\theta_r dSd\xi} = \frac{d\Phi_0}{dSd\xi} = L_0 \qquad (12.13)$$

- We next derive constant BRDF ($f_d$) for the Lambertian surface
- Radiant flux is evenly distributed over the hemisphere subtended by the surface patch at vicinity of **p**, BUT $f_d$ is not equal to $1/2\pi$
- Outgoing radiance is constant → does not depend on the reflected light direction on the hemisphere: $L_r(\theta_r,\phi_r) = L_r$
- Irradiance is not attenuated by the material & is equally spread to every outgoing differential solid angle → reflectance factor $\rho(\vec{\omega}_i \rightarrow \Omega)$ (ratio of total reflected light to incident light from $d\vec{\omega}_i$) equals 1

# The Lambert Illumination Model (6)

- From definition of irradiance, radiosity, and radiance ((12.4), (12.5), (12.3)) we get:

$$d\Phi_i = E_i dA \qquad (12.14)$$

$$L_r(\theta_r, \varphi_r) = \frac{dE_r(\theta_r, \varphi_r)}{d\vec{\omega}_r \cos\theta_r} = \frac{dE_r(\theta_r, \varphi_r)}{d\vec{\omega}_r^{proj}} \Rightarrow$$

$$dE_r = L_r d\vec{\omega}_r^{proj} \Rightarrow E_r = \int_\Omega L_r d\vec{\omega}_r^{proj} \qquad (12.15)$$

- Using the results from (12.14) & (12.15), the unit reflectance becomes:

$$\rho(\vec{\omega}_i \to \Omega) = 1 = \frac{d\Phi_r}{d\Phi_i} = \frac{dA\int_\Omega L_r d\vec{\omega}_r^{proj}}{E_i dA} = \frac{L_r dA\int_\Omega d\vec{\omega}_r^{proj}}{E_i dA} \qquad (12.16)$$

# The Lambert Illumination Model (7)

- From definition of BRDF & taking into account that BRDF for the Lambertian surface is constant, we have:

$$f_d = \frac{dL_r}{L_i \cos\theta_i d\vec{\omega}_i} \Rightarrow dL_r = f_d L_i d\vec{\omega}_i^{proj} \Rightarrow$$

$$L_r = \int_\Omega f_d L_i d\vec{\omega}_i^{proj} = f_d \int_\Omega L_i d\vec{\omega}_i^{proj} = f_d E_i \qquad (12.17)$$

- Now we can return to (12.16) and substitute $L_r$ from (12.17):

$$1 = \frac{L_r dA \int_\Omega d\vec{\omega}_r^{proj}}{E_i dA} = \frac{f_d E_i dA \int_\Omega d\vec{\omega}_r^{proj}}{E_i dA} =$$

$$= f_d \int_\Omega d\vec{\omega}_r^{proj} = f_d \pi \Leftrightarrow f_d = \frac{1}{\pi} \qquad (12.18)$$

# The Lambert Illumination Model (8)

- Summing up:
  - The radiance associated with an infinitesimal surface patch of area $dA$ around point $\boldsymbol{p}$ is proportional to the cosine of the angle $\theta_i$ between normal vector at $\boldsymbol{p}$ and the incident direction
  - The above happens due to the flow of energy that passes through the (projected) area $dA$ of the patch with respect to the incident light direction

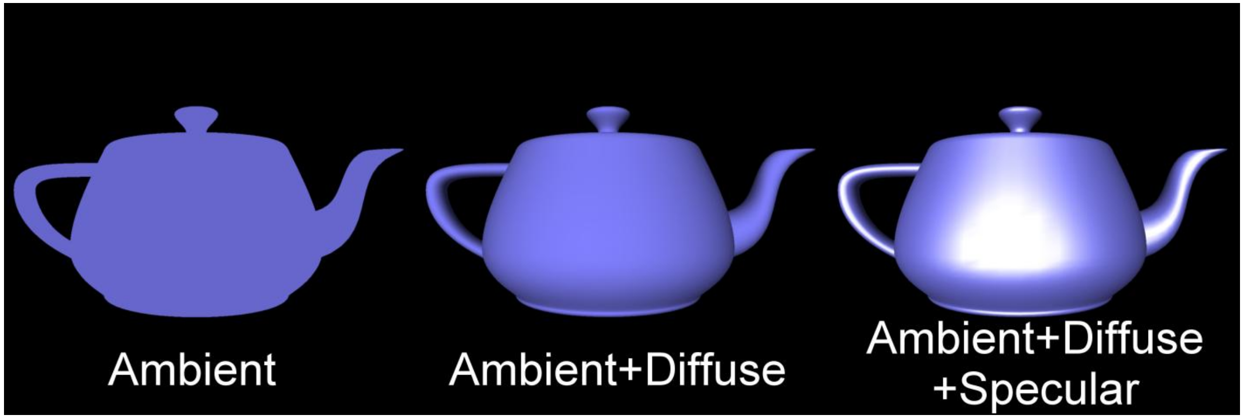
# The Phong Illumination Model

- The classic local empirical model
  - Does not take into account the interaction of light between objects
- Some of the terms used do not directly derive from physical laws
- Gives a reasonable approximation of reality
- Modest computational cost → widespread adoption
- Proposes a simplified BRDF which:
  - Relates incoming light intensity from direction $(\theta_i, \varphi_i)$ to reflected light intensity in direction $(\theta_r, \varphi_r)$ for an object point **p**
  - Estimates visible intensity as sum of 4 components:
    - Emission
    - Ambient reflection
    - Diffuse reflection
    - Specular reflection

$$\left.\begin{array}{l}\text{Emission}\\\text{Ambient reflection}\\\text{Diffuse reflection}\\\text{Specular reflection}\end{array}\right\} \quad I = I_e + I_g + I_d + I_s \qquad (12.19)$$

# The Phong Illumination Model (2)

- Components of the Phong Model:




- **Emission** component $I_e$: caters for objects with self illumination
- **Ambient** component $I_g$: compensates for the fact that the Phong is a **local** model (thus a surface not directly illuminated by a light source will not appear completely un-illuminated)
  - ambient light $I_a$: a constant value is assumed
  - each object reflects ambient light according to its ambient reflectance coefficient $k_a$: $I_g = I_a k_a \ (0 \leq k_a \leq 1)$ (12.20)

# The Phong Illumination Model (3)

- Light that hits an object directly from a light source is split into 2 reflected components:
  - Diffusely reflected light, which is uniformly scattered in all directions
  - Specularly reflected light, which has max value in the “mirror” of the lighting direction
- **Diffuse & Specular** reflection coefficients $k_d$ and $k_s$ depend mainly on the object’s surface properties
  - The rougher the surface the more light is diffusely reflected
  - The shinier the surface the more light is specularly reflected
  - We have:

$$0 \leq kd, ks \leq 1, kd + ks \leq 1$$

  - Sum of $k_d$ and $k_s$ may be slightly smaller than 1 to account for light that is transmitted or absorbed by the object
  - Diffuse component assumes a Lambertian surface & distributes incident light evenly in all directions

# The Phong Illumination Model (4)

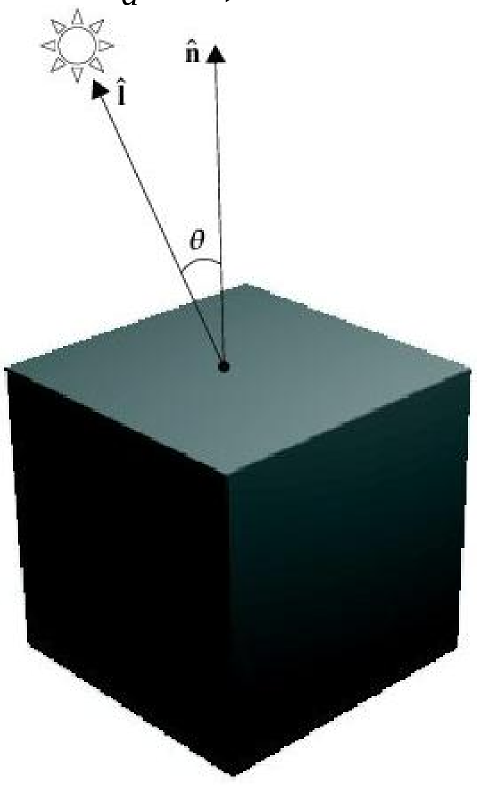
- Diffuse component does not depend on the viewing direction
  - Its value is proportional to the irradiance $E_i$ which is replaced by intensity $I_i$ according to the photometry law (12.8)
- The distance $d$ is ignored by assuming the light source at infinity:

$$I_d = I_i k_d \cos\theta = I_i k_d (\hat{\mathbf{n}} \cdot \hat{\mathbf{l}}), \quad (0 \le \theta \le \pi/2, \ \ 0 \le k_d \le 1)$$

where $I_i$ the intensity of the point light source,

$\theta$ the angle between $\hat{\mathbf{l}}$ and $\hat{\mathbf{n}}$

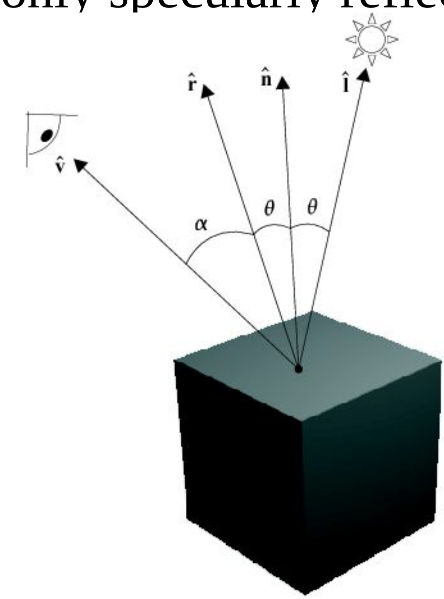
  - ( $k_d$ also depends on the wavelength of the incident light, not modeled here )



# The Phong Illumination Model (5)

- Vectors $\hat{\mathbf{l}}, \hat{\mathbf{n}}$ should be unit vectors
- $I_d$ is constant over a planar surface since the $\hat{\mathbf{l}}, \hat{\mathbf{n}}$ vectors are constant
- $cos\theta$ should be non-negative: $I_d = I_i k_d \max(0, \hat{\mathbf{n}} \cdot \hat{\mathbf{l}})$
- Diffuse component alone gives objects a totally matte appearance

- Specular component follows the rule of the mirror
  - A perfect mirror will only specularly reflect in the direction of reflection $\hat{\mathbf{r}}$



# The Phong Illumination Model (6)

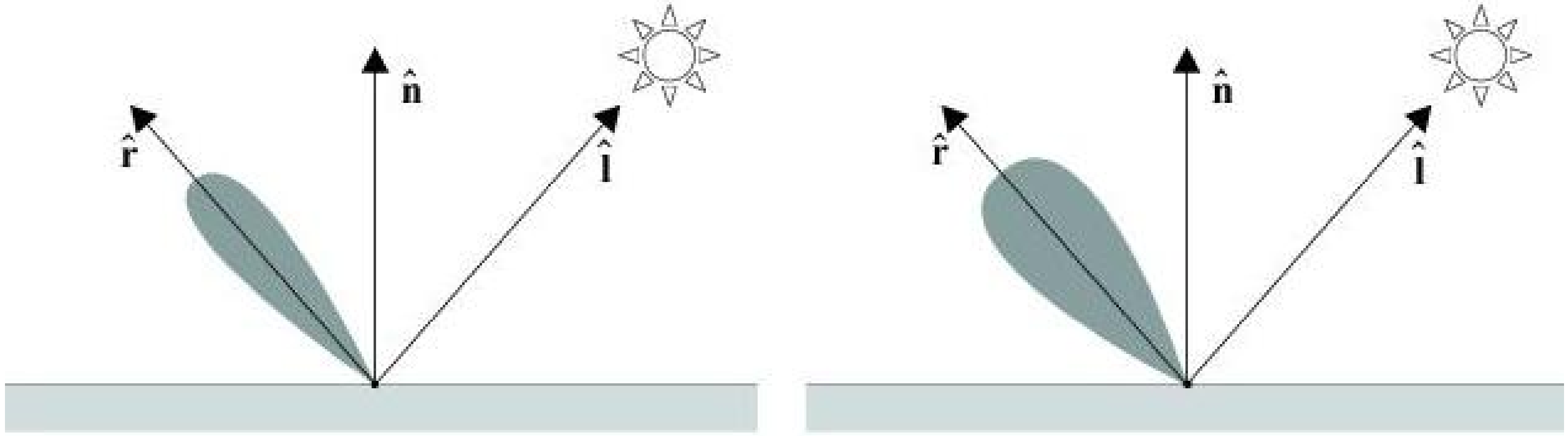
- Most surfaces have a diminishing function of specular reflection that attains its max value when the viewing direction $\hat{\mathbf{v}}$ coincides with $\hat{\mathbf{r}}$:

$$I_s = I_i k_s \cos^n \alpha = I_i k_s (\hat{\mathbf{r}} \cdot \hat{\mathbf{v}})^n \qquad (12.22)$$

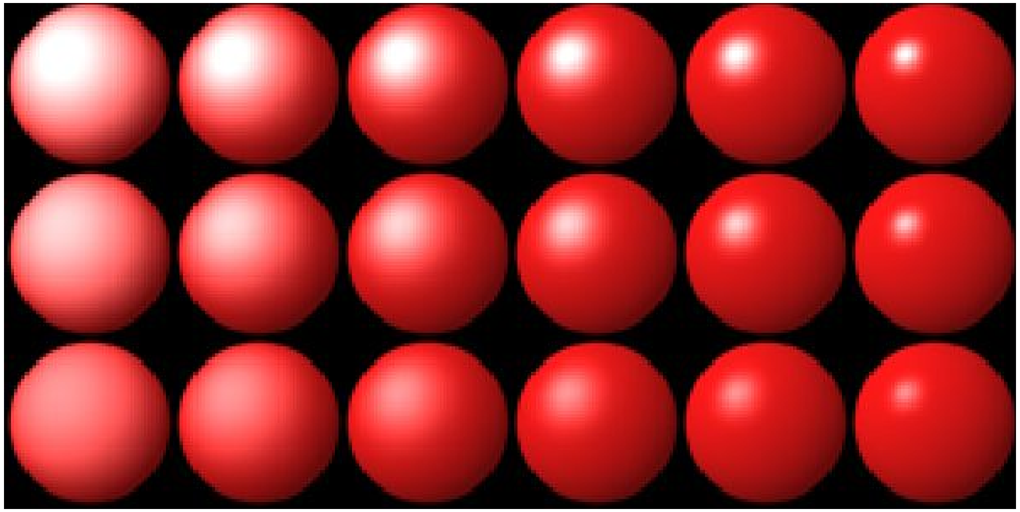
  where $\hat{\mathbf{r}}, \hat{\mathbf{v}}$ are unit vectors & $n$ an empirical value that corresponds to surface shininess
- Specular reflection is responsible for the *highlights* that are visible on shiny objects
- The $\cos^n\alpha$ term intuitively approximates the spatial distribution of the specularly reflected light

# The Phong Illumination Model (7)

- Effect of the material exponent *n* :




- *n* increases to the right, $k_s$ increases upwards:



# The Phong Illumination Model (8)

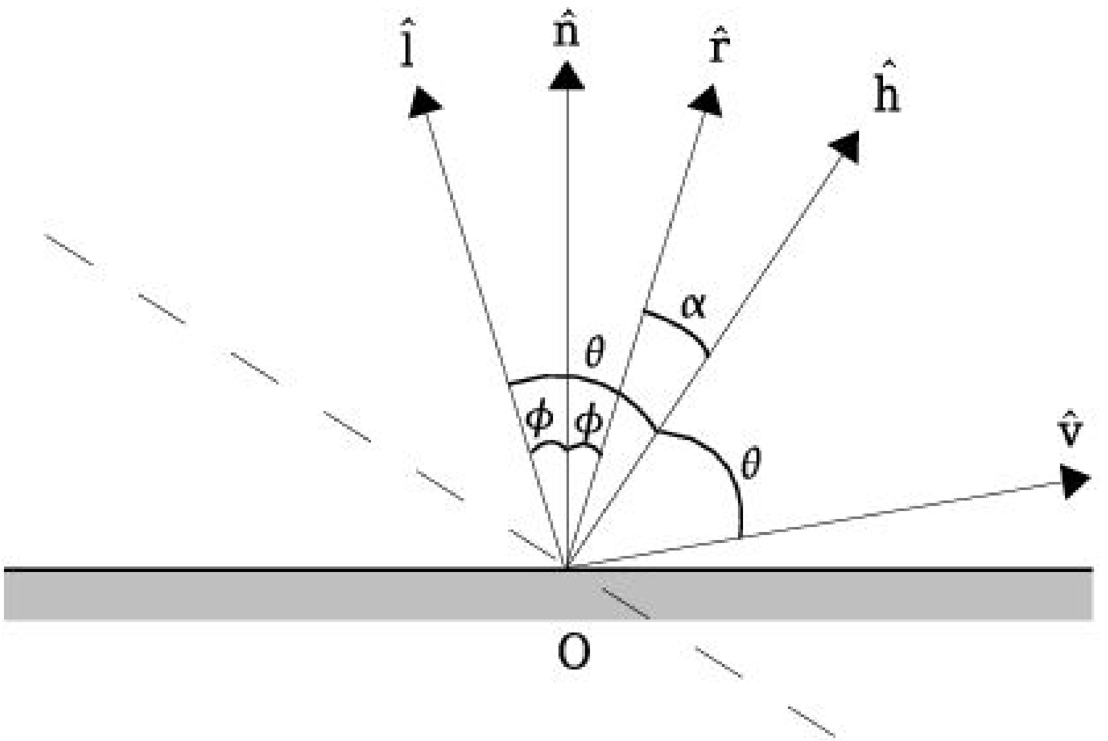
- Small values of *n* correspond to coarse materials where the size of the highlight is relatively large and scattered
- Large values of *n* correspond to shiny objects with a small and crisp highlight
- Specular reflection takes the color of the light source
- Example: If a blue object is illuminated by a white light source, the color of the diffuse reflection will be blue but that of the specular reflection will be white
- The value of the specular factor $\cos^n\alpha$ should not take on negative values, so we can replace it by $max(0, cos^n\alpha)$

- The **Phong model** computes the illumination value as:

$$I = I_e + I_a k_a + I_i (k_d (\hat{\mathbf{n}} \cdot \hat{\mathbf{l}}) + k_s (\hat{\mathbf{r}} \cdot \hat{\mathbf{v}})^n) \qquad (12.23)$$

