Introduction to Shading
CptS 442/542

Modeling how light interacts with surfaces is complex. We will start with a relatively simple model focusing on *achromatic* light (i.e., intensity only, no color) and then generalize to colored light. The model we introduce here is based on how light reflects off surfaces (much of this model is inaccurate physically, but produces nice images).

1. Ambient light

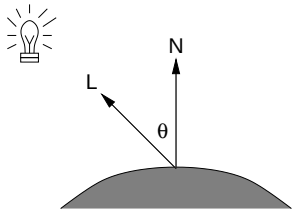
   - Light not tied to any single source, but uniformly bathes objects in the scene. Ambient light is non-directional, and is the collective product of multiple reflections of light from the many surfaces present in the environment.
   - Illumination equation

     $$I = I_a k_a$$

     I_a is the intensity of the ambient light (constant for all objects). The amount of ambient light reflected is governed by the surface's *ambient reflection coefficient* $0 \leq k_a \leq 1$ (a property of the surface's material).

2. Diffuse Reflection

   - Now we consider light emanated from a particular light source.
   - Dull surfaces (e.g. chalkboard) exhibit *diffuse reflection* (aka *Lambertian reflection*) which appears equally bright from all viewing directions.
   - The brightness depends only on the angle θ between the direction $\hat{L}$ to to the light source and the surface normal $\hat{N}$.

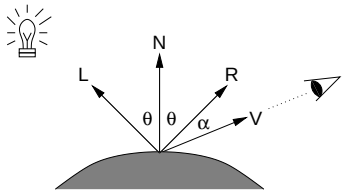
   - Illumination equation

     $$I = I_p k_d \cos\theta = I_p k_d (\hat{N} \cdot \hat{L})$$

     I_p is the intensity of the point light source, and $0 \leq k_d \leq 1$ is the material's *diffuse reflection coefficient*.

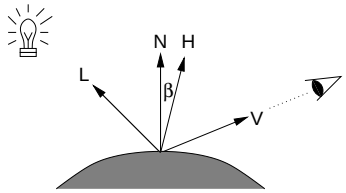
3. Specular Reflection

   - Specular reflection can be observed on a shiny surface (e.g., highlight on an apple).
   - The observed reflection depends on the position of the eye since shiny surfaces reflect light unequally in different directions.



A perfect mirror will *only* reflect light in the direction of the reflection $\hat{R}$ (which is L mirrored about R). With a mirror, the viewer will only see the reflected light if R is the same as the view vector $\hat{V}$ (i.e., $\alpha = 0$ in the above figure).

- The Phong Hack
  - The *Phong illumination model* was developed for non-perfect reflectors. It assumes maximum reflectance occurs when $\alpha = 0$, and falls of sharply as α increases. This falloff is approximated by $\cos^n \alpha$ where n is the material's *specular reflection exponent* (e.g., $n = \infty$ for a mirror, n small for a dull surface); n typically ranges from 1 to 100.
  - Illumination equation
    $$I = I_p k_s \cos^n \alpha = I_p k_s (\hat{R} \cdot \hat{V})^n$$
    $0 \leq k_s \leq 1$ is the *specular reflection coefficient*. In practice we clamp the $\cos^n \alpha$ term so that it can never be negative. We can compute $\vec{R}$ as follows:
    $$\vec{R} = 2(\hat{L} \cdot \hat{N})\hat{N} - \hat{L}.$$
  - The "halfway vector"
    In order to avoid computing R, another hack can be used to speed things up by computing the *halfway vector*
    $$\vec{H} = \vec{L} + \vec{V}$$
    and then using the illumination equation
    $$I = I_p k_s \cos^n \beta = I_p k_s (\hat{H} \cdot \hat{N})^n$$



4. Combining ambient, diffuse, and specular terms
   - For ambient light and a single point light source
     $$I = I_a k_a + I_p(k_d \cos\theta + k_s \cos^n \alpha).$$
   - For m point light sources
     $$I = I_a k_a + \sum_{i=1}^{m} I_{p_i}(k_d \cos\theta_i + k_s \cos^n \alpha_i).$$

5. Light Attenuation

- We simulate *atmospheric attenuation* of light using an *attenuation factor* f_{att}. In OpenGL, this factor is computed as follows:

$$f_{att} = \frac{1}{k_c + k_l d + f_q d^2}$$

where

d	distance between the light's position and the vertex
k_c	constant attenuation factor
k_l	linear attenuation factor
k_l	quadratic attenuation factor

