So far...

- We’ve talked exclusively about geometry.
  - What is the shape of an object?
    - `glBegin()` ... `glEnd()`
  - How do I place it in a virtual 3D space?
    - `glMatrixMode()` ...
  - How to change viewpoints
    - `gluLookAt()`
  - How do I know which pixels it covers?
    - Rasterization
  - How do I know which of the pixels I should actually draw?
    - Z-buffer, BSP
So far

```cpp
glColor(...);
Apply_transforms();
Draw_objects();
```
Next...

• Once we know geometry, we have to ask one more important question:
  – To what value do I set each pixel?
• Answering this question is the job of the shading model.
• Other names:
  – Lighting model
  – Light reflection model
  – Local illumination model
  – Reflectance model
  – BRDF
An abundance of photons

- Properly determining the right color is really hard.
An abundance of photons

- Properly determining the right color is really hard.
An abundance of photons

- Properly determining the right color is really hard.

Refraction
An abundance of photons

- Properly determining the right color is really hard.
Our problem

• We’re going to build up to an approximation of reality called the **Phong illumination model**.

• It has the following characteristics:
  – *not* physically based
  – gives a “first-order” approximation to physical light reflection
  – very fast
  – widely used

• In addition, we will assume **local illumination**, i.e., light goes: light source -> surface -> viewer.

• No interreflections, no shadows.
Setup...

• Given:
  – a point $P$ on a surface visible through pixel $p$
  – The normal $N$ at $P$
  – The lighting direction, $L$, and intensity, $L$, at $P$
  – The viewing direction, $V$, at $P$
  – The shading coefficients at $P$

• Compute the color, $I$, of pixel $p$.

• Assume that the direction vectors are normalized:

$$||N|| = ||L|| = ||V|| = 1$$
“Iteration zero”

• The simplest thing you can do is...
• Assign each polygon a single color:

\[ I = k_e \]

• where
  – \( I \) is the resulting intensity
  – \( k_e \) is the **emissivity** or intrinsic shade associated with the object

• This has some special-purpose uses, but not really good for drawing a scene.
“Iteration one”

• Let’s make the color at least dependent on the overall quantity of light available in the scene:

\[ I = k_e + k_a L_a \]

– \( k_a \) is the **ambient reflection coefficient**.
  • really the reflectance of ambient light
  • “ambient” light is assumed to be equal in all directions

– \( L_a \) is the **ambient light intensity**.

• Physically, what is “ambient” light?
Ambient Term

- Hack to simulate multiple bounces, scattering of light
- Assume light equally from all directions
Wavelength dependence

• Really, $k_e$, $k_a$, and $L_a$ are functions over all wavelengths $\lambda$.
• Ideally, we would do the calculation on these functions. For the ambient shading equation, we would start with:

$$I(\lambda) = k_a(\lambda) L_a(\lambda)$$

• then we would find good RGB values to represent the spectrum $I(\lambda)$.
• Traditionally, though, $k_a$ and $I_a$ are represented as RGB triples, and the computation is performed on each color channel separately:

$$I_R = k_{a,R} L_{a,R}$$

$$I_G = k_{a,G} L_{a,G}$$

$$I_B = k_{a,B} L_{a,B}$$
Diffuse reflection

\[ I = k_e + k_a L_a \]

• So far, objects are uniformly lit.
  – not the way things really appear
  – in reality, light sources are localized in position or direction

• **Diffuse**, or **Lambertian** reflection will allow reflected intensity to vary with the direction of the light.
Diffuse reflectors

• Diffuse reflection occurs from dull, matte surfaces, like latex paint, or chalk.
• These **diffuse** or **Lambertian** reflectors reradiate light equally in all directions.

![Image of diffuse reflectors with a woman painting a wall and a model of a sheep made from wooden sticks.]
Diffuse reflectors

- Diffuse reflection occurs from dull, matte surfaces, like latex paint, or chalk.
- These **diffuse** or **Lambertian** reflectors reradiate light equally in all directions.
- Picture a rough surface with lots of tiny microfacets.
Diffuse reflectors

• ...or picture a surface with little pigment particles embedded beneath the surface (neglect reflection at the surface for the moment):

![Diagram showing light rays scattered by pigments](image)

• The microfacets and pigments distribute light rays in all directions.
• Embedded pigments are responsible for the coloration of diffusely reflected light in plastics and paints.
• Note: the figures above are intuitive, but not strictly (physically) correct.
Diffuse reflectors, cont.