# The Phong Illumination Model (9)

- To simplify computations: Light source & observation point can be assumed to be at infinity → constant values for $\hat{\mathbf{l}}, \hat{\mathbf{v}}$ vectors over the area of planar objects
- Efficient variant of the specular reflection calculation uses the *halfway vector* $\hat{\mathbf{h}}$ which is the average of $\hat{\mathbf{l}}, \hat{\mathbf{v}}$:




$$\hat{\mathbf{h}} = \frac{(\hat{\mathbf{l}} + \hat{\mathbf{v}})/2}{|(\hat{\mathbf{l}} + \hat{\mathbf{v}})/2|} \qquad (12.24)$$

# The Phong Illumination Model (10)

- Angle $\mathbf{nh} = \varphi + \alpha$, angle $\mathbf{rv} = \theta + \alpha$, and since $\theta = 2\phi + \alpha$, we deduce that

$$\mathbf{rv} = 2\mathbf{nh}$$

- We can thus replace $\hat{\mathbf{r}} \cdot \hat{\mathbf{v}}$ by $\hat{\mathbf{n}} \cdot \hat{\mathbf{h}}$, and adjust $n$:

$$I = I_e + I_a k_a + I_i (k_d (\hat{\mathbf{n}} \cdot \hat{\mathbf{l}}) + k_s (\hat{\mathbf{n}} \cdot \hat{\mathbf{h}})^n) \qquad (12.25)$$

- The $\hat{\mathbf{h}}$ vector is much cheaper to compute than $\hat{\mathbf{r}}$
- If $\hat{\mathbf{l}}, \hat{\mathbf{v}}$ are constant then $\hat{\mathbf{h}}$ is also constant
- $\hat{\mathbf{h}}$ can be thought of as the normal vector to the plane for which the observer at $\hat{\mathbf{v}}$ would see the max value of the specular reflection from the light source at $\hat{\mathbf{l}}$
- Since we assumed the light source at infinity, the contribution of the specular and diffuse terms depends on the intensity of the light source and the ambient term is constant

# The Phong Illumination Model (11)

- Objects with same properties & orientation but different distances from light will, wrongly, have the same intensity of illumination
- Can be corrected by including a factor dependent on the distance of the object point from the light source
  - Distance factor (physically correct square term often ignored for efficiency):

$$f(d) = 1/(c_1 + c_2 d + c_3 d^2)$$

- The model thus becomes

$$I = I_e + I_a k_a + f(d) I_i (k_d (\hat{\mathbf{n}} \cdot \hat{\mathbf{l}}) + k_s (\hat{\mathbf{n}} \cdot \hat{\mathbf{h}})^n)$$

- Multiple point light sources can also be handled:

$$I = I_e + I_a k_a + \sum_j (f(d) I_{i,j} (k_d (\hat{\mathbf{n}} \cdot \hat{\mathbf{l}}_j) + k_s (\hat{\mathbf{n}} \cdot \hat{\mathbf{h}}_j)^n))$$

- For monochromatic light, the original gray level value $v$ of an object point $\boldsymbol{p}$ is thus modified by the result $I$ of the intensity computation:

$$v' = v\, I$$

# The Phong Illumination Model (12)

- Color can be handled by giving the color of the light source to the specular reflection
- The color of the ambient & diffuse components depends on the color coefficients of the object material
- 3 intensity values, one for each of the 3 primary colors, are then computed:

$$
\begin{aligned}
I_r &= I_{er} + I_a k_{ar} + (f(d) I_i (k_{dr}(\hat{\mathbf{n}}\cdot\hat{\mathbf{l}}) + k_s(\hat{\mathbf{n}}\cdot\hat{\mathbf{h}})^n)) \\
I_g &= I_{eg} + I_a k_{ag} + (f(d) I_i (k_{dg}(\hat{\mathbf{n}}\cdot\hat{\mathbf{l}}) + k_s(\hat{\mathbf{n}}\cdot\hat{\mathbf{h}})^n)) \qquad (12.28) \\
I_b &= I_{eb} + I_a k_{ab} + (f(d) I_i (k_{db}(\hat{\mathbf{n}}\cdot\hat{\mathbf{l}}) + k_s(\hat{\mathbf{n}}\cdot\hat{\mathbf{h}})^n))
\end{aligned}
$$

- Specular reflection contributes equally to the 3 equations, simulating a white light source
  - If $(r,g,b)$ is the original color of an object at point $\boldsymbol{p}$, this is modified by the result of the color intensity computation as: $(r', g', b')=(r\, I_r, g\, I_g, b\, I_b)$

# Numerical Example

- Let us assume that we want to estimate the intensity value for a point ***p*** which, for ease of calculations, lies at the origin of the coordinate system ***p***$=[0,0,0]^T$. Let the normal to the object at ***p***, the light and the viewing vectors respectively be:

$$\vec{n}=[0,2,0]^T,\ \vec{l}=[1,1,0]^T,\ \vec{v}=[0,1,1]^T$$

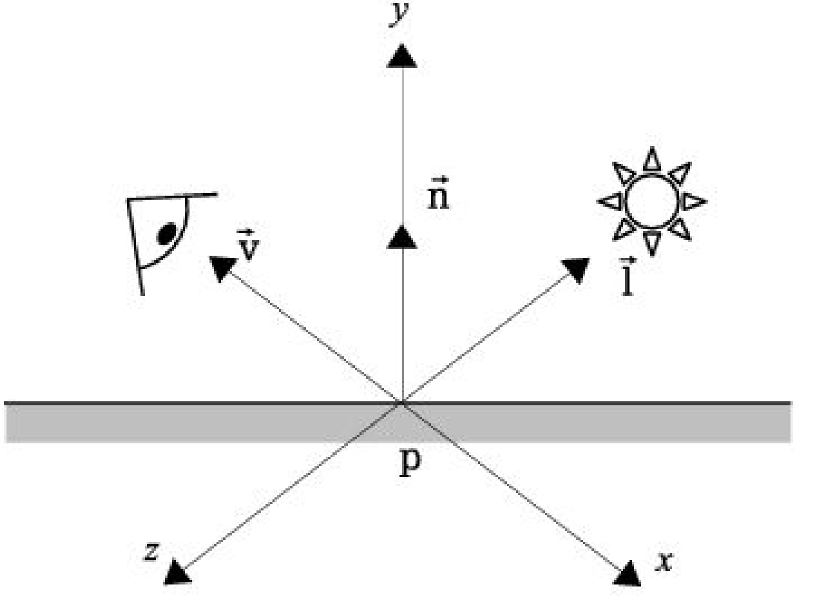
- Suppose the values of the emitted, ambient and incident intensity from the light source are:

$I_e = 2$ , $I_a = 1$, $I_i = 12$

- and that the constant values are:

$k_a = 0.3$, $k_d = 0.3$, $k_s = 0.6$

and $n = 3$



# Numerical Example (2)

- Light source is twelve times more intense than the ambient light and the object is self-illuminated and emits twice the ambient intensity
- $k_d + k_s = 0.9 \rightarrow$ 10% of the incident light is absorbed by the object
- Before applying the Phong formula we must compute the halfway vector and normalize all the vectors involved:

$$\hat{\mathbf{l}} = \frac{\vec{l}}{|\vec{l}\,|} = \frac{[1,1,0]^T}{\sqrt{1^2+1^2}} = [\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0]^T, \hat{\mathbf{v}} = \frac{\vec{v}}{|\vec{v}\,|} = \frac{[0,1,1]^T}{\sqrt{1^2+1^2}} = [0, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}]^T$$

$$\vec{h} = (\hat{\mathbf{l}} + \hat{\mathbf{v}})/2 = [\frac{1}{2\sqrt{2}}, \frac{1}{\sqrt{2}}, \frac{1}{2\sqrt{2}}]^T, \hat{\mathbf{h}} = \frac{\vec{h}}{|\vec{h}\,|} = \frac{[\frac{1}{2\sqrt{2}}, \frac{1}{\sqrt{2}}, \frac{1}{2\sqrt{2}}]^T}{\sqrt{3}/2} = [\frac{1}{\sqrt{2}\sqrt{3}}, \frac{\sqrt{2}}{\sqrt{3}}, \frac{1}{\sqrt{2}\sqrt{3}}]^T,$$

$$\hat{\mathbf{n}} = \frac{[0,2,0]^T}{\sqrt{2^2}} = [0,1,0]^T$$

# Numerical Example (3)

- We can now apply equation (12.25):

$$I = 2 + 1 \cdot 0.3 + 12 \cdot (0.3 \cdot (\frac{1}{\sqrt{2}}) + 0.6 \cdot (\frac{\sqrt{2}}{\sqrt{3}})^3) = 8.76$$

  - Final intensity value corresponds to the specified viewing angle and is related to the input intensities
  - The angle between the directions of reflection and viewing is:

$$\mathbf{rv} = 2\mathbf{nh} = 2arccos(\frac{\sqrt{2}}{\sqrt{3}}) = 70^\circ$$

  - If the viewing direction coincided with the direction of reflection (i.e. $\hat{\mathbf{v}} = [-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0]^T$ ) then the specular reflection would attain its max value

since $\mathbf{rv} = 2\mathbf{nh} = 2arccos(1) = 0^\circ$ :

$$\hat{\mathbf{h}} = [0,1,0]^T, \qquad I = 2 + 1 \cdot 0.3 + 12 \cdot (0.3 \cdot (\frac{1}{\sqrt{2}}) + 0.6 \cdot 1^3) = 12.05$$

# Phong Model Vectors

- The Phong model requires a number of vectors for the computation of the illumination value at a surface point, namely:

$$\vec{\mathbf{n}},$$
$$\vec{\mathbf{l}},$$
$$\vec{\mathbf{v}} \;\; \textit{and}$$
$$\vec{\mathbf{r}} \;\; \textit{or} \;\; \vec{\mathbf{h}}$$

- Important to use efficient formulae for the computation of these vectors, since such computation is repeated for every point where the model is applied

# The Normal Vector

- Defined as a vector perpendicular to a surface at a certain point
- Direction of normal vector defines the orientation of the surface
- Very useful in computer graphics

**Normal vector for implicit surfaces:**

- Implicit surfaces are defined by an equation of the form:

$$f\,(x, y, z) = 0$$

- The normal vector at a point **p** = $[a,b,c]^T$ of such a surface is given by the gradient vector in the vicinity of **p**:

$$\vec{\mathbf{n}} = \begin{bmatrix} \partial f / \partial x \\ \partial f / \partial y \\ \partial f / \partial z \end{bmatrix}$$

# The Normal Vector (2)

- In the case of a planar surface defined by:

$$f(x, y, z) = ax+by+cz+d = 0$$

- The normal vector, which is constant over the entire planar surface, is:

$$\vec{\mathbf{n}} = [a, b, c]^T$$

**<u>Normal vector for parametric surfaces:</u>**

- Surfaces are often represented parametrically
- In 3D, a surface is represented by 3 parametric equations in terms of 2 parameters *u* and *v:*

$$x = f_x(u, v)$$

$$y = f_y(u, v)$$

$$z = f_z(u, v)$$

# The Normal Vector (3)

- The normal vector is then:

$$\vec{\mathbf{n}} = \overrightarrow{\frac{\partial \mathbf{f}}{\partial \mathbf{u}}} \times \overrightarrow{\frac{\partial \mathbf{f}}{\partial \mathbf{v}}} \qquad (12.29)$$

*where*

$$\vec{\mathbf{f}} = \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix}, \overrightarrow{\frac{\partial \mathbf{f}}{\partial \mathbf{u}}} = \begin{bmatrix} \partial f_x / \partial u \\ \partial f_y / \partial u \\ \partial f_z / \partial u \end{bmatrix}, \overrightarrow{\frac{\partial \mathbf{f}}{\partial \mathbf{v}}} = \begin{bmatrix} \partial f_x / \partial v \\ \partial f_y / \partial v \\ \partial f_z / \partial v \end{bmatrix}$$

**Normal vector for polygons:**

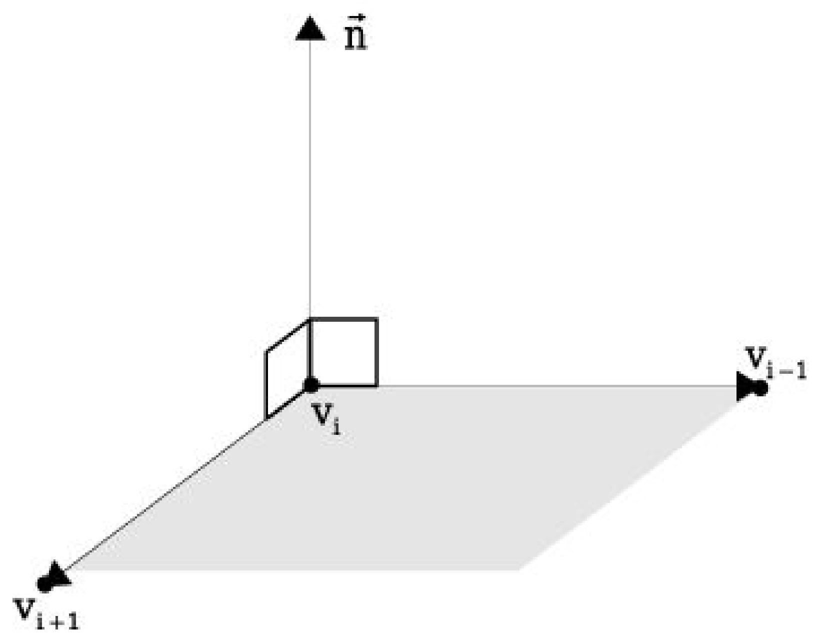
- Polygons are the usual building element for model composition
- In practice the equation of a polygon's plane is not known and the polygon is given in terms of a list of its vertices

# The Normal Vector (4)

- Given 3 consecutive, non-collinear vertices of a polygon $\mathbf{v_{i-1}}$, $\mathbf{v_i}$, and $\mathbf{v_{i+1}}$, we can compute the normal vector:

$$\vec{\mathbf{n}} = (\mathbf{v}_{i+1} - \mathbf{v}_i) \times (\mathbf{v}_{i-1} - \mathbf{v}_i)$$

- Cross product is **not** associative
- The above computation follows the right-hand rule:



# The Normal Vector (5)

- Must select correct orientation, otherwise normal computations will be reversed & objects will take an "inside-out" look
- For polygons with more than 3 vertices, a slight non-planarity may exist
  - We may compute the polygon normal as the average of the normal vectors given by each pair of consecutive polygon edges
  - Another technique, due to Martin-Newell:

if $[x_i, y_i, z_i]^T$, $i = 1,2, \ldots, n$ are the $n$ vertices of a polygon, then the coefficients $a,b,c$ of an approximating plane are computed as:

$$a = \sum_{i=1}^{n} (y_i - y_{i\oplus 1})(z_i + z_{i\oplus 1}), \quad b = \sum_{i=1}^{n} (z_i - z_{i\oplus 1})(x_i + x_{i\oplus 1}), \quad c = \sum_{i=1}^{n} (x_i - x_{i\oplus 1})(y_i + y_{i\oplus 1})$$

where $\oplus$ represents addition modulo $n$

The $d$ (constant) coefficient of the plane equation can be computed using the coordinates of a polygon vertex:

$$d = -(ax_1 + by_1 + cz_1)$$

# The Normal Vector (6)

- Another way of computing the normal vector of a polygon:

  if $[x_1, y_1, z_1]^T$, $[x_2, y_2, z_2]^T$, and $[x_3, y_3, z_3]^T$ are 3 non-collinear points, then they must satisfy the plane equation:

$$ax_1 + by_1 + cz_1 = -1$$
$$ax_2 + by_2 + cz_2 = -1$$
$$ax_3 + by_3 + cz_3 = -1$$

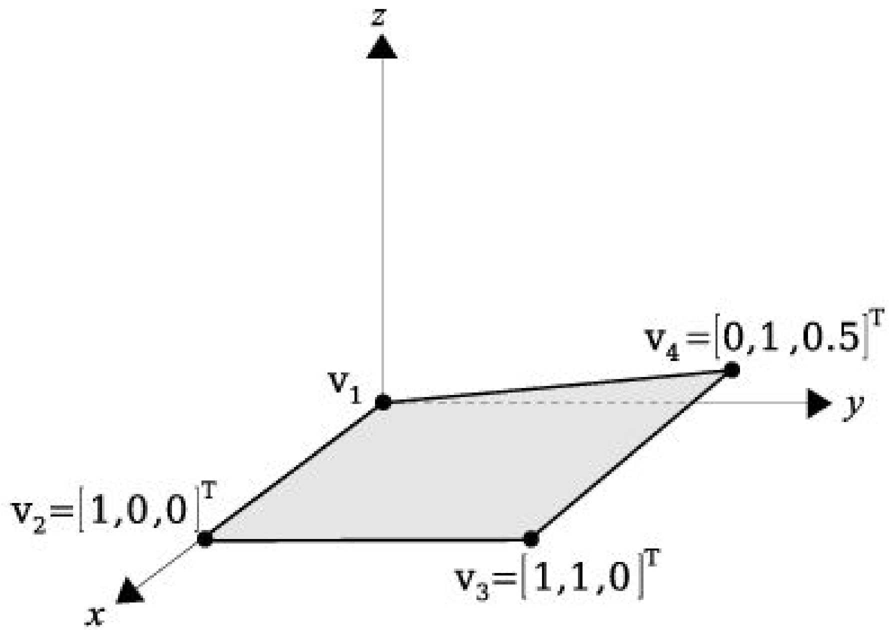
- Or

$$\begin{bmatrix} x1 & y1 & z1 \\ x2 & y2 & z2 \\ x3 & y3 & z3 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$$

- Or $\mathbf{XC = D}$
- So $\mathbf{C = X}^{-1}\boldsymbol{D}$

# Numerical Example

- Given polygon with vertices $\mathbf{v_1} = [0,0,0]^T$, $\mathbf{v_2} = [1,0,0]^T$, $\mathbf{v_3} = [1,1,0]^T$, and $\mathbf{v_4} = [0,1,0.5]^T$ (slightly non-planar). Compute its normal vector




- We shall consider two suitable methods:
  - 1) Average of the normals for each pair of successive edges
  - 2) Martin-Newell's technique

