Reflection and Light Source Models

Illumination: Energy physics...

- Radiance: the flux of light energy in a given direction
- Geometry/Visibility: how light energy falls upon a surface
- BRDF: the interaction function of a surface point with light
- Energy Balance Equation: the local balance of energy in a scene

Approximation: Hacks for interaction at a point...

- Ambient: approximating the global energy
- Lambertian: approximating the diffuse interaction
- Phong: approximating the specular interaction

Reflection VS. Illumination

Light: An electromagnetic *energy flux* that has

- intensity (power per unit area)
- direction of propagation

Reflection: A *local lighting model* that relates

- the properties of a surface at a point
- the incoming direction and energy at the point
- the outgoing direction and energy at the point

BRDF: *bidirectional reflectance distribution function*

- the function that embodies the surface properties

Illumination: A *global lighting model* that computes

- overall light distribution in an environment
  - from the reflection models
  - from the shape and location of all objects
  - from the shape and location of all light sources

Shading: A *local interpolation technique* used to

- reduce the cost of computing reflection
- shade polygons "nicely"

Energy of Illumination

Radiance: Electromagnetic *energy flux*, the amount of energy traveling

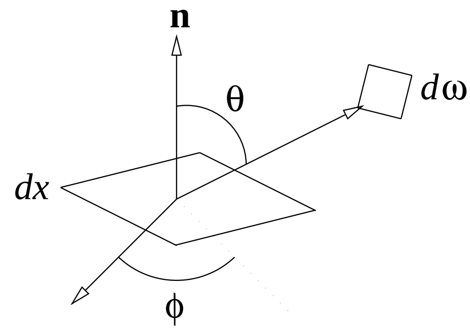
- at some point x
- in a specified direction θ, ϕ
- per unit time
- per unit area perpendicular to the direction
- per unit solid angle
- for a specified wavelength λ
- denoted by $L(x, \theta, \phi, \lambda)$

Spectral Properties: Total energy flux comes from flux at each wavelength

- $L(x, \theta, \phi) = \int_{\lambda_{\min}}^{\lambda_{\max}} L(x, \theta, \phi, \lambda) d\lambda$

Picture: For the indicated situation $L(x, \theta, \phi) dx \cos\theta d\omega dt$ is

- energy radiated through differential solid angle $d\omega = \sin\theta d\theta d\phi$
- through/from differential area dx
- not perpendicular to direction (projected area is $dx \cos\theta$)
- during differential unit time dt



Power: Energy per unit time (as in the picture)

- $L(x, \theta, \phi) dx \cos\theta d\omega$

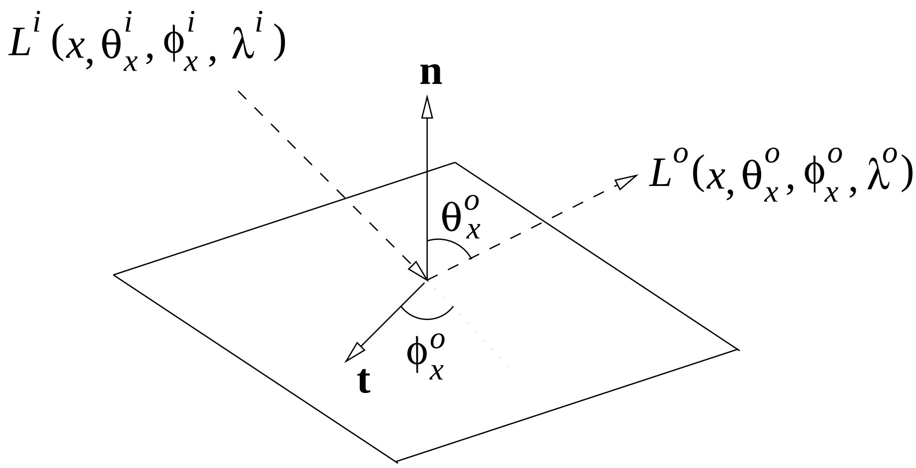
Radiosity: Total power leaving a surface point per unit area

- $\int_\Omega L(x, \theta, \phi) \cos\theta d\omega = \int_0^{\frac{\pi}{2}} \int_0^{2\pi} L(x, \theta, \phi) \cos\theta \sin\theta d\phi d\theta$ (integral is over the hemisphere above the surface point)

Bidirectional Reflectance Distribution Function:

