# Illumination I: The Phong Illumination Model

Components of Phong illumination or reflection model using RGB model:

OpenGL allows us to break this light's emitted intensity into 3 components: ambient $L_a$, diffuse $L_d$, and specular $L_s$. Each type of light component consists of 3 color components, so, for example, $L_{rd}$ denotes the intensity of the red component of diffuse illumination.

*Question:* What is the amount of light that is transmitted (either by emission or reflection) from each point in the direction of the viewer.

*Solution:* This is achieved by first associating reflectivity or material properties to all the modelled objects in the scene, and then applying a Phong reflection calculation to determine the transmitted light intensity.

*The Reflected Light Luminance/Intensity function shall be captured by:*

$$I = (I_r, I_g, I_b)$$

for each of Light's components. For example,

- ambient emission

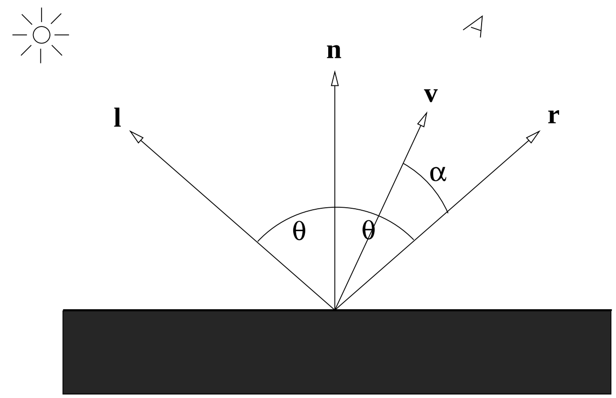
$$I_a = \begin{pmatrix} I_{ar} \\ I_{ag} \\ I_{ab} \end{pmatrix}$$

An object's material properties determines how much of a given input Light intensity is reflected. Under the Phong model, material properties are captured by reflectivity coefficient vectors $K = (k_r, k_g, k_b)$ for ambient, diffuse and specular. Thus $k_d r$ is the fraction of red diffuse light that is reflected from an object. If $k_r = 0$, then no red light is reflected.

The computation of reflected luminance/intensity function using Phong illumination, for each object and light source, shall be governed by the following four light/material interactions.

- *Emission intensity:* to model objects that glow
- *Ambient reflection:* A simple way to model indirect reflection. All surfaces in all positions and orientations are illuminated equally.
- *Diffuse reflection:* The diffuse shading produced by dull, smooth objects.
- *Specular reflection:* The bright spots appearing on smooth shiny (e.g., metallic or polished) surfaces.

# Relevant Vectors for Phong Shading



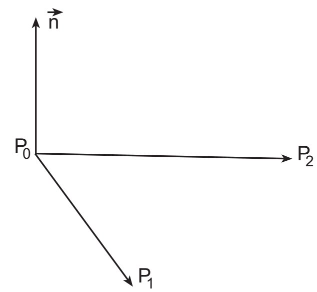

The shading of a point on a surface is a function of the relationship between the viewer, light sources, and surface. The following vectors are relevant to direct illumination. All vectors are assumed to be normalized to unit length.

- *Normal vector:* A vector $\vec{n}$ that is perpendicular to the surface and directed outwards from the surface.
- *View vector:* A vector $\vec{v}$ that points in the direction of the viewer.
- *Light vector:* A vector $\vec{l}$ that points towards the light source.
- *Reflection vector:* A vector $\vec{r}$ that indicates the direction of pure reflection of the light vector.

# Normals Computation

Given any three non-collinear points, $P_0$, $P_1$, $P_2$, on a polygon, a normal of the polygon is given through a cross product

$$\vec{n} = (P_1 - P_0) \times (P_2 - P_0).$$




Normals by cross product.

For a polygon is given by $n$ points $P_0$, $P_1$, . . . , $P_{n-1}$. If we can determine a plane equation (via least-squares fit):

$$ax + by + cz + d = 0$$

from these $n$ points, then normalizing $(a, b, c)$ is the unit normal vector $\vec{n}$ of the polygon.