- The reflected intensity from a diffuse surface does not depend on the direction of the viewer. The incoming light, though, does depend on the direction of the light source:
“Iteration two”

• The incoming energy is proportional to $\cos\theta$, giving the diffuse reflection equations:

$$I = k_e + k_a L_a + k_d L \cdot (L \cdot N)$$

$$= k_e + k_a L_a + k_d L \cdot \max(0, L \cdot N)$$

• where:
  – $k_d$ is the **diffuse reflection coefficient**
  – $L_d$ is the intensity of the light source
  – $N$ is the normal to the surface (unit vector)
  – $L$ is the direction to the light source (unit vector)
Specular reflection accounts for the highlight that you see on some objects.

It is particularly important for *smooth, shiny* surfaces, such as:

- Metal, polished stone, plastics, apples, Skin
Specular reflection

• Properties:
  – Specular reflection depends on the viewing direction \( \mathbf{V} \).
  – For non-metals, the color is determined solely by the color of the light.
  – For metals, the color may be altered (e.g., brass)
Specular reflection “derivation”

• For a perfect mirror reflector, light is reflected about \( \mathbf{N} \), so

\[
I = \begin{cases} 
L & \text{if } \mathbf{V} = \mathbf{R} \\
0 & \text{otherwise}
\end{cases}
\]

• For a near-perfect reflector, you might expect the highlight to fall off quickly with increasing angle \( \phi \).

• Also known as:
  – “rough specular” reflection
  – “directional diffuse” reflection
  – “glossy” reflection
• One way to get this effect is to take $(R \cdot V)$, raised to a power $n_s$.
• As $n_s$ gets larger,
  – the dropoff becomes {more,less} gradual
  – gives a {larger,smaller} highlight
  – simulates a {more,less} mirror-like surface
“Iteration three”

• The next update to the Phong shading model is then:

\[ I = k_e + k_a l_a + k_d \angle(\mathbf{N} \cdot \mathbf{L})_+ + k_s \angle(\mathbf{V} \cdot \mathbf{R})^{n_s}_+ \]

• where:
  
  – \( k_s \) is the **specular reflection coefficient**
  
  – \( n_s \) is the **specular exponent** or **shininess**
  
  – \( \mathbf{R} \) is the reflection of the light about the normal (unit vector)
  
  – \( \mathbf{V} \) is viewing direction (unit vector)
Specular Reflection Improvement

\[ H = \frac{(L + V)}{\|L + V\|} \]

\[ k_s L (H \cdot N)^p \]

- Compute based on normal vector and “halfway” vector, \( H \)
  - Always positive when the light and eye are above the tangent plane
  - Not quite the same result as the other formulation (need 2\( H \))
Putting It Together

\[ I = k_e + k_a L_a + L \left( k_d (L \cdot N) + k_s (H \cdot N)^p \right) \]

- Just sum all the terms
- If there are multiple lights, sum contributions from each light
- Several variations, and approximations ...
• OpenGL supports three different kinds of lights: ambient, directional, and point. Spot lights are also supported as a special form of point light.

• We’ve seen ambient light sources, which are not really geometric.

• **Directional light** sources have a single direction and intensity associated with them.
Point lights

- The direction of a **point light** sources is determined by the vector from the light position to the surface point.

\[ \mathbf{L} = \frac{\mathbf{E} - \mathbf{P}}{\|\mathbf{E} - \mathbf{P}\|} \]

\[ d = \|\mathbf{E} - \mathbf{P}\| \]

- Physics tells us the intensity must drop off inversely with the square of the distance:

\[ f_{\text{atten}} = \frac{1}{d^2} \]

- Sometimes, this distance-squared dropoff is considered too “harsh.” A common alternative is:

\[ f_{\text{atten}} = \frac{1}{a + bd + cd^2} \]

- with user-supplied constants for \( a, b, \) and \( c. \)
Spotlights

- OpenGL also allows one to apply a *directional attenuation* of a point light source, giving a **spotlight** effect.

![Spotlight Diagram]

- The spotlight intensity factor is computed in OpenGL as:

\[
 f_{\text{spot}} = (\mathbf{L} \cdot \mathbf{S})^e_\beta
\]

- where
  - \( \mathbf{L} \) is the direction to the point light.
  - \( \mathbf{S} \) is the center direction of the spotlight.
  - \( \beta \) is the cutoff angle for the spotlight
  - \( e \) is the angular falloff coefficient
  - \( (x)^e_\beta = \left[ \max\{\cos(x) - \beta, 0\} \right]^e \)
“Iteration four”

• Since light is additive, we can handle multiple lights by taking the sum over every light.

• Our equation is now:

\[ I = k_e + k_a L_a + \sum_{j} \frac{(L_j \cdot S_j)^{\epsilon_j}}{a_j + b_j d_j + c_j d_j^2} L_j \left[ k_d (N \cdot L_j)_+ + k_s (V \cdot R_j)^{n_s}_+ \right] \]

• This is the Phong illumination model.

• Which quantities are spatial vectors?

• Which are RGB triples?