- is a surface property at a point
- relates energy in to energy out
- depends on incoming and outgoing directions
- varies from wavelength to wavelength
- Definition: Ratio
  - of radiance in the outgoing direction
  - to radiant flux density for the incoming direction

$$\rho_{bd}(x, \theta_i, \phi_i, \lambda_i, \theta_o, \phi_o, \lambda_o) = \frac{L^o(x, \theta_x^o, \phi_x^o, \lambda^o)}{L^i(x, \theta_x^i, \phi_x^i, \lambda^i)\cos\theta_x^i d\omega_x^i}$$



Energy Balance Equation

$$
\begin{aligned}
L^o(x, \theta_x^o, \phi_x^o, \lambda^o) &= L^e(x, \theta_x^o, \phi_x^o, \lambda^o) \\
&\quad + \int_0^{\frac{\pi}{2}} \int_0^{2\pi} \int_{\lambda_{min}}^{\lambda_{max}} \rho_{bd}(x, \theta_x^i, \phi_x^i, \lambda^i, \theta_x^o, \phi_x^o, \lambda^o) \\
&\qquad \cos(\theta_x^i) L^i(x, \theta_x^i, \phi_x^i, \lambda^i) d\lambda^i \sin(\theta_x^i) d\phi_x^i d\theta_x^i
\end{aligned}
$$

- $L^o(x, \theta_x^o, \phi_x^o, \lambda^o)$ is the radiance
  - at wavelength λ^o
  - leaving point x
  - in direction θ_x^o, ϕ_x^o
- $L^e(x, \theta_x^o, \phi_x^o, \lambda^o)$ is the radiance emitted by the surface from the point
- $L^i(x, \theta_x^i, \phi_x^i, \lambda^i)$ is the incident radiance impinging on the point
- $\rho_{bd}(x, \theta_x^i, \phi_x^i, \lambda^i, \theta_x^o, \phi_x^o, \lambda^o)$ is the BRDF at the point
  - describes the surface's interaction with light at the point
- the integration is over the hemisphere above the point

Fast and Dirty Approximations

Rough Approximations:

- Use red, $green$, and $blue$ instead of full spectrum
  - Roughly follows the eye's sensitivity
  - Forego such complex surface behavior as metals
- Use finite number of point light sources instead of full hemisphere
  - Integration changes to summation
  - Forego such effects as soft shadows and color bleeding
- BRDF behaves independently on each color
  - Treat red, green, and blue as three separate computations
  - Forego such effects as iridescence and refraction
- BRDF split into three approximate effects
  - Ambient: constant, nondirectional, background light
  - Diffuse: light reflected uniformly in all directions
  - Specular: light of higher intensity in mirror-reflection direction
- Energy flux L replaced by simple "intensity" I
  - No pretense of being physically true

Approximate Intensity Equation: (single light source)

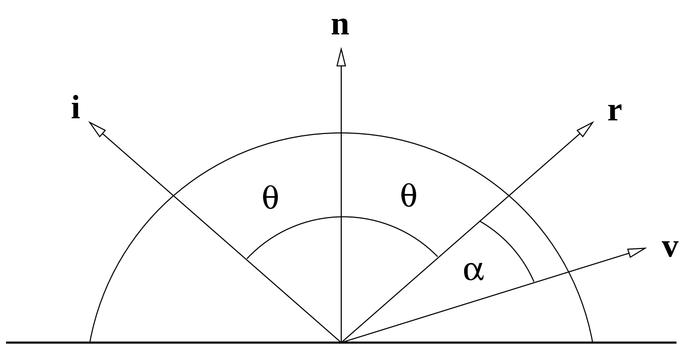
$$I^o_\lambda = I^e_\lambda + k^a_\lambda I^a_\lambda + k^d_\lambda I^l_\lambda \cos(\theta^l) + k^s_\lambda I^l_\lambda W(\theta^l) S(\alpha^l)$$

- λ stands for each of *red, green, blue*
- I^l_λ is the intensity of the light source (modified for distance)
- $\cos(\theta^l)$ accounts for the projected cross-sectional area of the incoming light
- the k are between 0 and 1 and represent absorption factors
- $W(\theta^l)$ accounts for any highlight effects that depend on the incoming direction
  - use $\cos(\theta^l)$ if there is nothing special
- α^l is the mirror reflection angle for the light
  - the angle between the view direction and the mirror reflection direction
- $S(\alpha^l)$ accounts for highlights in the mirror reflection direction
- the superscripts e, a, d, s stand for *emitted, ambient, diffuse, specular* respectively
- sum over each light l if there are more than one

Lambertian Reflection Model

Diffuse Geometry:

- **i** is the *unit vector* in the direction of the illumination (light source)
- **n** is the *unit vector* normal to the surface
- **r** is the *unit vector* in the mirror reflection direction
- **v** is the *unit vector* in the direction of the eyepoint



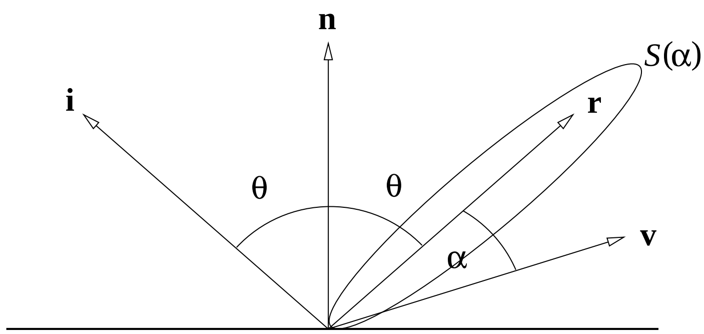
Formulas:

- $\cos(\theta) = \mathbf{n} \cdot \mathbf{i}$
- **r** and α are not needed

Phong Reflection Model

Specular Geometry (Phong Model):

- **i** is the *unit vector* in the direction of the illumination (light source)
- **n** is the *unit vector* normal to the surface
- **r** is the *unit vector* in the mirror reflection direction
- **v** is the *unit vector* in the direction of the eyepoint



Formulas:

- $\cos(\theta) = \mathbf{n} \cdot \mathbf{i}$
- $\mathbf{r} = 2(\mathbf{i} \cdot \mathbf{n})\mathbf{n} - \mathbf{i}$
- $\cos(\alpha) = \mathbf{r} \cdot \mathbf{v}$
- $S(\alpha) = \cos(\alpha)^{n_s}$

Point Light Sources

Point Light Sources:

- Point light sources has a **position** P_i and an **intensity** I_i
- Light energy is radiated equally in all directions

Distance Attenuation:

- Physically, need $1/r^2$ attenuation since light energy spreads out spherically
- This is too harsh, point light sources are rare in the real world
- Use modified attenuation factor:

$$a(r) = \frac{1}{\alpha_0 + \alpha_1 r + \alpha_2 r^2}$$

- This **simulates** the attenuation of an *area light source*
- Only use attenuation from light source to surface, **not** from surface to pixel
- Pixel is **area**, not point, so foreshortening cancels attenuation

Ambient Lighting:

- True global illumination difficult and expensive to calculate
- Often use constant low level lighting everywhere to fake global illumination: I_λ^a
- Each surface may reflect "ambient" lighting differently: k_λ^a
- Usually, $k_\lambda^a = k_\lambda^d$

Lambertian Lighting:
at point x with ℓ point light sources at points p_i is now:

$$\begin{aligned} d_i &= |p_i - x|\,, \\ a(d_i) &= \frac{1}{\alpha_{0i} + \alpha_{1i} d_i + \alpha_{2i} d_i^2}, \\ \mathbf{i_i} &= (p_i - x)/d_i, \\ I^o_\lambda &= k^a_\lambda I^a_\lambda + k^d_\lambda \sum_{i=1}^{\ell} a(d_i) I^i_\lambda |\mathbf{i_i} \cdot \mathbf{n}| \end{aligned}$$

Specular Lighting Similarly: For example,

$$\begin{aligned} \mathbf{r_i} &= 2\mathbf{n}(\mathbf{n} \cdot \mathbf{i_i}) - \mathbf{i_i} \\ I^o_\lambda &= k^a_\lambda I^a_\lambda + \sum_{i=1}^{l} a(d^i) I^i_\lambda (k^d_\lambda |\mathbf{i_i} \cdot \mathbf{n}| + k^s_\lambda |\mathbf{r_i} \cdot \mathbf{v}|^{ns}) \end{aligned}$$

- Sometimes we see the Phong model stated with the half-vector **h**:
  - $\mathbf{h} = (\mathbf{i} + \mathbf{v})/2$
  - Angle between **n** and **h** is twice that between **r** and **v** if coplanar
  - Use $|\mathbf{h} \cdot \mathbf{n}|^{ns}$
  - Not exactly the equivalent of using reflection vector **r**
  - Avoids recomputation of **v**

Shading

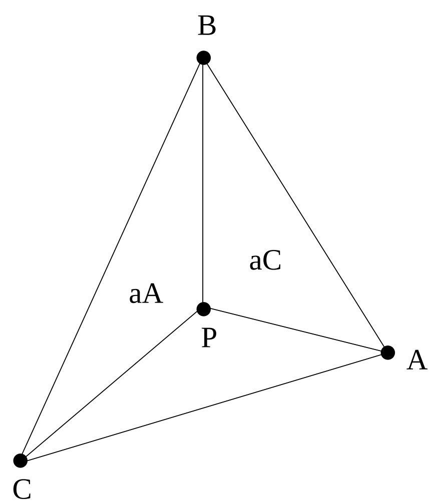
Shading algorithms apply lighting models to polygons, through interpolation from the vertices.