# Numerical Example (2)

**Method 1:**

- We first compute 4 normal vectors
- Normals are indexed by the vertex onto which both edges are incident

$$\vec{\mathbf{n}}_{\mathbf{v}_1} = [1,0,0]^T \times [0,1,0.5]^T = [0,-0.5,1]^T$$

$$\vec{\mathbf{n}}_{\mathbf{v}_2} = [0,1,0]^T \times [-1,0,0]^T = [0,0,1]^T$$

$$\vec{\mathbf{n}}_{\mathbf{v}_3} = [-1,0,0.5]^T \times [0,-1,0]^T = [0.5,0,1]^T$$

$$\vec{\mathbf{n}}_{\mathbf{v}_4} = [0,-1,-0.5]^T \times [1,0,-0.5]^T = [0.25,-0.5,1]^T$$

- Next, compute the polygon normal by averaging the above
- Normalize vectors before summation → give equal weight to all edges:

$$\vec{\mathbf{n}} = \frac{\hat{\mathbf{n}}_{\mathbf{v}_1} + \hat{\mathbf{n}}_{\mathbf{v}_2} + \hat{\mathbf{n}}_{\mathbf{v}_3} + \hat{\mathbf{n}}_{\mathbf{v}_4}}{4} = [0.17,-0.22,0.91]^T,$$

$$\hat{\mathbf{n}} = [0.18,-0.23,0.96]^T$$

# Numerical Example (3)

**Method 2:**

- Using Martin-Newell's technique ,we get:

$$a = 0 \cdot 0 + (-1) \cdot 0 + 0 \cdot 0.5 + 1 \cdot 0.5 = 0.5$$

$$b = 0 \cdot 1 + 0 \cdot 2 + (-0.5) \cdot 1 + 0.5 \cdot 0 = -0.5$$

$$c = (-1) \cdot 0 + 0 \cdot 1 + 1 \cdot 2 + 0 \cdot 0.5 = 2$$

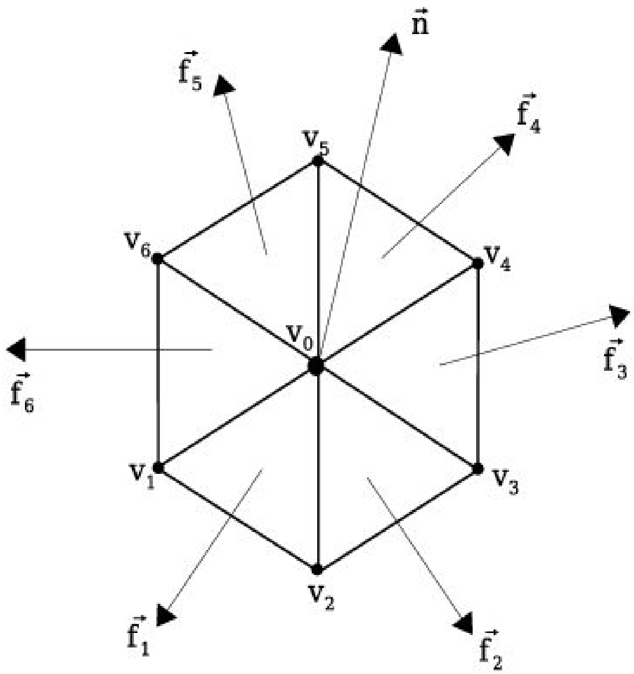
- Thus:

$$\vec{\mathbf{n}} = [0.5, -0.5, 2]^T ,$$

$$\hat{\mathbf{n}} = [0.24, -0.24, 0.94]^T$$

# Vertex Normal Vector for Polygonal Meshes

- Polygonal meshes are often used to approximate objects with smooth change of their surface normal vector
- Assume objects that consist of a single manifold surface
- In illumination, we need the normal vector to an object's surface at a discrete set of points covered by the surface :
  - Determine the normal at the vertices of the polygonal mesh as a weighted average of the normals to the adjacent faces to the vertex
  - Use this normal to perform bilinear interpolation along edges and finally across edges, on points of the underlying grid
- The polygons adjacent to a vertex are the *1-ring neighbors* or the *star* of the vertex



# Vertex Normal Vector for Polygonal Meshes (2)

- *Vertex normal* refers to a weighted average of the normals to the faces of the vertex's star
- 3 common approaches for computing the unit vertex normal $\hat{\mathbf{n}}$

**Approach 1:**

- Weights can be taken to be equal
- Achieved by normalizing the normals of the faces of the star $\vec{\mathbf{f}}_{\mathrm{i}}$ before averaging**:**

$$\hat{\mathbf{n}} = \frac{\sum_{i=1}^{m} \hat{\mathbf{f}}_{\mathbf{i}}}{\left| \sum_{i=1}^{m} \hat{\mathbf{f}}_{\mathbf{i}} \right|} \qquad (12.31)$$

where $\hat{\mathbf{f}}_{\mathrm{i}} = \vec{\mathbf{f}}_{\mathrm{i}} / |\vec{\mathbf{f}}_{\mathrm{i}}|$

*m*: number of faces in the star

# Vertex Normal Vector for Polygonal Meshes (3)

**Approach 2:**

- This approach observes that larger polygons should contribute more than smaller ones
  - Face normals are thus weighted by the area of the corresponding polygons
- For triangular faces, this amounts to taking the face normals as computed by the outer product of the vectors represented by 2 of the triangle's edges
  - This is because the outer product is equal to twice the area of the triangle

$$\hat{\mathbf{n}} = \frac{\sum_{i=1}^{m} \vec{\mathbf{f}}_{\mathbf{i}}}{|\sum_{i=1}^{m} \vec{\mathbf{f}}_{\mathbf{i}}|} \qquad (12.32)$$

# Vertex Normal Vector for Polygonal Meshes (4)

**Approach 3:**

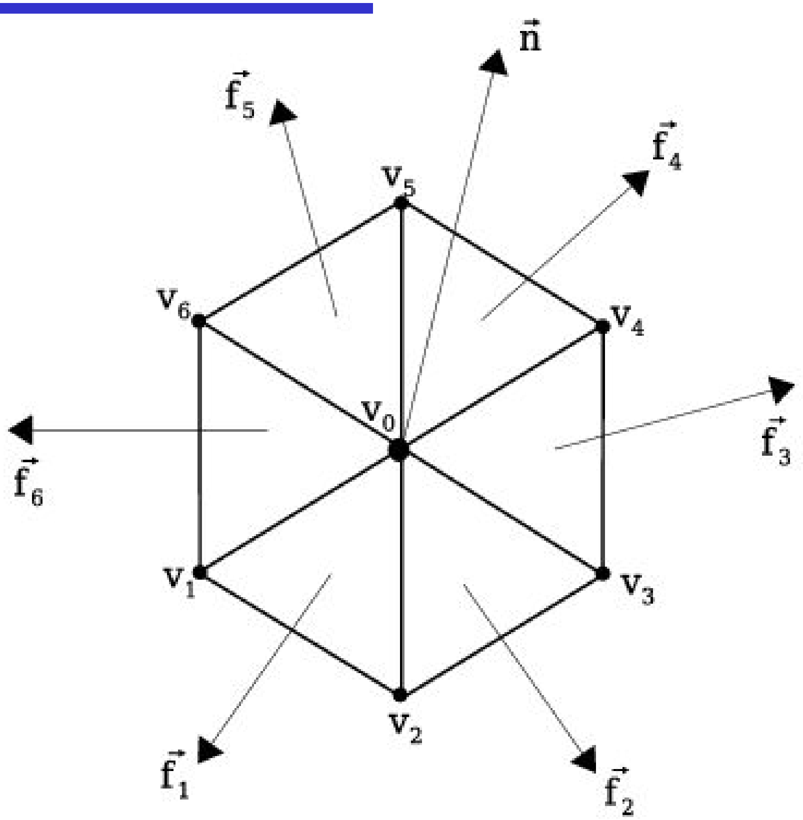
- This approach observes that in order to ensure that vertex normals are invariant to mesh restructuring, a good weight is the incident angle $\theta$ of the faces of the star
- Angle $\theta$ can be computed by taking the *arccos* of the dot product of the vectors defined by the incident edges that form it:

$$\hat{\mathbf{n}} = \frac{\sum_{i=1}^{m} \theta_i \hat{\mathbf{f}}_{\mathbf{i}}}{\left| \sum_{i=1}^{m} \theta_i \hat{\mathbf{f}}_{\mathbf{i}} \right|} \qquad (12.33)$$

- Note: Vertex normals should be computed before the perspective division (projection)

# Vertex Normal: Symbolic Example

- In the following figure *m* is 6 as there are 6 polygons in the star
- In order to evaluate all the vertex normal expressions above, we need to compute the $\vec{\mathbf{f}}_i$, $\hat{\mathbf{f}}_i$, $\theta_i$
- Take the first triangle $\boldsymbol{v_0 v_1 v_2}$:



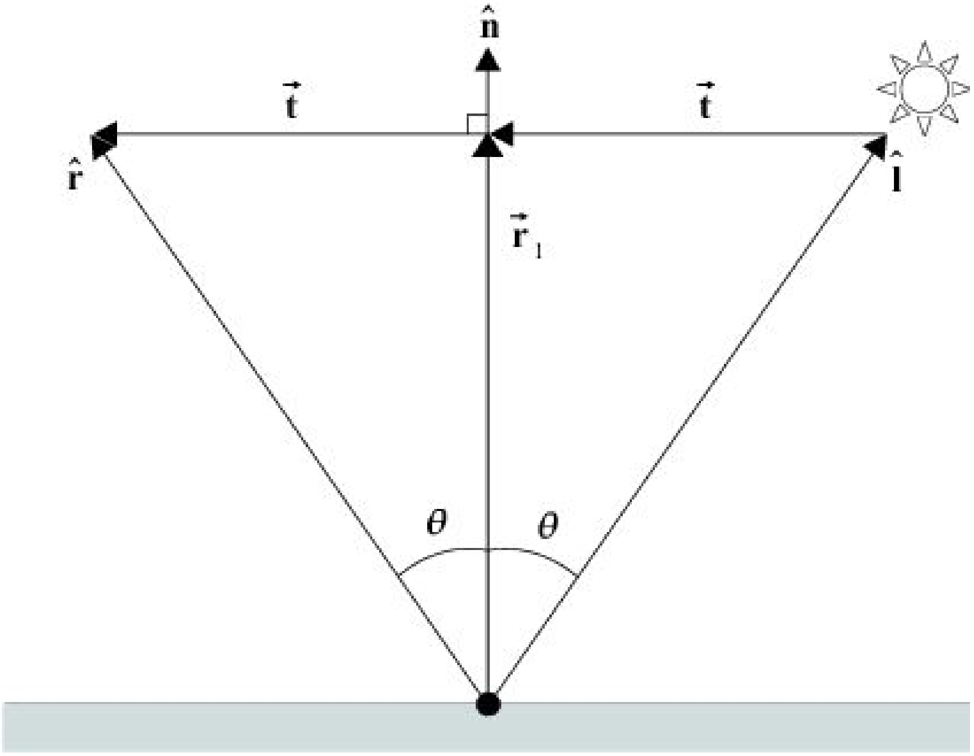

$$\overrightarrow{\mathbf{f}_1} = \overrightarrow{\mathbf{v}_0\mathbf{v}_1} \times \overrightarrow{\mathbf{v}_0\mathbf{v}_2}, \quad \hat{\mathbf{f}}_1 = \frac{\overrightarrow{\mathbf{f}_1}}{|\overrightarrow{\mathbf{f}_1}|}$$

$$\theta_1 = \arccos\left(\frac{\overrightarrow{\mathbf{v}_0\mathbf{v}_1}}{|\overrightarrow{\mathbf{v}_0\mathbf{v}_1}|} \cdot \frac{\overrightarrow{\mathbf{v}_0\mathbf{v}_2}}{|\overrightarrow{\mathbf{v}_0\mathbf{v}_2}|}\right)$$

- Similar computations are performed for the other five triangles in the star and expressions (12.31) – (12.33) can then be evaluated

# Reflection Vector

- *Reflection vector* $\vec{\mathbf{r}}$ is computed by noticing that:
  - Angles between pairs of vectors $(\hat{\mathbf{l}},\hat{\mathbf{n}})$ , $(\hat{\mathbf{n}},\vec{\mathbf{r}})$ are equal
  - $\hat{\mathbf{l}}$, $\hat{\mathbf{n}}$, $\vec{\mathbf{r}}$ are coplanar



# Reflection Vector (2)

- Let $\vec{\mathbf{r}}_1$ be the vector defined by the projection of $\hat{\mathbf{l}}$ onto the axis of $\hat{\mathbf{n}}$:

$$|\overrightarrow{\mathbf{r}_1}| = |\hat{\mathbf{l}}|\cos\theta = |\hat{\mathbf{l}}|(\hat{\mathbf{n}}\cdot\hat{\mathbf{l}}) = \hat{\mathbf{n}}\cdot\hat{\mathbf{l}},$$

since $\hat{\mathbf{l}}$ is a unit vector, so:

$$\overrightarrow{\mathbf{r}_1} = \hat{\mathbf{n}}\,|\overrightarrow{\mathbf{r}_1}| = \hat{\mathbf{n}}(\hat{\mathbf{n}}\cdot\hat{\mathbf{l}})$$

We also have:

$$\vec{\mathbf{r}} = \overrightarrow{\mathbf{r}_1} + \vec{\mathbf{t}} \qquad \vec{\mathbf{t}} = \overrightarrow{\mathbf{r}_1} - \hat{\mathbf{l}}$$

- Thus:

$$\vec{\mathbf{r}} = 2\overrightarrow{\mathbf{r}_1} - \hat{\mathbf{l}} = 2\hat{\mathbf{n}}(\hat{\mathbf{n}}\cdot\hat{\mathbf{l}}) - \hat{\mathbf{l}}$$

- 6 multiplications and 5 additions are required
- When performance is an issue, the reflection vector is replaced by the halfway vector

# Light, View & Halfway Vectors

- *Light* and *View* vectors $\vec{\mathbf{l}}, \vec{\mathbf{v}}$ are:
  - either given constant vectors, if the light & view points are placed at infinity,
  - or simply computed as:

$$\vec{\mathbf{l}} = \mathbf{l} - \mathbf{p} \qquad (12.35)$$

$$\vec{\mathbf{v}} = \mathbf{v} - \mathbf{p} \qquad (12.36)$$

where **p**: object point

**l**: light point

**v**: view point

- *Halfway* vector $\vec{\mathbf{h}}$, useful for specular reflection, is then computed as:

$$\vec{\mathbf{h}} = (\hat{\mathbf{l}} + \hat{\mathbf{v}}) / 2 \qquad (12.37)$$

# Illumination Algorithms History

- Illumination is applied to produce realistic synthetic images
- Warnock (1969):
  - intensity diminishes according to depth
  - objects were illuminated according to their distance from the light source
- Gouraud (1971):
  - interpolation of intensity values within polygons from intensity values computed at the vertices
- Phong :
  - compute intensity values at every pixel by linearly interpolating vertex normals
  - using the model he introduced in 1975
  - there are instances where the linear interpolation of the vertex normals does not work well
- Overveld (1997) :
  - proposed a quadratic interpolation scheme

# Illumination Algorithms based on the Phong Model

- Constant shading
- Gouraud shading
- Phong shading

# Constant Shading

- Is the simplest algorithm for polygonal objects
- Applies a constant illumination value to each polygonal facet
- Incorporated:
  - Constant ambient lighting
  - Diffuse reflection
- No Specular reflection
- The light & view points:
  - Are both placed at infinity and coincide, $\vec{\mathbf{l}} = \vec{\mathbf{v}}$
  - Eliminates shadows
  - $(\hat{\mathbf{n}} \cdot \hat{\mathbf{l}})$ is constant for each polygon

# Constant Shading (2)

- If the light & view points are on the +z-axis:
  - $\hat{\mathbf{l}} = \hat{\mathbf{v}} = [0, 0, 1]^T$
  - $(\hat{\mathbf{n}} \cdot \hat{\mathbf{l}}) = n_z$ for $\hat{\mathbf{n}} = [n_x, n_y, n_z]^T$
- Illumination equation:

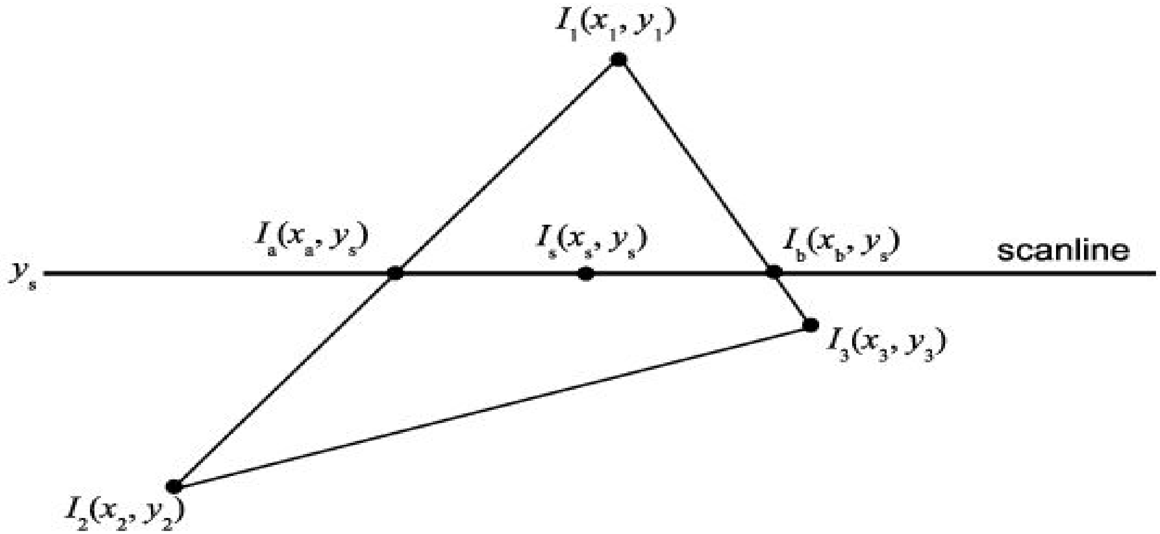
$$I = I_e + I_a k_a + I_i k_d n_z$$

- $I$ is computed once for each polygon
  - used for all pixels that the polygon covers
- Problem:
  - A polygon mesh often samples a curved surface
  - The human eye is sensitive to intensity discontinuities
  - Polygon silhouettes stand out → objects have a "polygonal look"
- One solution:
  - Use some form of illumination interpolation