Our illumination equation becomes

$$I = I_a k_a + \sum_{i=1}^{m} f_{att,i} I_{p_i} (k_d \cos\theta_i + k_s \cos^n \alpha_i).$$

6. Colored lights and surfaces

- We could define an object's diffuse color by its red, green, and blue components as (O_{dR}, O_{dG}, O_{dB}). We could also represent a light's illuminated color as (I_{pR}, I_{pG}, I_{pB}). The illuminated surface colors for a singled colored light become:

$$\begin{aligned} I_R &= I_{aR} k_a O_{dR} + f_{att} I_{pR} (k_d O_{dR} \cos\theta + k_s \cos^n \alpha) \\ I_G &= I_{aG} k_a O_{dG} + f_{att} I_{pG} (k_d O_{dG} \cos\theta + k_s \cos^n \alpha) \\ I_B &= I_{aB} k_a O_{dB} + f_{att} I_{pB} (k_d O_{dB} \cos\theta + k_s \cos^n \alpha) \end{aligned}$$

- Note that the above equations consider the diffuse color of a surface to be the same as its ambient color. The specular color is generally governed by the light source.
- OpenGL combines colors and coefficients into single coefficients yielding the following equations:

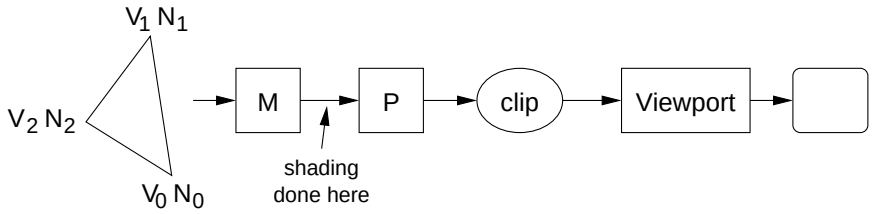
$$\begin{aligned} I_R &= I_{aR} k_{aR} + f_{att} (I_{dR} k_{dR} \cos\theta + I_{sR} d_{sR} \cos^n \alpha) \\ I_G &= I_{aG} k_{aG} + f_{att} (I_{dG} k_{dG} \cos\theta + I_{sG} d_{sG} \cos^n \alpha) \\ I_B &= I_{aB} k_{aB} + f_{att} (I_{dB} k_{dB} \cos\theta + I_{sB} d_{sB} \cos^n \alpha) \end{aligned}$$

- Colored lights
  - The ambient light color is (I_{aR}, I_{aG}, I_{aB}). Even though we do not consider point light sources to have an ambient term, OpenGL strangely enough allows light sources to have an ambient term (isn't this contrary to the idea of ambient light?). OpenGL also allows for a "model ambience" which is not tied to any particular light source.
  - The diffuse and specular light colors are (I_{dR}, I_{dG}, I_{dB}) and (I_{sR}, I_{sG}, I_{sB}) respectively (most of the time these are white).
- Material colors/coefficients (i.e. properties)

ambient reflection coefficients	k_{aR}, k_{aG}, k_{aB}
diffuse reflection coefficients	k_{dR}, k_{dG}, k_{dB}
specular reflection coefficients	k_{sR}, k_{sG}, k_{sB}

- Other material/light properties
  Shininess n, attenuation properties, ...

7. Shading in the graphics pipeline



- Transforming normal vectors
  - In OpenGL, a normal vector is associated with each vertex of a polygon. If the model-view matrix used to transform the vertices is M, we use $(M^{-1})^T$ to transform the normal vectors (remember how M transforms a plane).
    - The homogeneous coordinate of a direction vector is 0 (e.g., $\hat{N} = (N_x, N_y, N_z, 0)$).
    - For this to work right, M must be a rigid transformation (e.g., no scales or shears). A uniform scale can be used if you are willing to renormalize $\hat{N}$ after multiplication by M.
  - `glNormal3f()` sets the "current normal vector" which is applied to subsequent vertices passed to `glVertex3f()`.
  - `glShadeModel(GL_SMOOTH)` tells OpenGL to use different normals for each vertex (polygon will be shaded by interpolating vertex colors).
  - `glShadeModel(GL_FLAT)` tells OpenGL to use the same normal for each vertex (all pixels in the polygon will end up the same color).
- Light sources are objects too and are transformed by the model-view matrix (often this is *not* what you want).
- After passing through the model-view matrix, but before being passed through the projection matrix, vertex normals are converted to color values (Gouraud Shading). Each vertex is now associated with a color instead of a normal. The colors are clipped along with the vertices. More on this later...