# Normals for Implicitly Defined Surfaces

Given a surface defined by an *implicit representation*, i.e., defined by some equation

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

then the normal at some point is given by *gradient vector*

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

# Normals for Parametric Surfaces

Surfaces in computer graphics are most often represented parametrically. The *parametric representation* of a surface is defined by three functions of 2 variables or *parameters*:

$$
\begin{aligned}
x &= \phi_x(u, v), \\
y &= \phi_y(u, v), \\
z &= \phi_z(u, v).
\end{aligned}
$$

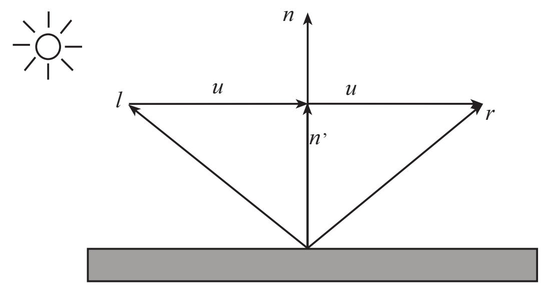
Then the normal of the surface at a point is defined as the

$$\vec{n} = \frac{\partial \phi}{\partial u} \times \frac{\partial \phi}{\partial v}$$

where

$$\frac{\partial \phi}{\partial u} = \begin{pmatrix} \partial \phi_x / \partial u \\ \partial \phi_y / \partial u \\ \partial \phi_z / \partial u \end{pmatrix} \qquad \frac{\partial \phi}{\partial v} = \begin{pmatrix} \partial \phi_x / \partial v \\ \partial \phi_y / \partial v \\ \partial \phi_z / \partial v \end{pmatrix}$$

# The Reflection Vector

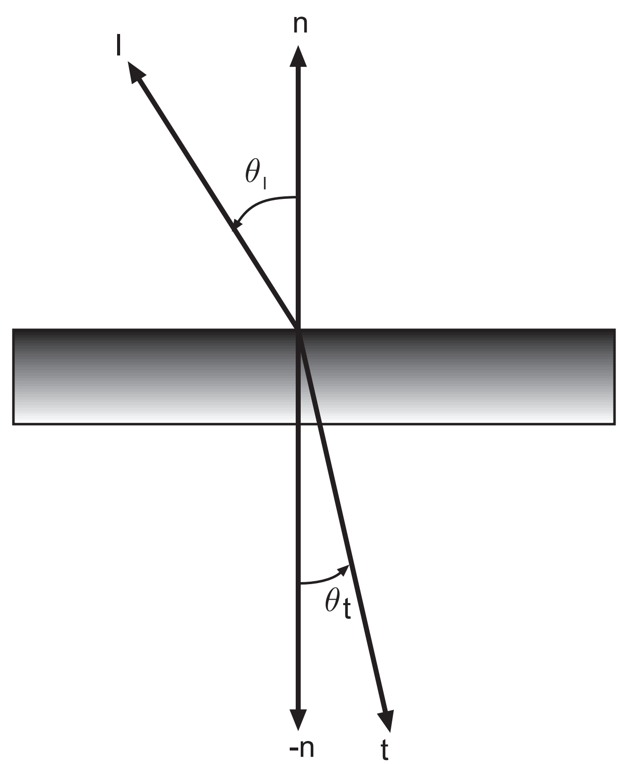



$$\vec{n}' = (\vec{n} \cdot \vec{l})\vec{n}$$

$$\vec{u} = \vec{n}' - \vec{l}$$

$$\vec{r} = \vec{l} + 2\vec{u} = \vec{l} + 2(\vec{n}' - \vec{l}) = 2(\vec{n} \cdot \vec{l})\vec{n} - \vec{l}$$

# The Refraction Vector




If $\eta_l$ and $\eta_t$ are the refractive indices of the materials on the two sides of the surface, then Snell's law states that

$$\eta_l Sin(\theta_l) = \eta_t Sin(\theta_t)$$

Using this and the fact that $\vec{l}$, $\vec{n}$, and $\vec{t}$ are assumed coplanar, we can calculate the unit transmitted light vector $\vec{t}$, as follows. let $\eta = \frac{\eta_t}{\eta_l}$, we have

$$Cos(\theta_t) = ((1 - \frac{1}{\eta^2}(1 - Cos^2(\theta_l)))^{\frac{1}{2}}$$

and

$$t = -\frac{1}{\eta}\vec{l} - (Cos(\theta_t) - \frac{1}{\eta}Cos(\theta_l))\vec{n}$$

# Ambient Light Reflection

Ambient light is simplest to deal with. Let $I_a$ denote the intensity of ambient light. For each surface, let

$$0 \leq k_a \leq 1$$

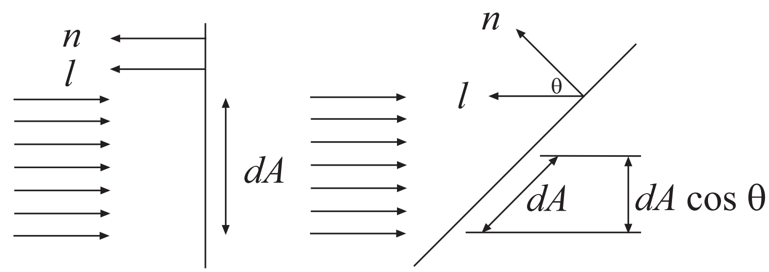
denote the surface's *coefficient of ambient reflection*, that is, the fraction of the ambient light that is reflected from the surface. The ambient component of illumination is

$$I_a = k_a L_a$$

Note that this is a vector equation (whose components are RGB).