# Gouraud Shading

- Is a simple illumination interpolation algorithm
- If the sampling density is sufficiently high, it can capture local maxima (highlights) and minima of shading distribution over the polygon mesh
- It computes intensity values for pixels inside a polygon:
  i. Interpolate the intensity values at its vertices
     - Intensity values at vertices estimated using the Phong model
     - Use vertex normals to evaluate the Phong equation at the vertices
  ii. Bi-linearly interpolate the vertex intensities along the polygon edges & between the edges (along the scanlines)
     - Scalar interpolation

# Gouraud Shading (2)




- Intensities $I_1$, $I_2$, $I_3$ are computed using the Phong model
- $I_a$, $I_b$ : using interpolation between ($I_1$, $I_2$, $I_3$ ):

$$I_{\mathbf{a}} = I_1 \frac{y_{\mathbf{s}} - y_2}{y_1 - y_2} + I_2 \frac{y_1 - y_{\mathbf{s}}}{y_1 - y_2} = \frac{1}{y_1 - y_2}(I_1(y_{\mathbf{s}} - y_2) + I_2(y_1 - y_{\mathbf{s}}))$$

$$I_{\mathbf{b}} = \frac{1}{y_1 - y_3}(I_1(y_{\mathbf{s}} - y_3) + I_3(y_1 - y_{\mathbf{s}}))$$

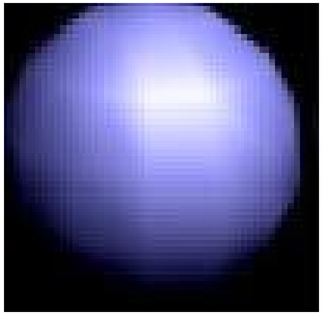
- $I_s$ : using interpolation between $I_a$, $I_b$ : $I_{\mathbf{s}} = \frac{1}{x_{\mathbf{b}} - x_{\mathbf{a}}}(I_{\mathbf{a}}(x_{\mathbf{b}} - x_{\mathbf{s}}) + I_{\mathbf{b}}(x_{\mathbf{s}} - x_{\mathbf{a}}))$

# Gouraud Shading (3)

- Intensity values are computed **incrementally** within a scanline:
- If $\mathbf{s}_1$ and $\mathbf{s_2}$ are the indices of 2 pixels on the same scanline, then:

$$I_{\mathbf{s}_1} = \frac{1}{x_{\mathbf{b}} - x_{\mathbf{a}}}\left(I_{\mathbf{a}}(x_{\mathbf{b}} - x_{\mathbf{s}_1}) + I_{\mathbf{b}}(x_{\mathbf{s}_1} - x_{\mathbf{a}})\right)$$

$$I_{\mathbf{s}_2} = \frac{1}{x_{\mathbf{b}} - x_{\mathbf{a}}}\left(I_{\mathbf{a}}(x_{\mathbf{b}} - x_{\mathbf{s}_2}) + I_{\mathbf{b}}(x_{\mathbf{s}_2} - x_{\mathbf{a}})\right)$$



- Subtracting the above equations:

$$\Delta I_{\mathbf{s}} = I_{\mathbf{s}_2} - I_{\mathbf{s}_1} = \frac{x_{\mathbf{s}_2} - x_{\mathbf{s}_1}}{x_{\mathbf{b}} - x_{\mathbf{a}}}(I_{\mathbf{b}} - I_{\mathbf{a}}) = \frac{\Delta x}{x_{\mathbf{b}} - x_{\mathbf{a}}}(I_{\mathbf{b}} - I_{\mathbf{a}})$$

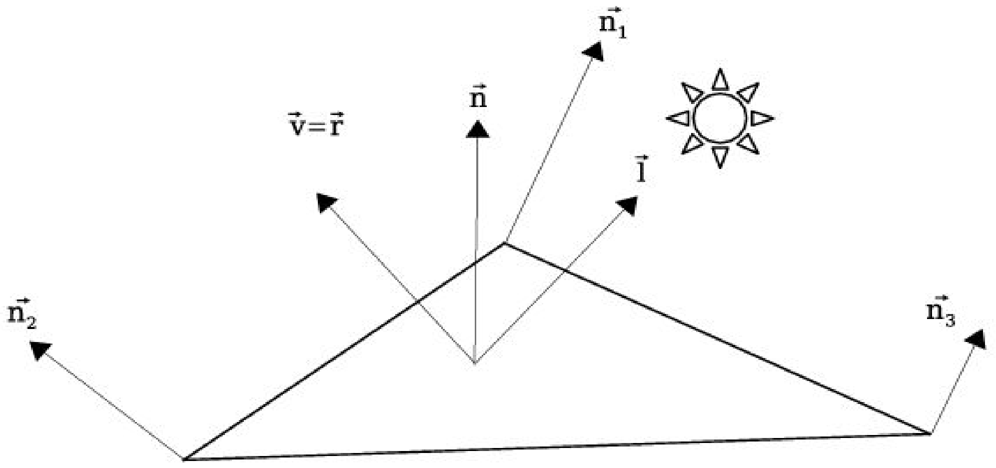
- In the case of neighboring pixels ($\Delta x$=1): $\Delta I_{\mathbf{s}} = \dfrac{I_{\mathbf{b}} - I_{\mathbf{a}}}{x_{\mathbf{b}} - x_{\mathbf{a}}}$

- Incremental intensity computation:

$$I_{\mathbf{s},n} = I_{\mathbf{s},n-1} + \Delta I_{\mathbf{s}}$$

# Phong Shading

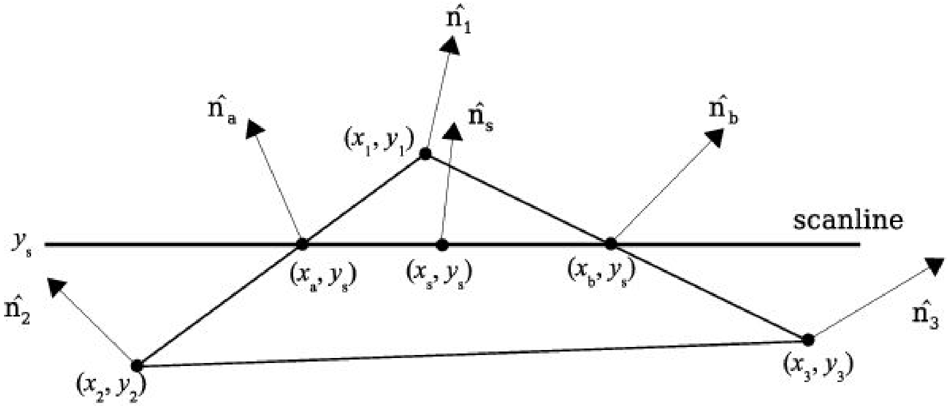
- Problems of Gouraud Shading:
  - The sampling density is rarely sufficient to capture highlights
    - The shading vectors are not interpolated within the polygon but are used to capture intensities at the vertices only
  - No elimination of mach-bands:
    - the linear intensity interpolation leaves second-order intensity discontinuities




- The Phong algorithm solves the problems of Gouraud shading by applying the Phong model to each pixel covered by a polygon

# Phong Shading (2)

- Phong algorithm computations:




  - The unit normal vectors: bi-linear interpolation from the unit vertex normals

$$\vec{\mathbf{n}}_{\mathbf{a}} = \frac{1}{y_1 - y_2}(\hat{\mathbf{n}}_1(y_{\mathbf{s}} - y_2) + \hat{\mathbf{n}}_2(y_1 - y_{\mathbf{s}}))$$

$$\vec{\mathbf{n}}_{\mathbf{b}} = \frac{1}{y_1 - y_3}(\hat{\mathbf{n}}_1(y_{\mathbf{s}} - y_3) + \hat{\mathbf{n}}_3(y_1 - y_{\mathbf{s}}))$$

$$\hat{\mathbf{n}}_{\mathbf{s}} = \frac{1}{x_{\mathbf{b}} - x_{\mathbf{a}}}(\hat{\mathbf{n}}_{\mathbf{a}}(x_{\mathbf{b}} - x_{\mathbf{s}}) + \hat{\mathbf{n}}_{\mathbf{b}}(x_{\mathbf{s}} - x_{\mathbf{a}}))$$

  - For neighboring pixels on the same scanline, use incremental computation

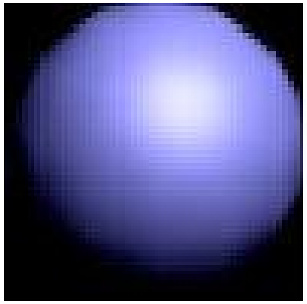
$$n_{\mathrm{s}x,n} = n_{\mathrm{s}x,n-1} + \Delta n_{\mathrm{s}x} \qquad \Delta n_{\mathbf{s}x} = \frac{n_{\mathbf{b}x} - n_{\mathbf{a}x}}{x_{\mathbf{b}} - x_{\mathbf{a}}}$$

$$n_{\mathrm{s}y,n} = n_{\mathrm{s}y,n-1} + \Delta n_{\mathrm{s}y} \qquad \text{where} \qquad \Delta n_{\mathbf{s}y} = \frac{n_{\mathbf{b}y} - n_{\mathbf{a}y}}{x_{\mathbf{b}} - x_{\mathbf{a}}}$$

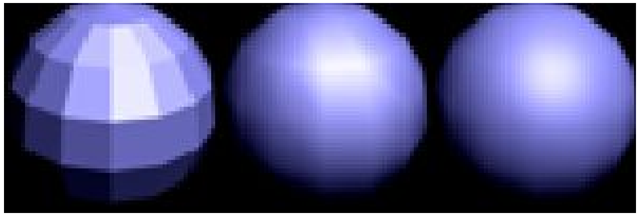
$$n_{\mathrm{s}z,n} = n_{\mathrm{s}z,n-1} + \Delta n_{\mathrm{s}z} \qquad \Delta n_{\mathbf{s}z} = \frac{n_{\mathbf{b}z} - n_{\mathbf{a}z}}{x_{\mathbf{b}} - x_{\mathbf{a}}}$$

# Phong Shading (3)

- The Phong algorithm:
  - Is a significant improvement over Gouraud
  - Requires considerably more computations
  - Is implemented on graphics accelerators

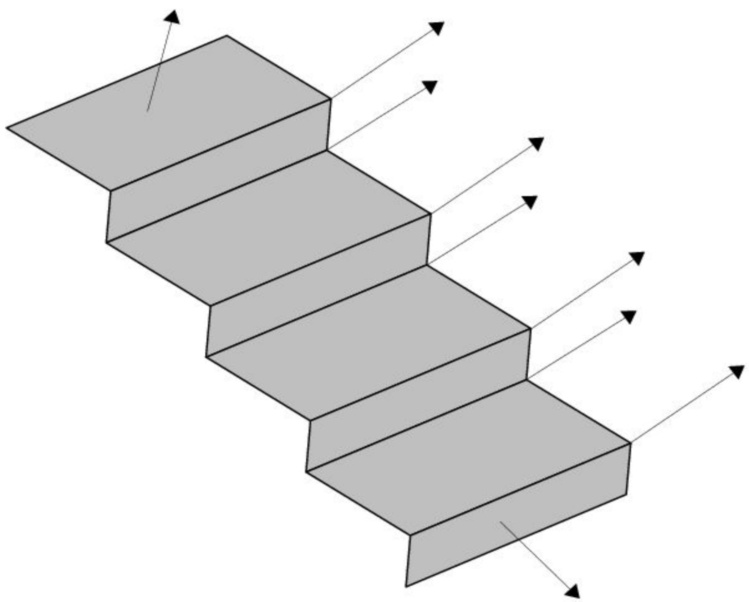


Constant – Gouraud – Phong shading example:



# Quadratic Interpolation of Vertex Normals

- Phong shading algorithm
  - polygonal mesh sufficiently dense → acceptable quality
  - large polygons → shading artifacts
- The *silhouette edge problem:*
  - In e.g. below, normal vectors do not vary at all over the surface
  - Completely flat illumination appearance
  - This is at odds with the appearance of the silhouette



# Quadratic Interpolation of Vertex Normals (2)

- Vertex normal interpolation essentially aims to reconstruct a surface from discrete samples
- Reconstruction cannot add information
- Tries to come up with a reconstructed surface consistent with the sampled data:
  - interpolates the vertex normals
  - is perpendicular to the face normals
- The linear interpolation of vertex normals in Phong shading is not consistent in this sense (previous e.g.)

# Quadratic Interpolation of Vertex Normals (3)

- Overveld and Wyvill showed that the quadratic interpolation of normals achieves better results
- If $\hat{\mathbf{n}}_0, \hat{\mathbf{n}}_1$ **:** the normal vectors to be interpolated and
  $\vec{\delta}$: the vector defined by the subtraction of the 1st from the last interpolation point
- Then the interpolated vector $\vec{\mathbf{n}}(s)$ is:

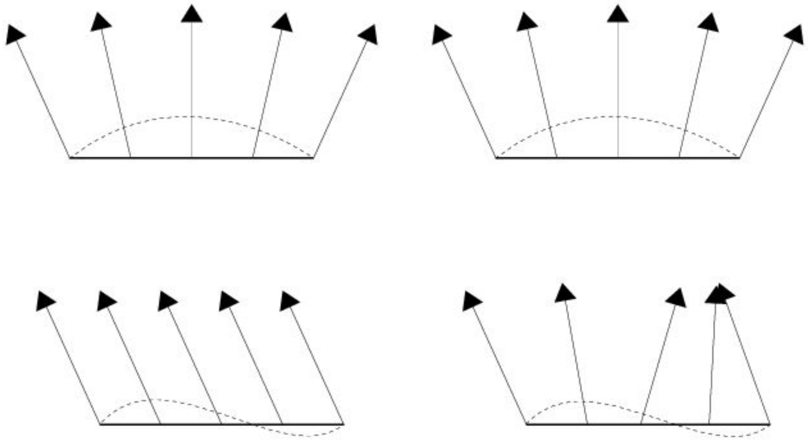
$$\vec{\mathbf{n}}(s) = \hat{\mathbf{n}}_0 + s\vec{\mathbf{a}} + s^2\vec{\mathbf{b}} \quad \text{with } s \in [0..1] \text{ and}$$

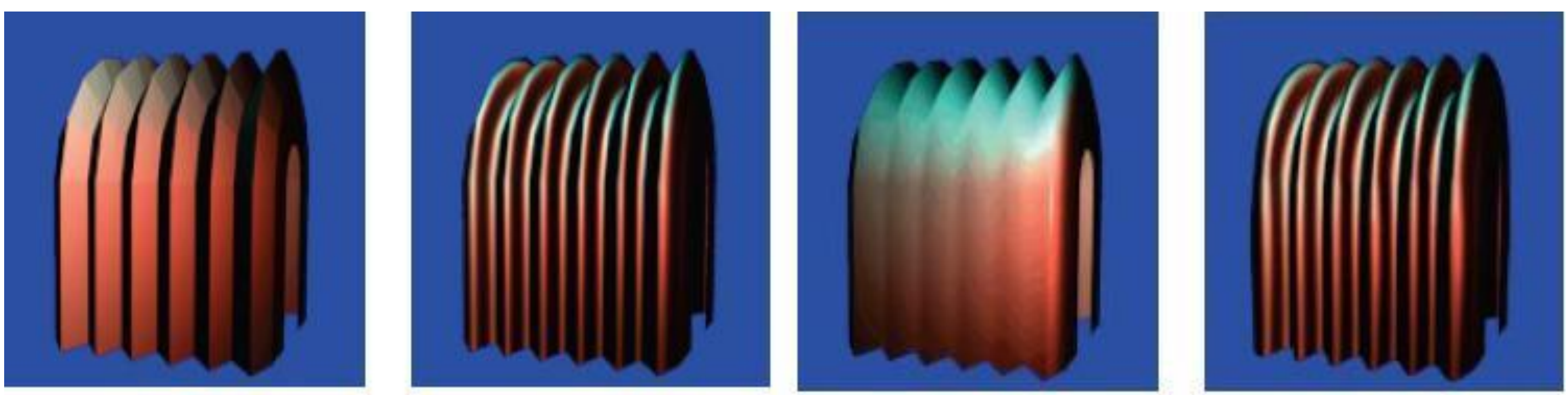
$$\vec{\mathbf{a}} = \hat{\mathbf{n}}_1 - \hat{\mathbf{n}}_0 - \vec{\mathbf{b}} \ , \ \ \vec{\mathbf{b}} = 3\left(\frac{(\hat{\mathbf{n}}_0 + \hat{\mathbf{n}}_1)\cdot\vec{\delta}}{\vec{\delta}^2}\right)\vec{\delta}$$

- As expected: $\vec{\mathbf{n}}(0) = \hat{\mathbf{n}}_0$ and $\vec{\mathbf{n}}(1) = \hat{\mathbf{n}}_1$
- Implemented by taking the forward differences of the quadratic function, at a cost of 2 vector additions per pixel

# Quadratic Interpolation of Vertex Normals (4)

- Linear (left) vs Quadratic (right) vector interpolation:



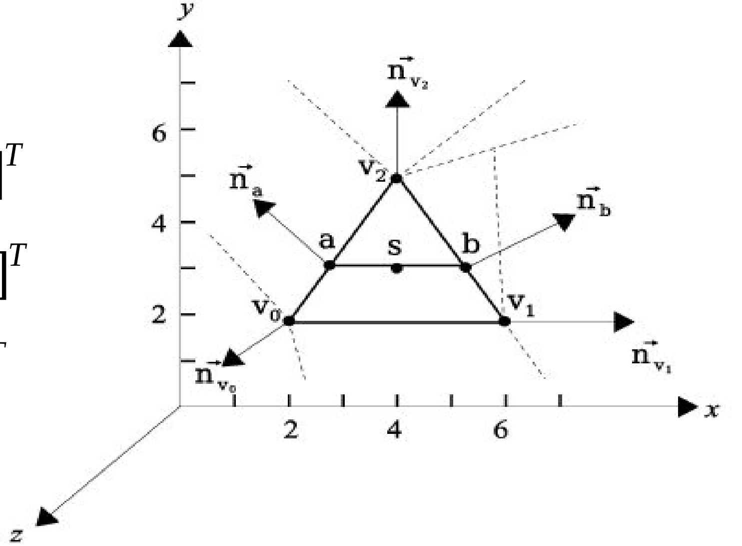


(a) (b) (c) (d)

flat quadratic linear linear/dense mesh

# Numerical Example

- Given the triangle mesh:




$$\mathbf{v_0} = [2,2,1]^T \qquad \mathbf{v_1} = [6,2,1]^T \qquad \mathbf{v_2} = [4,5,1]^T$$

$$\overrightarrow{\mathbf{n}_{\mathbf{v}_0}} = [-1,-1,1]^T \qquad \overrightarrow{\mathbf{n}_{\mathbf{v}_1}} = [1,0,0]^T \qquad \overrightarrow{\mathbf{n}_{\mathbf{v}_2}} = [0,1,1]^T$$

$$\mathbf{a} = [2.66,3,1]^T \qquad \mathbf{b} = [5.33,3,1]^T \qquad \mathbf{s} = [4,3,1]^T$$