Gouraud Shading: Lighting in only computed at the vertices, and the colors are interpolated across the (convex) polygon

Phong Shading: A normal is specified at each vertex, and this normal is interpolated across the polygon. At each pixel, a lighting model is calculated.

Gouraud Shading

- *Gouraud shading* interpolates colors across a polygon from the vertices
- Lighting calculations are only performed at the vertices
- Highlights can be missed or blurred
- Common in hardware renderers; model that OpenGL supports
- Gouraud shading is well-defined only for triangles...
  Equivalent to a *barycentric combination*
- Barycentric combinations are also *affine combinations*...
  Triangular Gouraud shading is *invariant* under affine transformations



aA = D P B C/D A B C

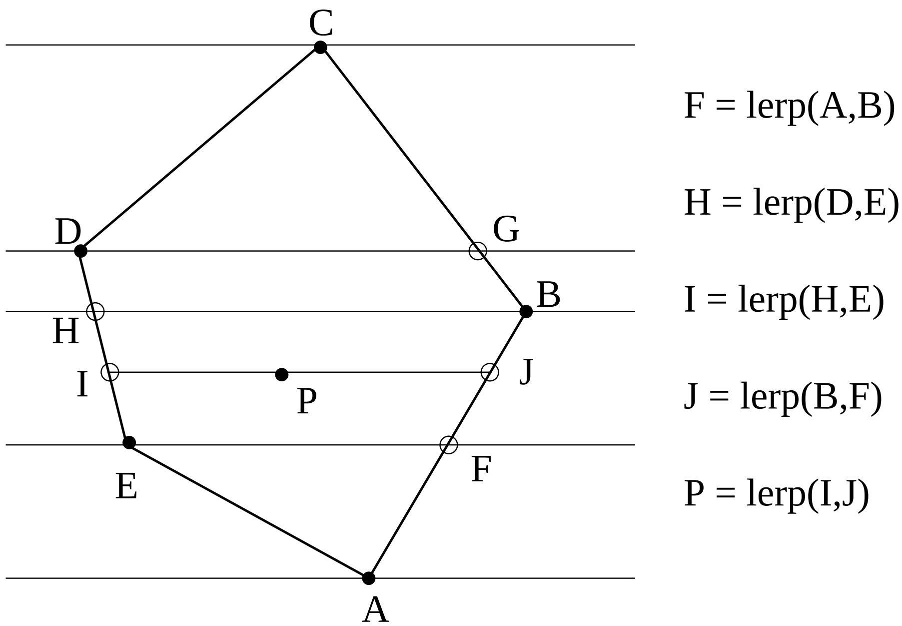
aB = D A P C/D A B C

aC = D A B P/D A B C

aA + aB + aC = 1

P = aA A + aB B + aC C

- For polygons with more than three vertices:
  - Sort the vertices by y coordinate
  - Slice the polygon into trapezoids with parallel top and bottom
  - Interpolate colors along each edge of the trapezoid...
  - Interpolate colors along each scanline



- Gouraud shading gives *bilinear* interpolation within each trapezoid
- Since rotating the polygon can result in a different trapezoidal decomposition, n-sided Gouraud interpolation is *not affine invariant*
- Exercise: Provide an example of the above effect.
- Exercise: Prove the above algorithm forms a barycentric combination on triangles

Phong Shading

- *Phong Shading* interpolates lighting model parameters, *not* colors
- Much better rendition of highlights
- A *normal* is specified at each vertex of a polygon
- Vertex normals are independent of the polygon normal
- Vertex normals should relate to the surface being approximated by the polygon
- The normal is interpolated across the polygon (using Gouraud techniques).
- At each pixel,
  - Interpolate the normal...
  - Interpolate other shading parameters...
  - Compute the view and light vectors...
  - Evaluate the lighting model
- The lighting model does not have to be the Phong lighting model...
- Normal interpolation is nominally done by vector addition and renormalization
- Several "fast" approximations are possible
- The view and light vectors may also be interpolated or approximated