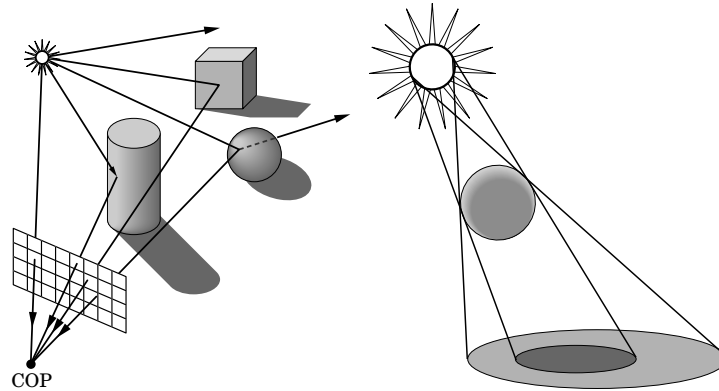


6. Illumination, Lighting

No ray tracing in OpenGL!

ray tracing: direct paths



interreflection: soft shadows, color bleeding.
umbra, penumbra, shadows: `Cex/cubes.c`

Ray-object intersection reduces to root finding.

Illumination

Global lighting model = energy preservation:

see e.g. [Cohen,Wallace]: Radiosity and realistic image synthesis, AP

Radiance: $r(\text{position, direction, wavelength}) == \text{energy flux}$

. (direction: expressed as 2 Euler angles; wavelength indicates spectrum)

Bidirectional Reflectance Distribution Function:

. $BRDF(\text{position, dir-in, spectrum-in, dir-out, spectrum-out}) \dots$

polarization, delay

Radiosity: $\int_{\text{hemisphere}} r \cos(\theta) \sin(\theta) d\theta d\phi$,

where $\cos(\theta)$ stands for $\cos(\theta_1) \cos(\theta_2)$

$$r^{\text{out}} = r^{\text{emit}} + \int_{\text{hemisphere}} \int_{\text{spectrum}} BRDF \cdot \text{Radiosity}^{\text{in}} .$$

Finite elements:

prior to rendering, split primitives into patches into elements (pieces);
 distribute energy from the current maximal one around scene
 Cost increased by smaller elements, more steps.

Problem: scene meshing required (of CSG, trimmed NURBS, procedurals, subdivision triangles, large scenes)

Monte Carlo evaluation: lazy evaluation of integral (randomized Riemann sum), during rendering
 Scales well

BRDF– form factors (advanced)

- Reduction of one patch to a point (else subdivide). (transport theory: $P(x) = p(x)dV$;
- Hemi-cube instead of hemisphere: project scene onto Hemi-cube face and for every face record closest (in *item buffer*) object and add its form factor.
 Can use z-buffer; finite resolution; includes visibility calculation.
- Nusselt's analogy: Two surfaces with the same projection have the same form factor
- Radiosity (r) Integral Equation:

Terminology:

flux: energy (number of particles) per time, per area

radiance: energy (number of particles) per time, per area, per solid angle

$dA = (rd\theta)(r \sin \theta d\phi)$.

solid angle ω subtended by a spherical area: area on unit sphere (steradians=radians squared; total on sphere = 4π)

Differential solid angle: $d\omega = \sin \theta d\theta d\phi$.

Radiosity = Luminosity (photometric)

OpenGL Approximation:

- spectrum → RGB (no refraction, incandescence)
- Radiosityⁱⁿ → sum over light sources (no soft shadows,color bleed)
- BRDF → ambient,diffuse,specular
- Radiance → intensity.

Lighting in RGBA mode

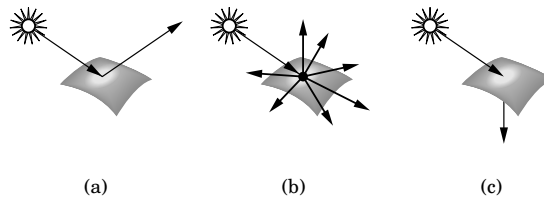
	m	material
	ℓ	light source
	l	lighting model
	\mathbf{v}	vertex
Intensity is associated with	\mathbf{n}	normal
	\mathbf{p}	light position
	\mathbf{e}	eye position
	d	$\ \mathbf{p} - \mathbf{v}\ $
	$\mathbf{s} = \frac{\mathbf{s}'}{\ \mathbf{s}'\ }$	$\mathbf{s}' := \frac{\mathbf{v}-\mathbf{p}}{\ \mathbf{v}-\mathbf{p}\ } + \frac{\mathbf{v}-\mathbf{e}}{\ \mathbf{v}-\mathbf{e}\ },$