- Assume emitted, ambient & incident intensities from light source:

$$I_e = 2, \quad I_a = 1, \quad I_i = 12$$

- And constant values:

$$k_a = 0.3\ , \quad k_d = 0.3, \quad k_s = 0.6, \quad n = 3$$

- Also assume the light and view points at infinity on the +z-axis :

$$\hat{\mathbf{l}} = \hat{\mathbf{v}} = [0,0,1]^T$$

# Numerical Example – Constant Shading

- Compute the polygon normal:

$$\vec{\mathbf{n}} = (\mathbf{v_1} - \mathbf{v_0}) \times (\mathbf{v_2} - \mathbf{v_0}) = [0, 0, 12]^T$$

$$\text{or} \;\; \hat{\mathbf{n}} = [0, 0, 1]^T$$

- From equation $I = I_e + I_a k_a + I_i k_d n_z$:

$$I = 2 \; + \; 1 \cdot 0.3 \; + \; 12 \cdot 0.3 \cdot 1 = \; \mathbf{5.9}$$

# Numerical Example – Gouraud Shading

- Normalize the vertex normals:

$$\hat{\mathbf{n}}_{\mathbf{v}_0} = [-\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}]^T \quad \hat{\mathbf{n}}_{\mathbf{v}_1} = [1,0,0]^T \quad \hat{\mathbf{n}}_{\mathbf{v}_2} = [0, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}]^T$$

- Use the Phong model to compute the intensities at the vertices:

$$I_{\mathbf{v}_0} = 2 + 1 \cdot 0.3 + 12(0.3(\hat{\mathbf{n}}_{\mathbf{v}_0} \cdot \hat{\mathbf{l}}) + 0.6(\hat{\mathbf{n}}_{\mathbf{v}_0} \cdot \hat{\mathbf{h}})^3) = 5.76$$

$$I_{\mathbf{v}_1} = 2 + 1 \cdot 0.3 + 12(0.3(\hat{\mathbf{n}}_{\mathbf{v}_1} \cdot \hat{\mathbf{l}}) + 0.6(\hat{\mathbf{n}}_{\mathbf{v}_1} \cdot \hat{\mathbf{h}})^3) = 2.3$$

$$I_{\mathbf{v}_2} = 2 + 1 \cdot 0.3 + 12(0.3(\hat{\mathbf{n}}_{\mathbf{v}_2} \cdot \hat{\mathbf{l}}) + 0.6(\hat{\mathbf{n}}_{\mathbf{v}_2} \cdot \hat{\mathbf{h}})^3) = 7.39$$

- From Gouraud shading equation:

$$I_{\mathbf{a}} = \frac{1}{3}(1 \cdot I_{\mathbf{v}_2} + 2 \cdot I_{\mathbf{v}_0}) = 6.3$$

$$I_{\mathbf{b}} = \frac{1}{3}(1 \cdot I_{\mathbf{v}_2} + 2 \cdot I_{\mathbf{v}_1}) = 4.0$$

$$I_{\mathbf{s}} = \frac{1}{2.67}(1.33 \cdot I_{\mathbf{a}} + 1.33 \cdot I_{\mathbf{b}}) = \mathbf{5.13}$$

# Numerical Example – Phong Shading

- Compute the normals at the edge points **a**, **b** from the unit vertex normals (by linear interpolation):

$$\overrightarrow{\mathbf{n}_\mathbf{a}} = \frac{1}{3}(1 \cdot \hat{\mathbf{n}}_{v_2} + 2 \cdot \hat{\mathbf{n}}_{\mathbf{v}_0}) = [-0.39, 0.15, 0.62]^T$$

$$\overrightarrow{\mathbf{n}_\mathbf{b}} = \frac{1}{3}(1 \cdot \hat{\mathbf{n}}_{\mathbf{v}_2} + 2 \cdot \hat{\mathbf{n}}_{\mathbf{v}_1}) = [0.67, 0.71, 0.71]^T$$

- Convert them to unit vectors:

$$\hat{\mathbf{n}}_\mathbf{a} = [-0.52, 0.2, 0.83]^T \quad , \quad \hat{\mathbf{n}}_\mathbf{b} = [0.55, 0.59, 0.59]^T$$

- Compute the unit normal vector at scanline point **s**

$$\overrightarrow{\mathbf{n}_\mathbf{s}} = \frac{1}{2.67}(1.33 \cdot \hat{\mathbf{n}}_\mathbf{a} + 1.33 \cdot \hat{\mathbf{n}}_\mathbf{b}) = [0.02, 0.4, 0.71]^T = [0.02, 0.49, 0.87]^T$$

- Apply the Phong model using the unit normal vector $\hat{\mathbf{n}}_\mathbf{s}$:

$$I_\mathbf{s} = 2 + 1 \cdot 0.3 + 12(0.3(\hat{\mathbf{n}}_\mathbf{s} \cdot \hat{\mathbf{l}}) + 0.6(\hat{\mathbf{n}}_\mathbf{s} \cdot \hat{\mathbf{h}})^3) = \mathbf{10.25}$$

# Numerical Example

- Phong shading gives significantly higher intensity value compared to Constant or Gouraud shading
- This is explained by the existence of a highlight at **s**
- The quadratic interpolation scheme computes $I_s$ similarly to Phong
  - The only difference is the quadratic formulae used for the computation of $\hat{\mathbf{n}}_{\mathbf{a}}, \hat{\mathbf{n}}_{\mathbf{b}}$ and $\hat{\mathbf{n}}_{\mathbf{s}}$

# The Cook-Torrance Illumination Model*

- Problems with the Phong model:
  - Objects often appear too plastic
  - The metallic shine or the off-specular-direction highlights are not captured correctly for many shiny materials
  - The reflected light scattering distribution due to the geometric variation of a rough surface cannot be captured
- Cook – Torrance Model:
  - Extension of the Phong model
  - General illumination model for rough surfaces
  - Takes into account the directional distribution and the wavelength dependence of the reflected light

# The Cook-Torrance Illumination Model (2)

- Distinguishes the reflected light into :
  - The ambient term
  - The diffuse scattering
  - The specular highlight
- Provides a modeling & parameterization of the BRDF $f_r$ of a material
- The BRDF $f_r$ is linearly composed of :
  - a pure diffuse term
  - a pure specular term

$$f_r = k_d f_d + k_s f_s, \qquad k_d + k_s = 1$$

# The Cook-Torrance Illumination Model (3)

- The Cook-Torrance reflectance model for $N_L$ light sources:

$$I_r = I_a f_a + \sum_{l=1}^{N_L} I_i^{(l)} (\hat{\mathbf{n}} \cdot \hat{\mathbf{l}}^{(l)}) \, [k_s f_s + k_d f_d] \, d\vec{\omega}_i^{(l)}$$

where:

  - $I_i^{(l)}$ : the incident light intensity from light source $l$ located at a direction $\hat{\mathbf{l}}^{(l)}$ through a solid angle $\vec{\omega}_i^{(l)}$
  - $\hat{\mathbf{n}}$ : the normal vector at the given surface location
- $I_a f_a$ is the ambient term & $I_a$ can be regarded as constant
- $f_d$ is the diffuse BRDF of a Lambertian surface
- $f_a$ uses the same distribution as $f_d$
- The specular part of the BRDF depends on:
  - the relative location of the observer
  - the properties of the material

# The Cook-Torrance Illumination Model (4)

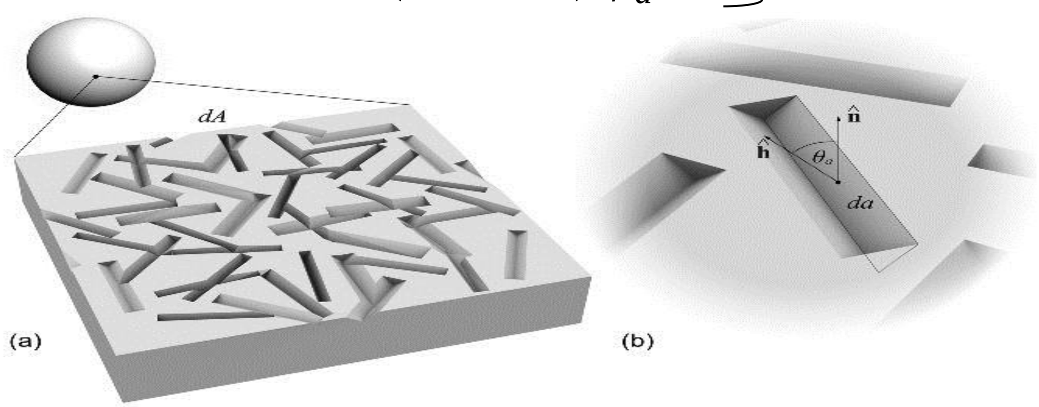
- In the original paper $I_a$ was multiplied by a visibility factor $f$:
  - $f$ is the amount of incoming ambient light that was not blocked by the surrounding environment
  - A distant uniformly luminous hemisphere radiates light toward the inspected surface point **p**
  - Uses a binary visibility function $V(\mathbf{p}, \hat{\mathbf{l}})$ with max value 1 when there is a clear line of sight between point **p** and the surrounding distant hemisphere in direction $\hat{\mathbf{l}}$:

$$f = \int_{unblocked\ \Omega} (\hat{\mathbf{n}} \cdot \hat{\mathbf{l}}) d\vec{\omega} = \int_{\Omega} (\hat{\mathbf{n}} \cdot \hat{\mathbf{l}}) V(\mathbf{p}, \hat{\mathbf{l}}) d\vec{\omega}$$

# The Cook-Torrance Illumination Model (5)

- The micro-facet model of Torrance and Sparrow is used for the derivation of the specular term $f_s$
  - a surface is assumed to be composed of long symmetric V-shaped grooves
  - each groove consists of two planar facets
- The facets:
  - Are tilted at equal but opposite angles to the surface normal at $dA$
  - Are considered perfect mirrors
  - Reflect light only in the direction of perfect reflection
  - The slope of the facets (polar angle) $\theta_a$
  - The orientation of the cavities (azimuth) $\varphi_a$

} determined by a statistical distribution for the material



# The Cook-Torrance Illumination Model (6)

- *da* : the area of a micro-facet
- *dA* : the inspected area, where reflectance is calculated
- In order for the Torrance-Sparrow model to work:
  - $da \ll dA$
  - wavelength $\lambda$ of the incident light << micro-facet dimensions →
    - avoid interference phenomena
    - be able to work with geometrical optics
    - dispense with wave theory
- Cook-Torrance model depends on
  - Micro-facet distribution term *D*
  - Geometric term *G*
  - Fresnel term *F*

# The Cook-Torrance Illumination Model (7)

## The Micro-facet distribution D

- Is the fraction of micro-facets aligned with direction $\hat{\mathbf{h}}$
- The contribution of each facet is binary :
  - light reflected fullyfrom direction $\hat{\mathbf{l}}$ to $\hat{\mathbf{v}}$
  - or, no light reflected at all
- Determines the fraction of incident light reflected back to the environment in the direction of view. Several possibilities:
- Gaussian distribution (easier to compute)

$$D_{(Gaussian)} = c \cdot e^{-(\theta_a / m)^2}$$

- Beckmann distribution (more physically correct)

$$D_{(Beckmann)} = \frac{1}{m^2 \cos^4 \theta_a} \cdot e^{-(\tan \theta_a / m)^2}$$

where $m$: the RMS slope of the surface

$\theta_a$ : the angle between the normal $\hat{\mathbf{n}}$ of $dA$ and $\hat{\mathbf{h}}$ of $da$

# The Cook-Torrance Illumination Model (8)

- The larger *m* is:
  - the more rough the surface
  - the specular highlight is spread out
- Small *m* :
  - Micro-facets with normal vectors closer to $\hat{\mathbf{n}}$
  - The material has a polished look
  - Specular highlight is tighter

# The Cook-Torrance Illumination Model (9)

<u>Incoming light interception</u>

- Some of the outgoing light in the direction of $\vec{\mathbf{v}}$ is attenuated due to the interception by blocking geometry
- The amount of blocking depends on
  - the outgoing direction
  - the slope of the micro-facet relative to $\hat{\mathbf{n}}$
- The amount of light blocked due to light interception $G_{intercept} \in [0,1]$ is:

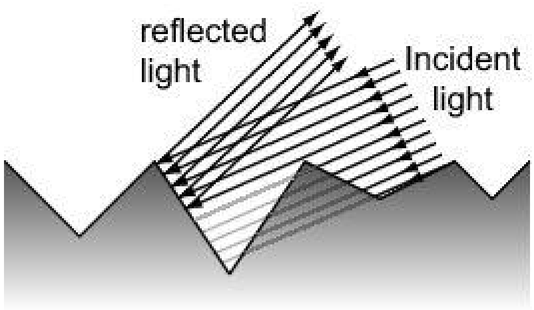
$$G_{intercept} = \frac{2(\hat{\mathbf{n}} \cdot \hat{\mathbf{h}})(\hat{\mathbf{n}} \cdot \hat{\mathbf{v}})}{\hat{\mathbf{v}} \cdot \hat{\mathbf{h}}}$$

# The Cook-Torrance Illumination Model (10)

Shadow

- Some of the light incoming from a direction $\hat{\mathbf{l}}$ on a facet *da* is blocked by the opposite facet of the groove
- This leaves the lower part of the micro-facet in shadow

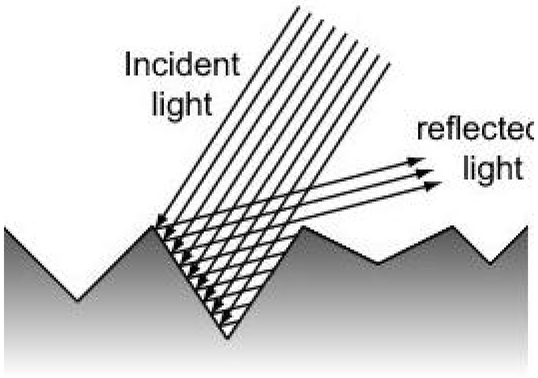
$$G_{shadow} = \frac{2(\hat{\mathbf{n}}\cdot\hat{\mathbf{h}})(\hat{\mathbf{n}}\cdot\hat{\mathbf{l}})}{\hat{\mathbf{l}}\cdot\hat{\mathbf{h}}} = \frac{2(\hat{\mathbf{n}}\cdot\hat{\mathbf{h}})(\hat{\mathbf{n}}\cdot\hat{\mathbf{l}})}{\hat{\mathbf{v}}\cdot\hat{\mathbf{h}}}$$




masking (shadow)

Combining, Geometric attenuation factor G

$$G = \min\{1, \frac{2(\hat{\mathbf{n}}\cdot\hat{\mathbf{h}})(\hat{\mathbf{n}}\cdot\hat{\mathbf{v}})}{\hat{\mathbf{v}}\cdot\hat{\mathbf{h}}}, \frac{2(\hat{\mathbf{n}}\cdot\hat{\mathbf{h}})(\hat{\mathbf{n}}\cdot\hat{\mathbf{l}})}{\hat{\mathbf{v}}\cdot\hat{\mathbf{h}}}\}$$




interception

# The Cook-Torrance Illumination Model (11)

- Spectral composition for Cook-Torrance model:

1. Diffuse part of BRDF
   - Is constant
   - Equal to the reflectance at normal incidence
2. Specular part of BRDF
   - Is associated with the angle of incidence
   - It leads to a color shift when the direction of incidence and reflection are at grazing angles

# The Cook-Torrance Illumination Model (12)

## The Fresnel term *F*

- Describes how a single micro-facet reflects light
- Implements the dependence on:
  - *n*: the relative index of refraction of the material
  - *k*: the extinction coefficient In the Cook-Torrance model
- For *k = 0* and unpolarized light, the Fresnel equation is:

$$F = \frac{1}{2}\frac{(g-c)^2}{(g+c)^2}\left(1+\frac{[c(g+c)-1]^2}{[c(g-c)+1]^2}\right)$$

where

$$c = \hat{\mathbf{v}} \cdot \hat{\mathbf{h}}$$

$$g = \sqrt{n^2 + c^2 - 1}$$

# The Cook-Torrance Illumination Model (13)

## The Fresnel term *F*

- $F \rightarrow 1$
  - The angle between $\hat{\mathbf{v}}$ and $\hat{\mathbf{h}}$ tends to $\pi / 2$
  - When we look at the direction of the light source from a very low position with respect to the surface (*grazing angle*)
  - Is independent of the *n* and *k* values
  - At a grazing angle, the spectral composition of the reflected light is the same as that of the light source
- $F \neq 1$ for other angles
- $k = 0$ is also true for non-metals
- The Fresnel equation produces a good approximation for metals

# The Cook-Torrance Illumination Model (14)

- The specular part of the BRDF:
  - Gathering *D*, *G*, *F* in a single equation:

$$f_s = \frac{1}{\pi} \frac{DGF}{(\hat{\mathbf{n}} \cdot \hat{\mathbf{l}})(\hat{\mathbf{n}} \cdot \hat{\mathbf{v}})}$$

  - $(\hat{\mathbf{n}} \cdot \hat{\mathbf{l}})(\hat{\mathbf{n}} \cdot \hat{\mathbf{v}})$ maximizes the specular highlight when viewing the light source from a grazing angle

<u>Approximation of Cook and Torrance:</u>

- Since the calculation of the Fresnel term is expensive

1. Measure/estimate the reflected color at normal incidence $F_0$ via the Fresnel equation

# The Cook-Torrance Illumination Model (15)

2. At grazing angle ($F_{\pi/2} = 1$) for all wavelengths →
   - (R, G, B) of the reflected light = (R,G,B) of the incident light
   - The reflected specular color component at an angle $\theta = \widehat{\hat{\mathbf{v}}\hat{\mathbf{h}}}$ may be interpolated:

$$c_i = c_{i,0} + (c_{i,\frac{\pi}{2}} - c_{i,0}) \frac{\max(0, F_\theta(\lambda) - F_0(\lambda))}{F_{\frac{\pi}{2}} - F_0(\lambda)}$$