# Diffuse Reflection

Diffuse reflection arises from the assumption that light from any direction is reflected uniformly in all direction. Such a reflector is called a pure *Lambertian reflector*.




The key parameter of surface that controls diffuse reflection is $k_d$, the surface's *coefficient of diffuse reflection*. Let $I_d$ denote the diffuse reflection component of the light source. Assume $\vec{l}$ and $\vec{n}$ are normalized, then $\cos\theta = (\vec{n} \cdot \vec{l})$. If $(\vec{n} \cdot \vec{l}) < 0$, then the point is on the dark side of the object.

The diffuse component to illumination is

$$I_d = k_d \max(0, \vec{n} \cdot \vec{l}) L_d$$

# Specular Reflection I

Most objects are not perfect Lambertian reflector. One of the most common deviation is for smooth metallic or highly polished objects. They tend to have *specular highlights* (or "shiny spots").

The parameters of surface that control specular reflection under Phong model, are $k_s$, the surface's *coefficient of specular reflection*, and $s$, *shininess*.

The formula for the specular component is

$$I_s = k_s(\vec{r} \cdot \vec{v})^s L_s$$

# Specular Reflection II

Another way of calculating specular reflection under the Phong model, is via the *halfway vector* (OpenGL).

Define $\vec{h}$ to be the *halfway vector*, the normalized vector which is the halfway of $\vec{l}$ and $\vec{v}$. Define $\vec{h} =$ Normalize $(\vec{l} + \vec{v})$.

The formula for the specular component can then be written as

$$I_s = k_s(\vec{n} \cdot \vec{h})^s L_s$$

# The Phong Model Illumination Equation

The total illumination of a point in OpenGL is computed for the supported Light sources and is calculated

$$\begin{aligned} I &= I_e + I_a + \frac{1}{a + bd + cd^2}(I_d + I_s) \\ &= I_e + k_a L_a + \frac{1}{a + bd + cd^2}(k_d \max(0, \vec{n} \cdot \vec{l}) L_d + k_s (\vec{n} \cdot \vec{h})^s L_s), \end{aligned}$$

where $d$ is the distance from the object to the light source.

The reflection material properties for front/back of each surface is specified by OpenGL using for example,

GLfloat ambient[]=0.1,0.25,0.0,1.0

GLfloat diffuse[]=0.1,0.25,0.0,1.0

GLfloat specular[]=1.0,0.0,1.0,1.0

GLfloat emission[]=0.0,0.8,0.0,1.0

glMaterialfv(GL-front-and-back,GL-specular,specular)

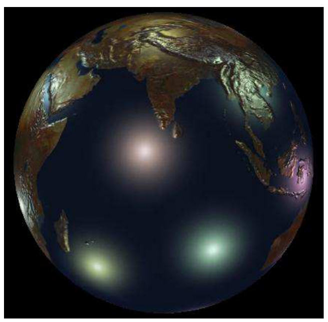
glMaterialf(GL-front-and-back,GL-shininess, 100.0)

Function glLightModel*() allows us to tell OpenGL how to carry out the lighting calculations. Since normals are reversed for back faces and the front , back faces can have different material properties, calculating shading for back faces requires extra work.

If we need correct two-sided lighting calculations (when we can see inside an object) one uses

glLightModeli(GL-LIGHT-MODEL-TWO-SIDED, GL-TRUE);

For multiple light sources, we add up the ambient, diffuse, and specular components for each light source.



## Reading Assignment and News

Please review the appropriate sections related to this weeks' lectures in chapter 5, and 11, and associated exercises, of the recommended text.

(Recommended Text: Interactive Computer Graphics, by Edward Angel, Dave Shreiner, 6th edition, Addison-Wesley)

Please track Blackboard for the most recent Announcements and Project postings related to this course.

(http://www.cs.utexas.edu/users/bajaj/graphics2012/cs354/)