Lighting

Dr. Scott Schaefer

TAMU
Computer
Science

Lighting/Illumination

- Color is a function of how light reflects from surfaces to the eye

- *Global illumination* accounts for light from all sources as it is transmitted throughout the environment
- *Local illumination* only accounts for light that directly hits a surface and is transmitted to the eye

Global Illumination

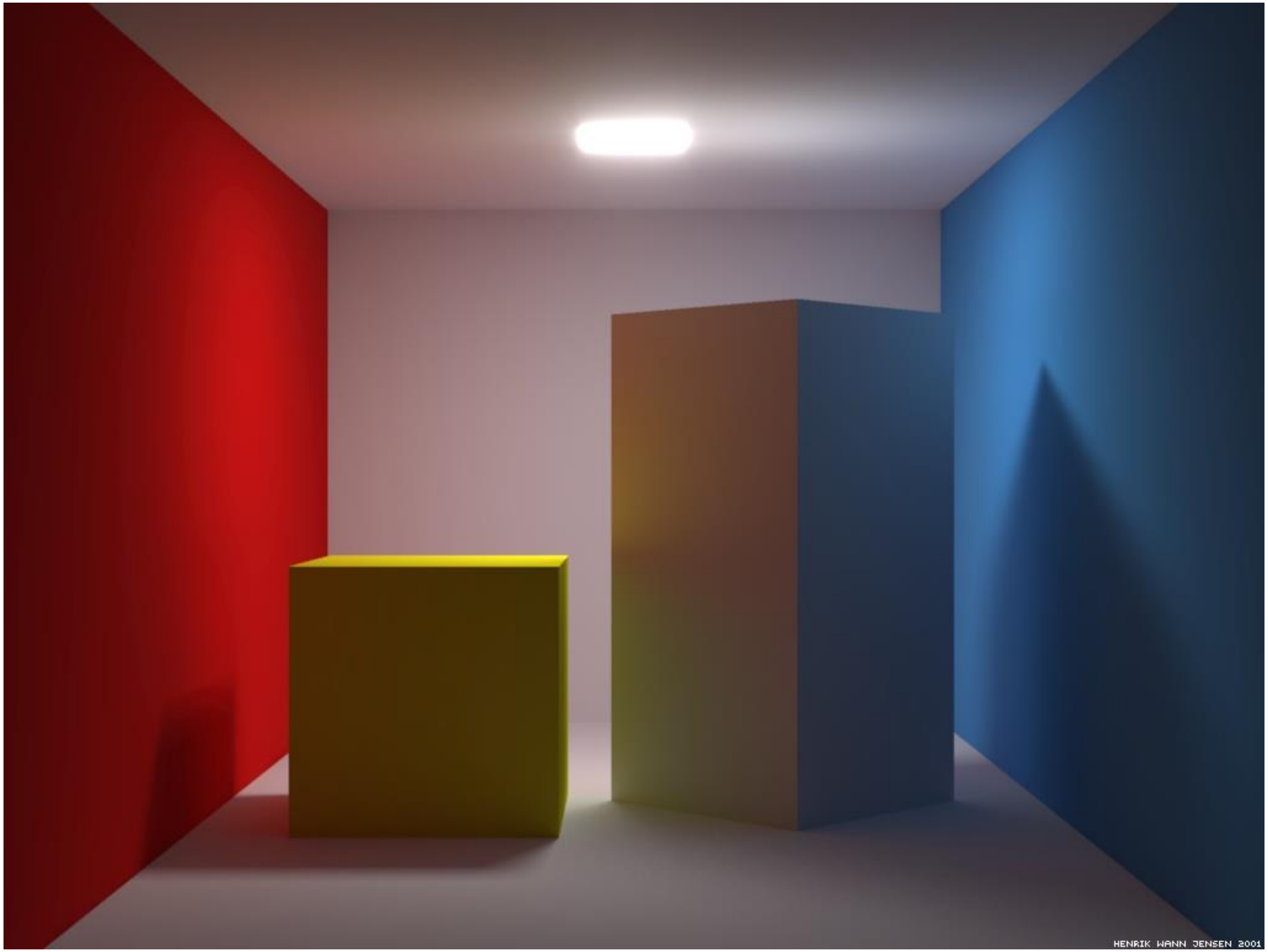
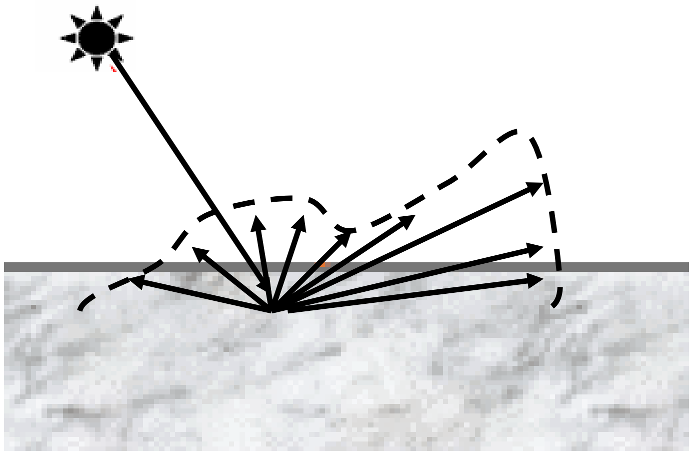