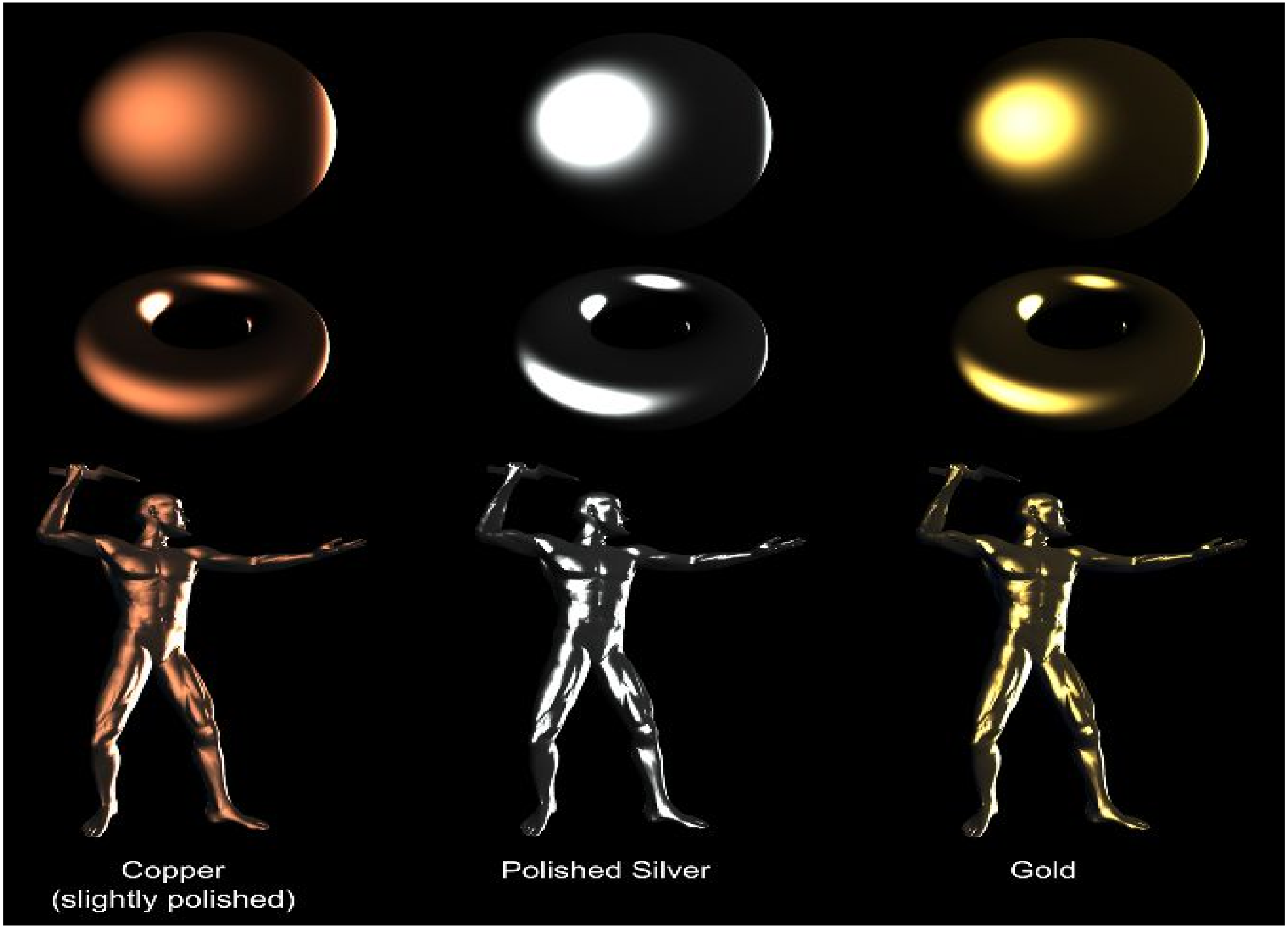
where

- $c_i$ : the color components ($i$=R,G,B ) of the resulting color
- $c_{i,\pi/2}$: the color components of the material color at normal incidence
- $c_{i,0}$ : the color components of the incident light color

3. The final color $c_i$ is:

$$c_i = \frac{1}{\pi}[c_{i,0} + (c_{i,\frac{\pi}{2}} - c_{i,0}) \frac{\max(0, F_\theta(\lambda) - F_0(\lambda))}{F_{\frac{\pi}{2}} - F_0(\lambda)}] \frac{DGF_\theta(\lambda)}{(\hat{\mathbf{n}}\cdot\hat{\mathbf{l}})(\hat{\mathbf{n}}\cdot\hat{\mathbf{v}})}$$

# The Cook-Torrance Illumination Model (16)



# The Oren-Nayar Illumination Model

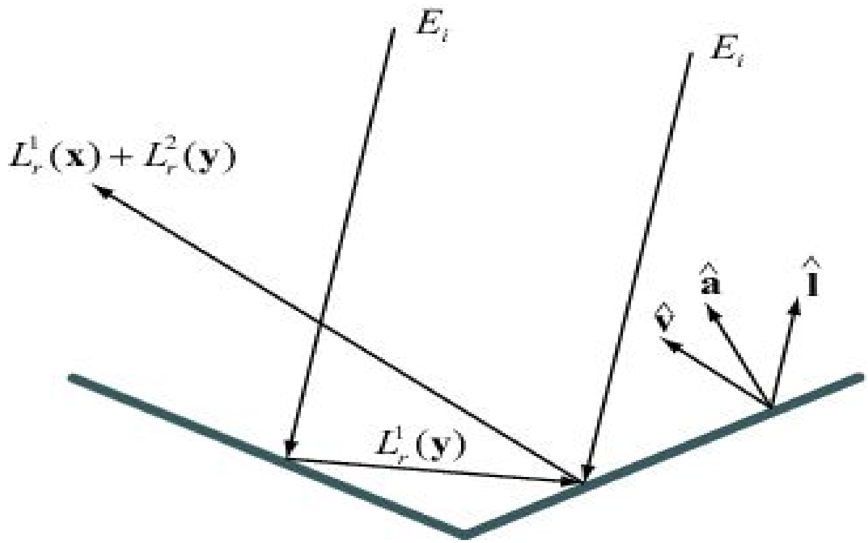
- So far the **diffuse** component of illumination was based on the Lambertian principle, i.e. equal brightness from all view directions
  - Works well for smooth surfaces
  - **Rough** surfaces are not in general Lambertian
  - E.g.  Full Moon, clay, cement and sand
- A rough surface exhibits phenomena such as
  - Light masking and shadows
  - Secondary reflections of light on the walls of the irregular microscopic structures
  - => Brightness of the reflected light increases as the viewing direction approaches the light direction
- Oren and Nayar model:
  - Incorporates the above factors to predict the diffuse behavior of rough materials
  - Adopts the micro-facet model of Torrance-Sparrow
  - A rough surface consists of long V-shaped grooves

# The Oren-Nayar Illumination Model (2)

- The facets are Lambertian surfaces (not perfect mirrors)
  - The reflected light in direction $(\theta_r, \varphi_r)$ from incident direction $\hat{\mathbf{l}}$ is computed as:

i. The 1st order reflected radiance $L_r^1(\theta_r, \phi_r, \theta_i, \phi_i)$
   - The light directly reflected in direction $\hat{\mathbf{v}}$ from a micro-facet

ii. The 2nd reflected radiance $L_r^2(\theta_r, \phi_r, \theta_i, \phi_i)$
   - The light reflected in the same direction after having bounced off the opposite facet of the groove



# The Oren-Nayar Illumination Model (3)

- Projected Radiance the contribution of a facet to the total radiance of patch *dA*

$$L_{rp}(\theta_a, \phi_a) = \frac{d\Phi_r(\theta_a, \phi_a)}{(da\cos\theta_a)\cos\theta_r d\vec{\omega}_r} \qquad (12.53)$$

where $\theta_a$ : the slope of the facet wrt surface tangent plane

- From the relation between *L*, *E* and *Φ*:

$$\left.\begin{aligned} dE_r(\theta_r, \phi_r) = L_r(\theta_r, \phi_r)\cos\theta_r d\vec{\omega}_r = L_r(\theta_r, \phi_r)(\hat{\mathbf{a}}\cdot\hat{\mathbf{v}})d\vec{\omega}_r \\ d\Phi_r(\theta_r, \phi_r) = dE_r(\theta_r, \phi_r)da \end{aligned}\right\} \Leftrightarrow$$

$$d\Phi_r(\theta_r, \phi_r) = L_r(\theta_r, \phi_r)(\hat{\mathbf{a}}\cdot\hat{\mathbf{v}})d\vec{\omega}_r da$$

- Substituting radiant flux in (12.53), $L_{rp}(\theta_a, \phi_a)$ becomes

$$L_{rp}(\theta_a, \phi_a) = \frac{L_r(\theta_r, \phi_r)(\hat{\mathbf{a}}\cdot\hat{\mathbf{v}})d\vec{\omega}_r da}{(da\cos\theta_a)\cos\theta_r d\vec{\omega}_r} = \frac{L_r(\theta_r, \phi_r)(\hat{\mathbf{a}}\cdot\hat{\mathbf{v}})}{(\hat{\mathbf{a}}\cdot\hat{\mathbf{n}})(\hat{\mathbf{v}}\cdot\hat{\mathbf{n}})} \qquad (12.55)$$

# The Oren-Nayar Illumination Model (4)

- Micro-facets are Lambertian → BRDF is constant & equal to $1/\pi$
- From the definition of BRDF:

$$L_r(\theta_r,\phi_r) = \rho f_d E_i(\theta_i,\phi_i) = \rho f_d E_0 \cos\theta_i = \rho f_d E_0 (\hat{\mathbf{l}}\cdot\hat{\mathbf{a}}) = \frac{\rho}{\pi} E_0 (\hat{\mathbf{l}}\cdot\hat{\mathbf{a}})$$

where $\rho$ = the surface albedo

$E_0$ = the irradiance from the source at normal incidence

- Replacing the radiance in (12.55):

$$L_{rp}(\theta_a,\phi_a) = \frac{\rho}{\pi} E_0 \frac{(\hat{\mathbf{l}}\cdot\hat{\mathbf{a}})(\hat{\mathbf{a}}\cdot\hat{\mathbf{v}})}{(\hat{\mathbf{a}}\cdot\hat{\mathbf{n}})(\hat{\mathbf{v}}\cdot\hat{\mathbf{n}})}$$

# The Oren-Nayar Illumination Model (5)

- The contribution of all facets facing in the direction of $\hat{\mathbf{a}}$:

$$L_r^1(\theta_r,\phi_r,\theta_i,\phi_i) = \int_{\theta_a=0}^{\pi/2}\int_{\phi_a=0}^{2\pi} P(\theta_a,\phi_a)L_{rp}^1(\theta_a,\phi_a)\sin\theta_a d\phi_a d\theta_a \quad (12.58)$$

- Geometric factor is a generalization of the Cook-Torrance factor $G$
  - works for any facet normal $\hat{\mathbf{a}}$
  - not necessarily the halfway vector $\hat{\mathbf{h}}$ between the viewing and the incident direction

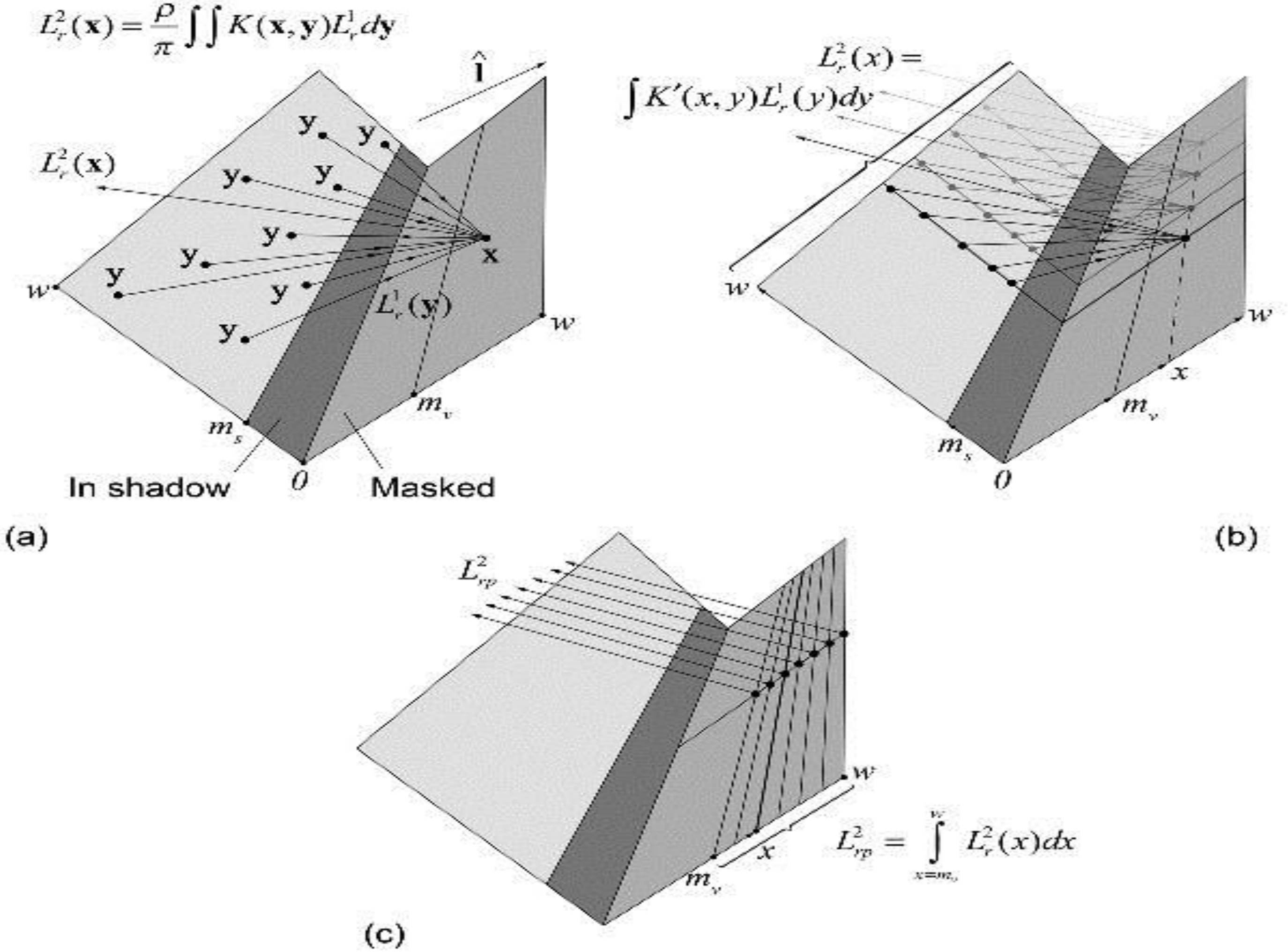
$$GAF = \min\left\{1, \max\left\{0, \frac{2(\hat{\mathbf{l}}\cdot\hat{\mathbf{n}})(\hat{\mathbf{a}}\cdot\hat{\mathbf{n}})}{\hat{\mathbf{l}}\cdot\hat{\mathbf{a}}}, \frac{2(\hat{\mathbf{v}}\cdot\hat{\mathbf{n}})(\hat{\mathbf{a}}\cdot\hat{\mathbf{n}})}{\hat{\mathbf{v}}\cdot\hat{\mathbf{a}}}\right\}\right\}$$

- Taking also into account the blocked incident and reflected light, (12.58) becomes:

$$L_r^1(\theta_r,\phi_r,\theta_i,\phi_i) = \int_{\theta_a=0}^{\pi/2}\int_{\phi_a=0}^{2\pi} P(\theta_a,\phi_a)L_{rp}^1(\theta_a,\phi_a)GAF\sin\theta_a d\phi_a d\theta_a$$

# The Oren-Nayar Illumination Model (6)

- Radiance from second-order reflections :



# The Oren-Nayar Illumination Model (7)

- The overall radiance leaving patch *dA* in the direction $\hat{\mathbf{v}}$ $(\theta_r, \phi_r)$ :

$$L_r(\theta_r, \phi_r, \theta_i, \phi_i) = L_r^1(\theta_r, \phi_r, \theta_i, \phi_i) + L_r^2(\theta_r, \phi_r, \theta_i, \phi_i) \qquad (12.61)$$