The intensity of one of the R,G,B channels at one vertex is computed as

$$\begin{aligned}
 \text{intensity} &= \text{emission}_m \\
 &+ \text{ambient}_\ell \cdot \text{ambient}_m \\
 &+ \sum_{\text{lights}} \frac{1}{k_0 + k_1 d + k_2 d^2} \cdot \text{spot}_\ell \cdot [\dots \\
 &\dots \text{ambient}_\ell \cdot \text{ambient}_m \dots \\
 &\dots + \max\{(\mathbf{p} - \mathbf{v}) \cdot \mathbf{n}, 0\} \cdot \text{diffuse}_\ell \cdot \text{diffuse}_m \dots \\
 &\dots + \max\{\mathbf{s} \cdot \mathbf{n}, 0\}^{\text{shininess}} \cdot \text{specular}_\ell \cdot \text{specular}_m]
 \end{aligned}$$

Note Lights are objects! They are affected by model-view transformations.

Aggregation cancels **attenuation**
 glColor is ignored after `glEnable(GL_LIGHTING)`
 Spot light cone: `glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 45.0);`
 . 180 = everywhere spot: $\max((\mathbf{v} - \mathbf{p}) \cdot \text{spotdir})^{\text{spotexp}}$ or 0 or 1.



- specular (Phong) laser beam, mirror

- *diffuse* (Lambertian) nature, equal scattering (but still directional light source)
- *ambient* (global energy) background glow, equal scattering

Question: Given a unit sphere, where is the highlight (point of highest intensity)? Reduce to plane through 0, \mathbf{e} , \mathbf{p} since \mathbf{n} lies in that plane.

translucency refraction $\alpha = \alpha_m. \quad (1 - \alpha)Z_0 + \alpha((1 - \alpha)Z_1 + \alpha(\dots))$

Computing Normals

Surface in *implicit* representation $p(\mathbf{x}) = p(x, y, z) = 0$.

The normal is the (normalized) gradient $\nabla p = \begin{bmatrix} \frac{\partial}{\partial x} p \\ \frac{\partial}{\partial y} p \\ \frac{\partial}{\partial z} p \end{bmatrix}$ 10spnor.maple

Surface in *parametric* representation $\mathbf{x}(u, v) = \begin{bmatrix} x(u, v) \\ y(u, v) \\ z(u, v) \end{bmatrix}$.

The normal is $\frac{\partial \mathbf{x}}{\partial u} \times \frac{\partial \mathbf{x}}{\partial v}$.

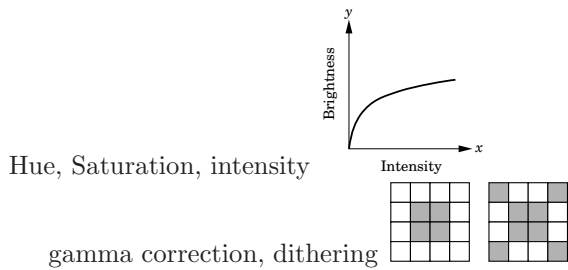
Blackboard Examples: $p(\mathbf{x}) = x^2 + y^2 + z^2 - 1$, $\mathbf{x}(u, v) = \begin{bmatrix} \cos(u) \cos(v) \\ \cos(u) \sin(v) \\ \sin(u) \end{bmatrix}$

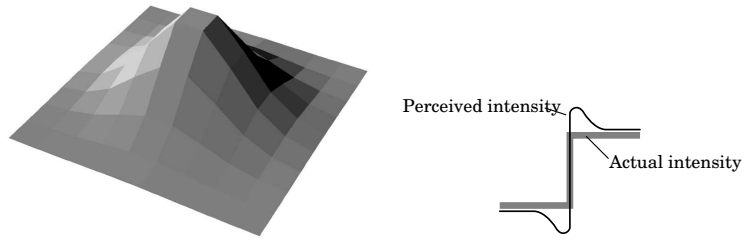
GL_LIGHT_MODEL_TWO_SIDE (a look inside the teapot)

Polygonal Shading

- flat
- *Gouraud*: averaged vertex color using barycentric weights.
- *Phong*: averaged vertex normal (and other lighting factors)

Color Gamut





Example programs

OPENGL/6light.c

OPENGL/6colorformat.c (mouse buttons = rgb diffuse)

OPENGL/6movelight.c

stationary, independent+moving, viewpoint-moving lights