Image taken from http://graphics.ucsd.edu/~henrik/images/cbox.html

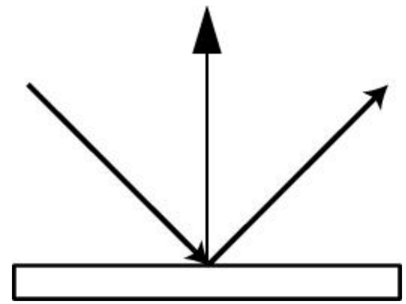
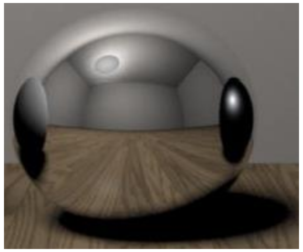
Reflection Models

- Definition: Reflection is the process by which light incident on a surface interacts with the surface such that it leaves on the incident side without change in frequency.

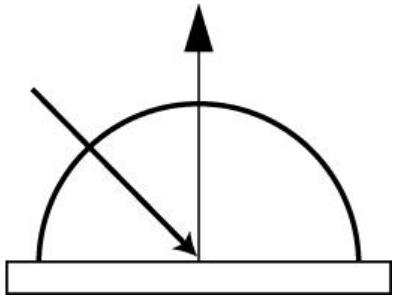
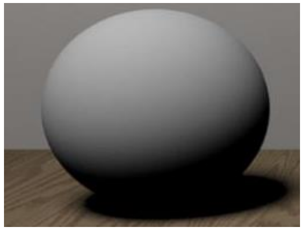


Types of Reflection Functions

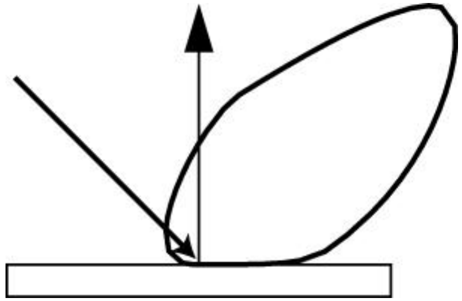
- Ideal Specular
  - Reflection Law
  - Mirror

- Ideal Diffuse
  - Lambert's Law
  - Matte

- Specular
  - Glossy
  - Directional diffuse




Illumination Model

- Ambient Light
  - Uniform light caused by secondary reflections
- Diffuse Light
  - Light scattered equally in all directions
- Specular Light
  - Highlights on shiny surfaces