- Simplification of the original model:

$$L_r^1(\theta_r, \phi_r, \theta_i, \phi_i) = \frac{\rho}{\pi} E_0 \cos\theta_i [C_1 + \cos(\phi_r - \phi_i) C_2 \tan\beta + (1 - |\cos(\phi_r - \phi_i)|) C_3 \tan(\frac{\alpha + \beta}{2})]$$

$$L_r^2(\theta_r, \phi_r, \theta_i, \phi_i) = 0.17 \frac{\rho}{\pi} E_0 \frac{\sigma^2}{\sigma^2 + 0.13} \cos\theta_i [1 - (\frac{2\beta}{\pi})^2 \cos(\phi_r - \phi_i)]$$

where

$$C_1 = 1 - 0.5 \frac{\sigma^2}{\sigma^2 + 0.3}$$

$$C_2 = \begin{cases} 0.45 \dfrac{\sigma^2}{\sigma^2 + 0.09} \sin\alpha, & \cos(\phi_r - \phi_i) \geq 0 \\ 0.45 \dfrac{\sigma^2}{\sigma^2 + 0.09} (\sin\alpha - (\dfrac{2\beta}{\pi})^3), & \text{otherwise} \end{cases}$$

$$C_3 = 0.125 \frac{\sigma^2}{\sigma^2 + 0.09} (\frac{4\alpha\beta}{\pi^2})^2$$

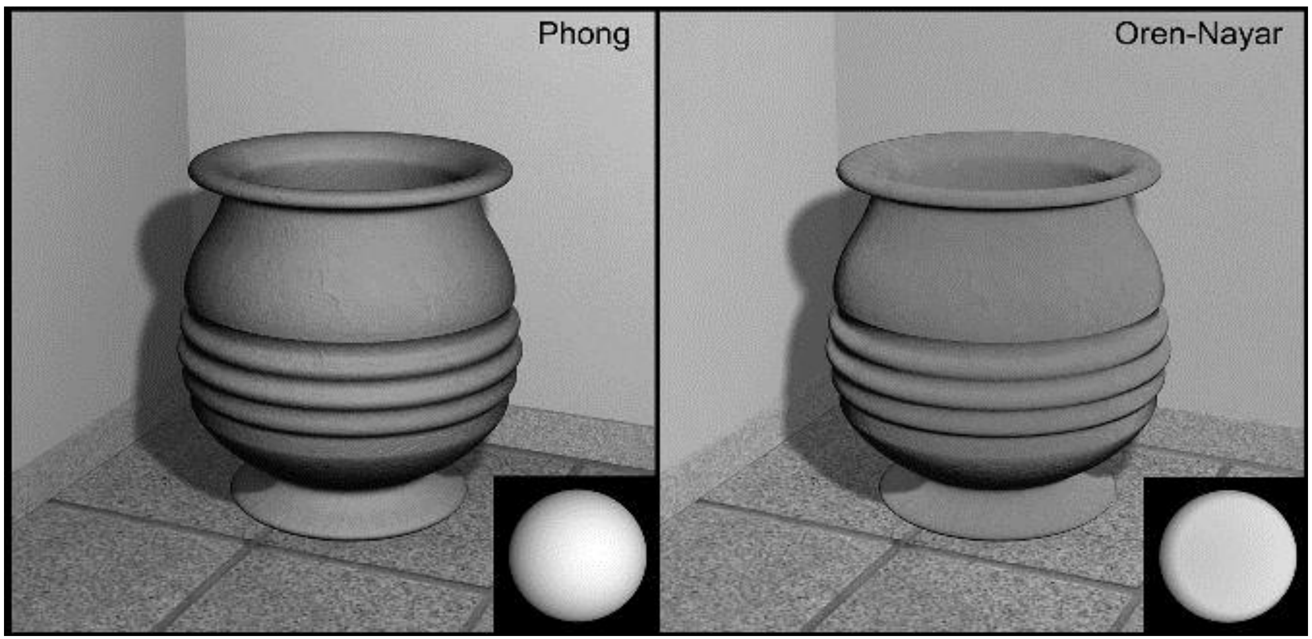
$$\alpha = \max(\theta_r, \theta_i), \beta = \min(\theta_r, \theta_i)$$

# The Oren-Nayar Illumination Model (8)

- The BRDF is acquired by applying the BRDF definition to (12.61):

$$f_{Oren-Nayar} = \frac{L(\theta_r, \phi_r, \theta_i, \phi_i)}{E_i} = \frac{L(\theta_r, \phi_r, \theta_i, \phi_i)}{E_0 \cos\theta_i} =$$

$$\frac{L_r^1(\theta_r, \phi_r, \theta_i, \phi_i) + L_r^2(\theta_r, \phi_r, \theta_i, \phi_i)}{E_0 \cos\theta_i}$$



# The Strauss Illumination Model

- Illumination models based on geometrical optics (Blinn, Cook-Torrance and Oren-Nayar)
  - Produce very realistic shading
  - But, use actual physical parameters found in material science which are generally unintuitive for users (e.g. artists)
- The Phong model
  - Cannot effectively capture the appearance of metallic surfaces
  - The specular exponent is specified as an unbounded positive number →
    - Cannot easily produce a balanced shininess between a dull surface & a fully reflective one by adjusting its value between two limits
  - The shininess adjustment is complex
    - Two seemingly independent parameters (the exponent and the specular coefficient) control the same material attribute

# The Strauss Illumination Model (2)

- Strauss Illumination Model:
  - Borrows many lighting calculations from Phong
  - Incorporates features like
    - metallic appearance
    - off-specular reflections
    - unified shininess control

    through intuitive normalized parameters
  - Empirical model (targeting animators and 3D modelers)
- Normalized parameters that control surface appearance:

i. The material color $\mathbf{c} = (r, g, b)$ : represents the albedo of the surface

ii. The smoothness $s$ : – ranges from 0 (dull surface) to 1 (perfect mirror)

– controls : the specular/diffuse contribution ratio

the size of the highlight

i. The metalness $m$ : – is ranges from 0 to 1 (metallic surface)

– affects the color of the specularly reflected light

# The Strauss Illumination Model (3)

- The intensity of the reflected light per color channel $c_r$ :

$$c_r = c_i(Q_d + Q_s + Q_a)$$

where

- $c_i$ : the corresponding incident light component
- $Q_d$, $Q_s$, $Q_a$ : the diffuse, specular and ambient components of the Strauss model

- The amount of diffuse illumination $Q_d$ :
  - Depends on the shininess of the surface *s*
    - The more shiny the surface, the less it behaves as a Lambertian reflector
  - Decreases with the increase of the metalness *m*
  - Depends on the angle of incidence

# The Strauss Illumination Model (4)

- The Strauss diffuse and ambient components are:

$$Q_d = (\hat{\mathbf{n}} \cdot \hat{\mathbf{l}}) r_d d c \qquad Q_a = r_d c$$

$$r_d = (1 - s^3)(1 - t)$$

$$d = (1 - ms)$$

where :

- *t:* the transparency of the surface (0 (fully opaque) → 1 )
- *c:* one of the red, green or blue components of the surface color
- ($1$-$s^3$) is experimentally chosen to account for a linear perceptual transition from a dull surface to a perfect mirror, with a corresponding linear change in the *s* parameter

# The Strauss Illumination Model (5)

- The specular component $Q_s$ is:

$$Q_s = r_s \; c_s$$

  where
  - $r_s$: the <u>specular reflectivity</u>, defines the shape of the highlight
  - $c_s$: the <u>specular color</u> is interpolated for metallic surfaces between the surface color and the light color
- The specular reflectivity $r_s$:
  - Depends on the angle between the mirror reflection direction & the view vector
  - Is raised to a power to tighten the highlight

$$r_s = (\hat{\mathbf{r}} \cdot \hat{\mathbf{v}})^h r_j$$

$$h = \frac{3}{1-s}$$

# The Strauss Illumination Model (6)

- The adjusted reflectivity $r_j$ encapsulates
  - the specular attenuation due to the Fresnel term
  - the geometric attenuation factor
- $r_j$ depends on the reflectivity of the surface at normal incidence

$r_n = 1 - t - r_d$ so: $r_j = \min[1, \; r_n + (r_n + k_j)\, F(\theta_i)\, G(\theta_i)\, G(\theta_r)]$

where

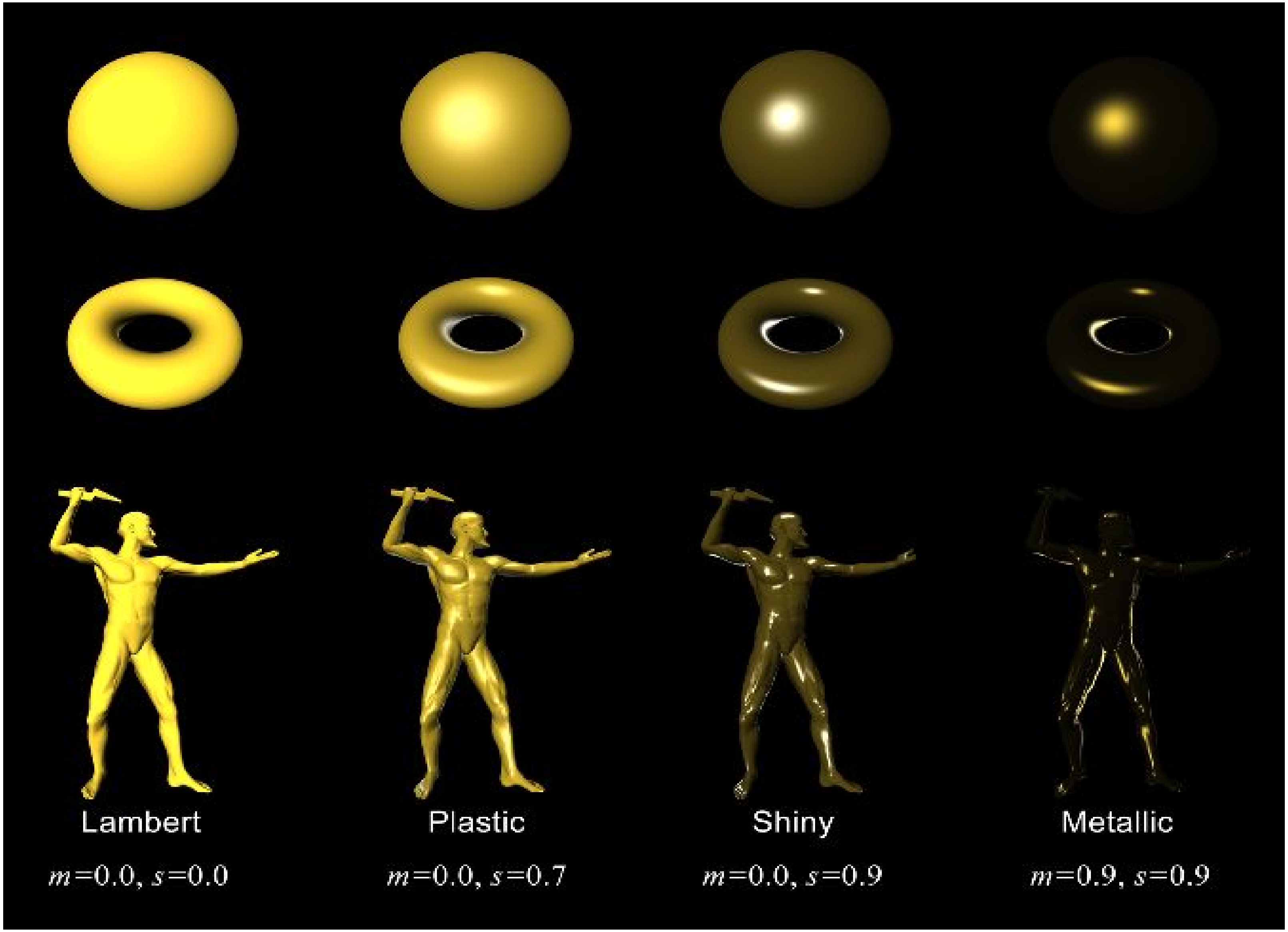
  - $F(x)$ : an empirical Fresnel-like function
  - $G(x)$ : a geometric attenuation function

$$F(x) = [\frac{1}{(x-k_f)^2} - \frac{1}{k_f^2}] / [\frac{1}{(1-k_f)^2} - \frac{1}{k_f^2}]$$

$$G(x) = [\frac{1}{(1-k_g)^2} - \frac{1}{(x-k_g)^2}] / [\frac{1}{(1-k_g)^2} - \frac{1}{k_g^2}]$$

  - The constants $k_j$ , $k_f$, $k_g$ are experimentally chosen
  - Strauss suggests the values $k_j = 0.1$ , $k_f = 1.12$ , $k_g = 1.01$

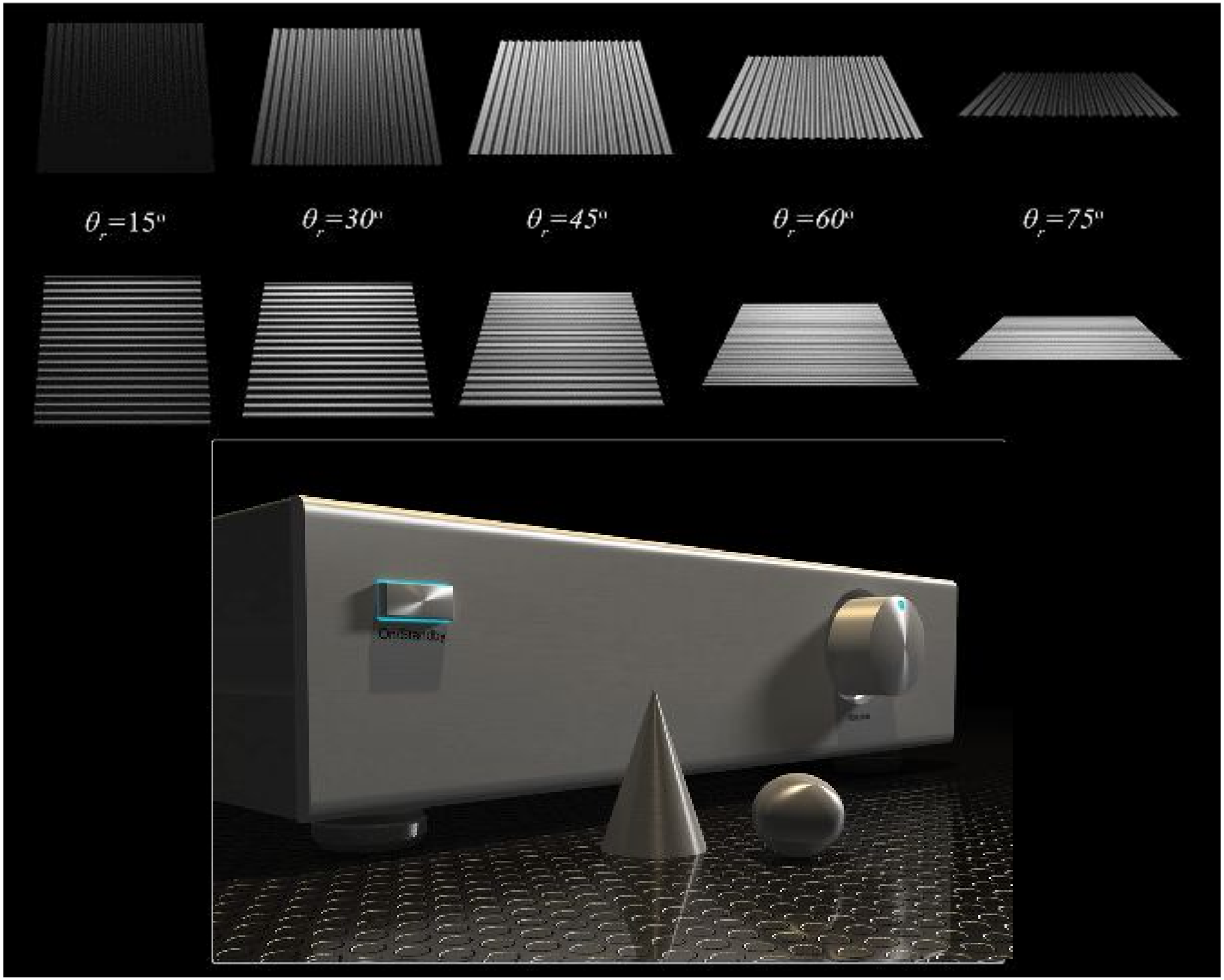
# The Strauss Illumination Model (7)



# Anisotropic Reflectance

- All previous lighting models possessed an isotropic BRDF
  - Reflected light did not depend on the azimuth angle of incidence
- Many real materials and treated surfaces exhibit a distinctive directional bias
- Anisotropic specular reflection is caused by the microscopic geometric structures of the surface
- Most anisotropic reflective materials possess a characteristic grain or a set of very small grooves which are roughly oriented in a specific direction
- The grooves appear parallel within a magnified surface area
- Good examples of anisotropic reflectors
  - brushed metals (e.g. brushed aluminum)
  - varnished wood
  - vinyl music records

# Anisotropic Reflectance (2)

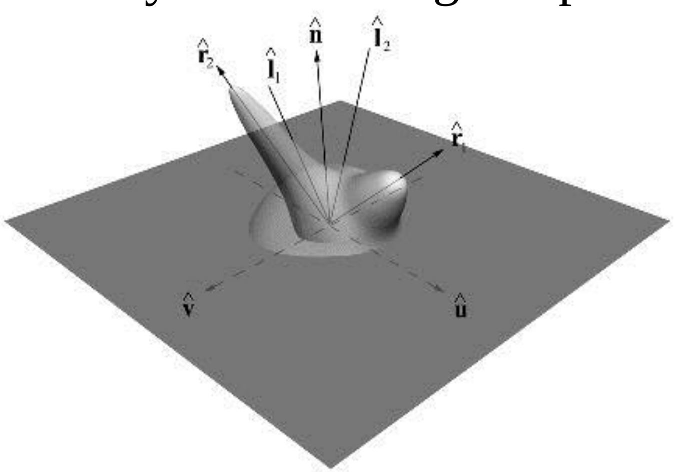


# Anisotropic Reflectance (3)

- Model the surface according to the micro-facet approach
  - Assume that the surface grain lays on a longitude direction $\varphi_g$
  - The distribution of the facets *da* with respect to their normal direction $\hat{\mathbf{a}} = (\theta_a, \phi_a)$ is clearly directional
    - $\theta_a = 0$ for $\varphi_a = \varphi_g,\ \varphi_g + \pi$
    - $\theta_a$ ranges from $-\theta_s$ to $\theta_s$ for $\varphi_a = \varphi_g \pm \pi/2$
    - $\theta_s$ : the maximum slope
- Observing the surface from a macroscopic level
  - Incident light coming from $(\theta_i, \varphi_i)$
  - In the extreme case where all grooves are aligned with $\varphi_g$
    - The surface becomes a perfect mirror when $\varphi_i = \varphi_g,\ \varphi_g + \pi$
    - The surface has a wider spread of the highlight as $\varphi_i \rightarrow \varphi_g \pm \pi/2$ (maximum anisotropy)

# Anisotropic Reflectance (4)

- Several models to deal with anisotropy
  - Kajiya
  - Poulin-Fournier
- For arbitrary geometry it is difficult to represent the direction of maximum (and minimum) reflectance on the surface, which is dependent on the azimuth angle $\varphi_g$
  - This direction is a local attribute of the model
  - Cannot in general be expressed relative to the object or world reference frame
  - Most implementations rely on local tangent space (e.g. using texture mapping)