Ambient Light

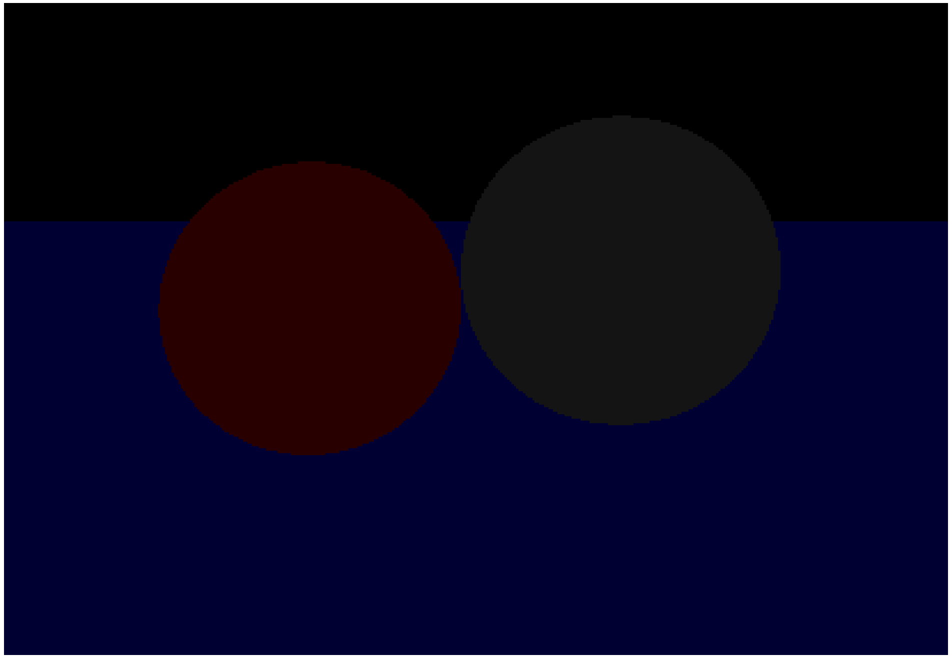
$$I = k_a A$$

- A =intensity of ambient light
- k_a=ambient reflection coefficient

- Really 3 equations! (Red, Green, Blue)
- Accounts for indirect illumination
- Determines color of shadows

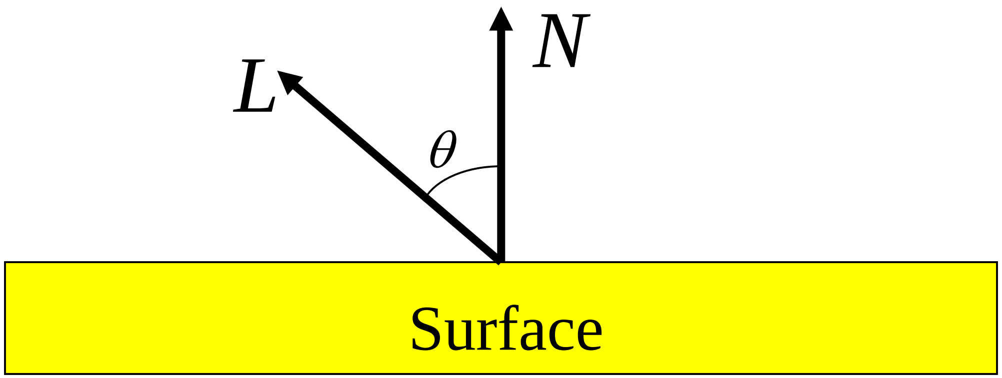
Total Illumination

$$I = k_a A$$



Diffuse Light

- Assumes that light is reflected equally in all directions
- Handles both local and infinite light sources
  - Infinite distance: *L* doesn't change
  - Finite distance: must calculate L for each point on surface

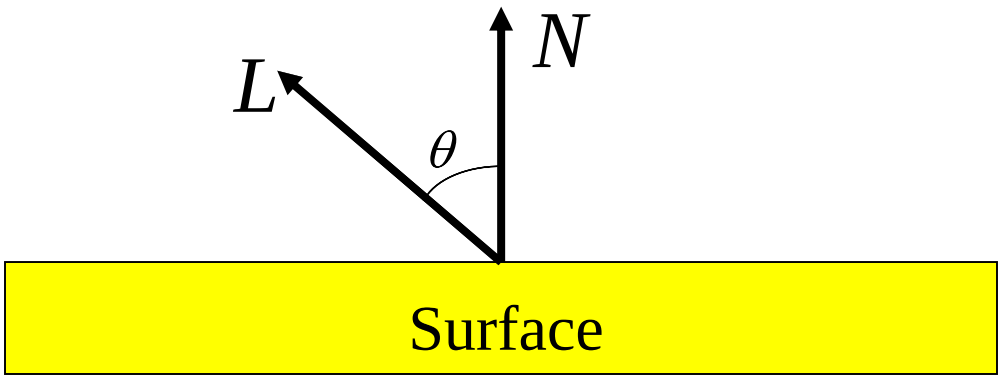


Diffuse Light

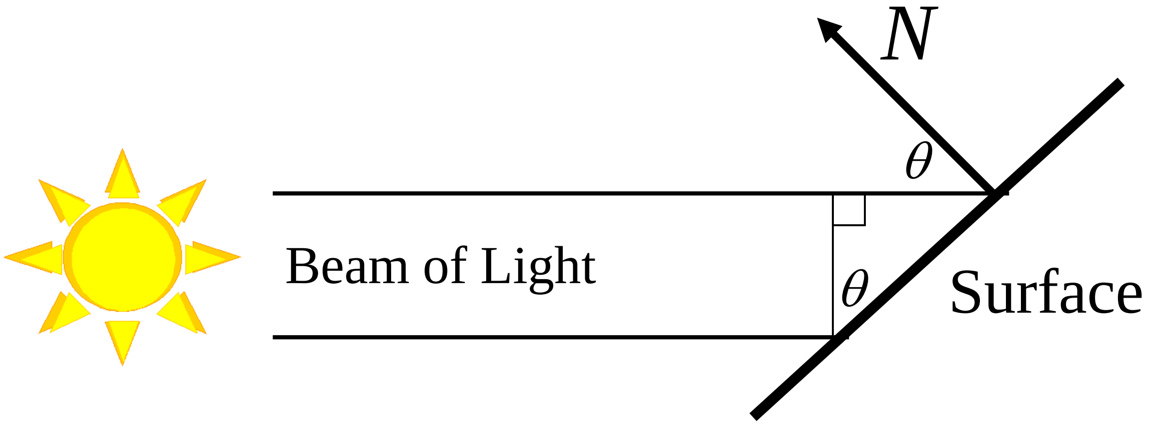
$$I = Ck_d \cos(\theta) = Ck_d (L \cdot N)$$