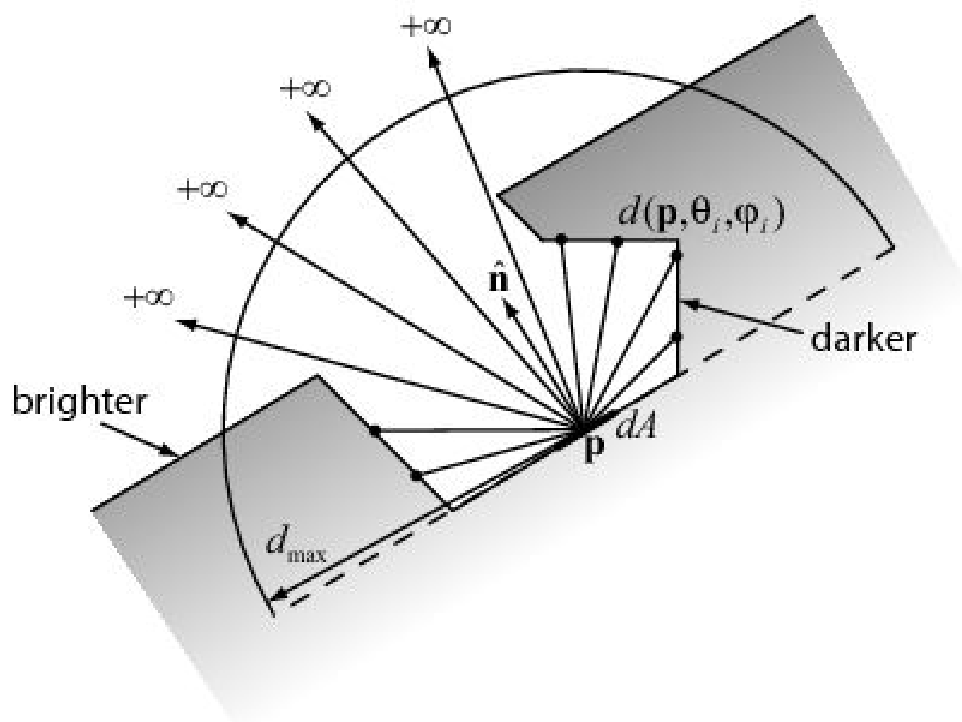


# Ambient Occlusion

- Local illumination models regard ambient illumination as constant
  - Ambient term is the irradiance that reaches a surface as the summed contribution of the emitted or reflected light from the environment and
  - Accounts for the exchange of energy between a patch *dA* and all other possibly contributing patches in a scene
- A constant ambient illumination is clearly a very rough approximation, even for simple scenes
- Exchange of energy in a closed environment is simulated via a *global illumination* method (Chapter 16)
- One aspect of the global energy exchange that affects the ambient term, **the darkening effect in obscured parts of a scene**, can be simulated in a more efficient manner
- <u>**Ambient occlusion:**</u> Assuming a uniform (ambient) distant environment irradiance from every direction, estimates the portion of it that finally reaches a small patch *dA*

# Ambient Occlusion (2)

- Equivalent to calculating the visibility of a patch due to the presence of the rest of the geometry
  - i.e. portion of the solid angle around the patch, from where *dA* is visible
- Inversely, the *obscurance* of a patch *dA* is the portion of the hemispherical solid angle around the patch that is blocked by other geometry



# Ambient Occlusion (3)

- The higher the obscurance, the darker the patch
  - *dA* is blocked at many incident directions by other patches → less light from the environment can hit the surface
- *Obscurance* $w(\mathbf{p})$ reflects the “openness” of a patch *dA* centered at a point **p**
  - A purely geometric property
  - Does not depend on any particular lighting conditions or viewing direction
  - Is usually pre-calculated and stored as vertex data
- $w(\mathbf{p})$ can be multiplied with a constant ambient term & provides a convincing estimate of the incident light from the environment

# Ambient Occlusion (4)

- Note that ambient occlusion is not a physical simulation model and does not provide an accurate global illumination calculation:
  - Misses the high-order bounces of energy that eventually hit the surface
  - Regards irradiance to be constant in all incident directions
- Assumptions:
  - No specific light sources in the environment
  - The (uniform) incident ambient illumination can be modeled as a perfectly diffuse light that radiates from all directions towards *dA*
  - Light is not emitted from some infinite medium far from the scene itself, but the geometry is immersed in a radiating, non-absorbing, gaseous medium.
- Let $d(\mathbf{p},\theta_i,\varphi_i)$: distance between **p** and the closest surface point in direction $(\theta_i,\varphi_i)$:

$$d(\mathbf{p},\theta_i,\phi_i)=\begin{cases}|\mathbf{c}-\mathbf{p}|, & \mathbf{c}:\text{first intersection point in direction }(\theta_i,\phi_i)\\ +\infty, & \text{no intersections in direction }(\theta_i,\phi_i)\end{cases} \quad (12.71)$$

# Ambient Occlusion (5)

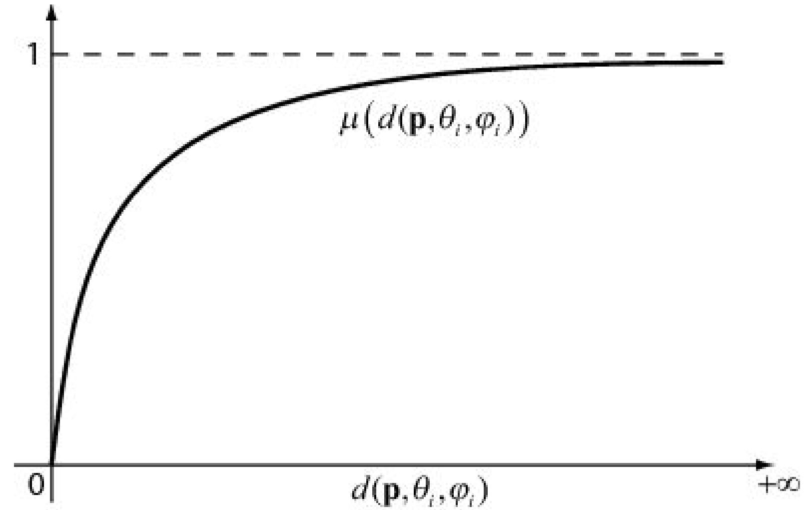
- The farther from **p** an intersection point is, the more light reaches the surface of the patch *dA*
- If the hemispherical solid angle above the patch is completely open up to a distance $d_{max}$, the obscurance $w(\mathbf{p})$ is set to 1
- Obscurance becomes 0 only in degenerate cases or where 2 surfaces firmly touch each other
- $d_{max}$ is the maximum distance at which the contribution of the surrounding geometry is non-negligible
- Intensity of reflected light from patch *dA* centered at **p,** due to ambient illumination coming from hemisphere *Ω* above *dA* can be approximated as:

$$I_a(\mathbf{p}) = k_a I_a w(\mathbf{p})$$

$$w(\mathbf{p}) = \frac{1}{\pi}\int_{\Omega} \mu(d(\mathbf{p},\theta_i,\phi_i))\cos\theta_i d\vec{\omega} \qquad (12.72)$$

# Ambient Occlusion (6)

- Function $\mu(x)$ maps the distance $x = d(\mathbf{p},\theta_i,\varphi_i)$ to a normalized obscurance factor
  - It represents the energy emitted by the gaseous medium in the line of sight from **p** to the closest surface in the direction $(\theta_i,\varphi_i)$
- $\mu(x)$ must be:
  - Monotonically increasing and smooth
  - 0 for zero distance and 1 at infinity with a decreasing slope




$$\mu(x) = \begin{cases} 0, & x = 0 \\ 1, & x = +\infty \end{cases}, \quad \frac{d\mu(x)}{dx} = \begin{cases} 0, & x = +\infty \\ > 0, & \text{otherwise} \end{cases}, \quad \frac{d^2\mu(x)}{dx^2} < 0$$

(12.73)

# Ambient Occlusion (7)

- Common family of functions that conforms to the requirements:

$$\mu(x) = 1 - e^{-\tau x} \qquad (12.74)$$

- Parameter $\tau$ regulates the spread of the shadowed area
- Since $d_{max}$ defines a range of distance from **p** beyond which no patch is taken into account, $\mu(x)$ has to be modified to normalize this input range

# Ambient Occlusion (8)

- Let us now introduce $N_L$ light sources with intensities:

$$I_L(j), j = 1...N_L,$$

  at distance $d_j$ from the patch $dA$ & direction of incidence $\hat{\mathbf{l}}_j$
- Assuming Lambertian surfaces, these light sources contribute to the illumination of the patch both in the ambient & the diffuse term
- The resulting illumination for a point **p** of the patch is:

$$I(\mathbf{p}) = [k_a I_a + k_d I_d(\mathbf{p})]w(\mathbf{p}) + I_d(\mathbf{p})$$

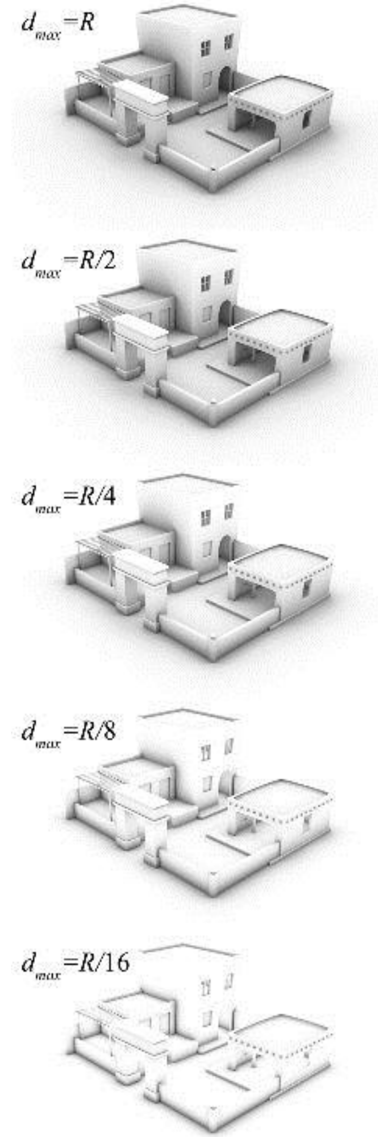
$$I_d(\mathbf{p}) = \sum_{j=1}^{N_L} \delta(\mathbf{p}, j) \frac{I_L(j)}{d_j^2} (\hat{\mathbf{l}}_j \cdot \hat{\mathbf{n}}) \qquad (12.75)$$

  where

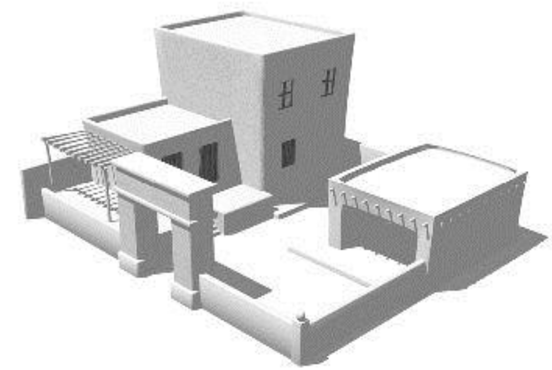
  δ (**p,** j) is a visibility factor that:
  - becomes 1 if the *j*th light source is visible from the patch
  - becomes 0 if the patch is in shadow for the specific light source

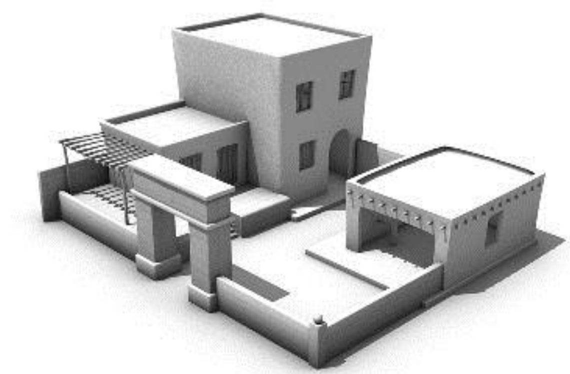
# Ambient Occlusion (9)

- EXAMPLE
- Obscurance estimation for various values of the distance limit $d_{max}$ (left). $R$ is the scene radius.
- Scene rendered with constant ambient illumination (top right) & with obscurance-weighted ambient-diffuse illumination (bottom right).



Obscurance


Scene rendered with constant ambient lighting


Scene rendered with ambient occlusion ($d_{max}=R/8$)

# Shader Source Code - Cook-Torrance

```
//##### Cook-Torrance Model #####//
//##### Vertex program ##########//
varying vec3 N,P;
void main()
{
  gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
  N = normalize ( gl_NormalMatrix * gl_Normal );
  P = vec3 (gl_Position) / gl_Position.w;
}
```

# Shader Source Code - Cook-Torrance (2)

```
//###### Cook-Torrance Model #####//
//###### Fragment program ########//
varying vec3 N, P; const float pi = 3.1415936;
const float e = 2.718282; const int numLights = 2;
uniform float Ka, Kd, Ks,       // ambient, diffuse, specular coefs.
                          m;    // RMS micro-facet slope
uniform vec3 n; // n(630nm) n(530nm) n(465nm) at normal incidence
uniform vec3 color;             // The material color
// The Beckmann distribution function
float Beckmann ( in float a ) {
      float tana = tan(a)/m; float cosa = cos(a); cosa *= cosa;
      return pow ( e, -tana*tana ) / (m*m*cosa*cosa); }
// The Fresnel term
float Fresnel( in float n, in float c ) {
      float g, gc, F; g = clamp ( n*n+c*c-1, 0.000001, 1.0); g = sqrt(g);
      gc = g+c; F = (g-c)*(g-c)/(2*gc*gc);
      return F * ( 1 + (c*gc-1)*(c*gc-1)/( (c*gc+1)*(c*gc+1) ) );
}
```

# Shader Source Code - Cook-Torrance (3)

```
// The Cook-Torrance model for the specular reflectance
void CookTorrance (   in vec3 L,      // light direction
                      in vec3 V,      // view direction
                      in vec3 H,      // half-way vector
                      in float a,     // angle ( N, H )
                      in vec3 Il,     // incident illumination
                      in vec3 C0,     // material color
                      out vec3 Is_I   // resulting specular color
                  )
{
float NL, NV, VH, NH;          // dot products
float D, G;                    // D and G scalar terms
vec3 F0, F;                    // The tri-chromatic Fresnel terms
                               // for normal & arbitrary incidence
```

# Shader Source Code - Cook-Torrance (4)

```
NL = dot(N,L); NV = dot(N,V); VH = dot(V,H); NH = dot(N,H);
D = Beckmann(a); G = min ( 1, min( 2*NH*NV/VH, 2*NH*NL/VH ) );
F0.r = Fresnel(n.r,1); F0.g = Fresnel(n.g,1); F0.b = Fresnel(n.b,1);
F.r = Fresnel(n.r,VH); F.g = Fresnel(n.g,VH); F.b = Fresnel(n.b,VH);
Is_i = (C0+(Il-C0)*(max(F-F0,0)/(1.0-F0)))*((F.r+F.g+F.b)/3) *D*G/
       pi*NL*NV);
}

void main() {
     vec3 Pl;                          // Light position
     vec3 L, H, V;                     // directions (unit vectors)
     vec3 Ia, Id, Is, Is_i, Il;        // Intensity values
     int i;
     float NL, a;
     V = vec3 (0.0, 0.0, 1.0);         // View direction
```

# Shader Source Code - Cook-Torrance (5)

```
Ia = vec3 (0.0, 0.0, 0.0);        // Init. amb/dif/spec values
Id = vec3 (0.0, 0.0, 0.0);  Is = vec3 (0.0, 0.0, 0.0);
// Add the contribution of all light sources
for ( i = 0; i< numLights; i++ ) {
   Pl = vec3 (gl_LightSource[i].position); L = normalize( Pl - P );
   H = normalize( L + V ); NL = dot (N,L);
   // Diffuse
Id += gl_LightSource[i].diffuse * NL; a = acos( dot(N,H) );
   Il = vec3 (gl_LightSource[i].diffuse);
   CookTorrance ( L, V, H, a, Il, color, Is_i );
   // Specular
   Is += Is_i; }
// Ambient
Ia = Ka * gl_FrontLightModelProduct.sceneColor;
gl_FragColor = vec4(Ia,1)+Kd *vec4(Id,1)*vec4(color,1)+Ks*vec4(Is,1);
}
```

# Shader Source Code - Strauss

```
//##### Strauss Model #####//
//##### Vertex program ####//
varying vec3 N; varying vec3 P;
void main() {
   gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
   N = normalize ( gl_NormalMatrix * gl_Normal );
   P = vec3 (gl_Position) / gl_Position.w;
}
```

# Shader Source Code - Strauss (2)

```
//##### Strauss Model #####//
//##### Fragment program ##//
varying vec3 N; varying vec3 P; const float pi = 3.1415936;
const int numLights = 2; uniform float m;       // metalness uniform
float s; // shininess uniform float t; //transparency
uniform vec3 C; // surface color
//----- Fresnel term -----
float F ( in float x ) {
   const float kf = 1.12f; const float kf2 = kf*kf;
   const float denom = ( 1.0/((1.0-kf)*(1.0-kf)) - 1.0/kf2 );
   return ( ( 1.0/((x-kf)*(x-kf)) - 1.0/kf2 ) / denom);
}
//----- Geometric Attenuation-----
float G ( in float x ) {
   const float kg = 1.01f; const float kg2 = kg*kg;
   const float denom = ( 1.0/((1.0-kg)*(1.0-kg)) - 1.0/kg2 );
   return ( 1.0/((1.0-kg)*(1.0-kg)) - 1.0/((x-kg)*(x-kg)) )/ denom;
}
```

# Shader Source Code - Strauss (3)

```
void main() {
   vec3 Pl, L, V, H; vec3 Qa, Qd, Qs, Ir, Cs; int I; float NL, NV, f;
   float theta_i, theta_r; float rn, rj, rd, rs, d; const float kj = 0.1;
   // Note that conventions in the original paper differ from standard
   // normalized vector definitions: L & V face towards the local point P
   // View direction
   V = -normalize(P); NV = dot(N,V); Ir = vec3 (0.0, 0.0, 0.0);
   for ( i = 0; i< numLights; i++ ) {
      Pl = vec3 (gl_LightSource[i].position); L = normalize( P - Pl );
      NL = dot(N,L); H =normalize( L-2*NL*N ); theta_i = 2*acos(abs(NL))/pi;
      theta_r = 2*acos(abs(NV))/pi; rd = (1-s*s*s)*(1-t); d = 1-m*s;
      rn = 1-t-rd; f = F((theta_i+theta_r)/2);
      rj = min (1, rn+(rn+kj)*f*G(theta_i)*G(theta_r));
      rs = pow(-dot(H,V),3/(1.0001-s))*rj; Cs = 1 + m*(1-f)*(C-1);
      Qd = clamp (-NL*d*rd*C,0,1); Qs = clamp (rs*Cs,0,1);
      Ir += gl_LightSource[i].diffuse * (Qd+Qs) +gl_LightSource[i].ambient * Qa;}
   gl_FragColor = vec4(Ir,1-t);
}
```