- C = intensity of point light source
- k_d = diffuse reflection coefficient
- θ = angle between normal and direction to light

$$\cos(\theta) = L \cdot N$$

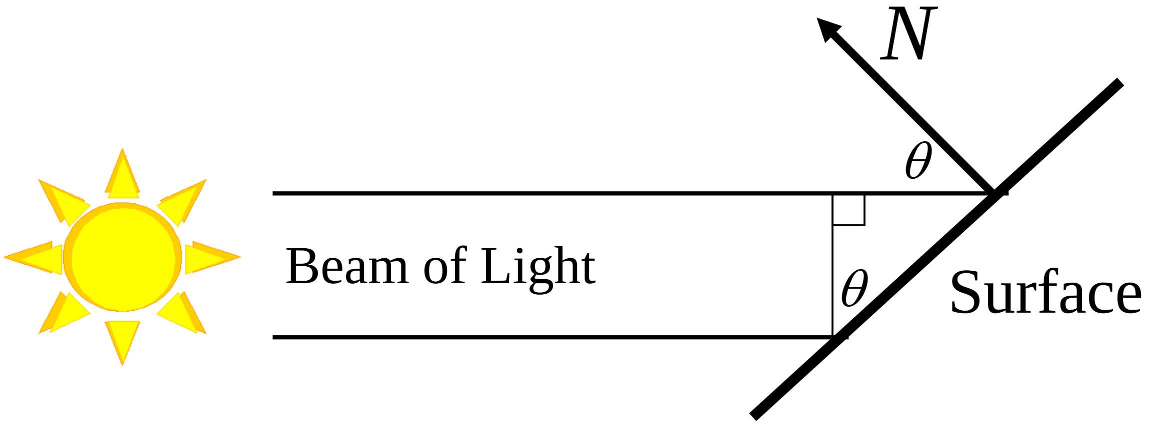


Lambert's Law



$$I = \frac{Light}{Area} = \frac{Beam\ Width \times I_{source}}{Surface\ Area}$$

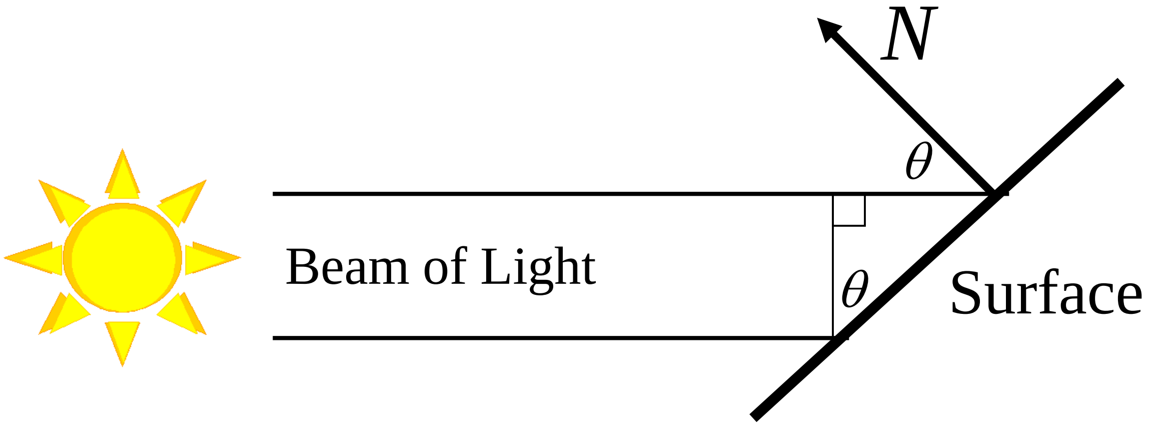
Lambert's Law



$$I = \frac{Light}{Area} = \frac{Beam\ Width \times I_{source}}{Surface\ Area}$$

$$\frac{Beam\ Width}{Surface\ Area} = \cos(\theta)$$

Lambert's Law

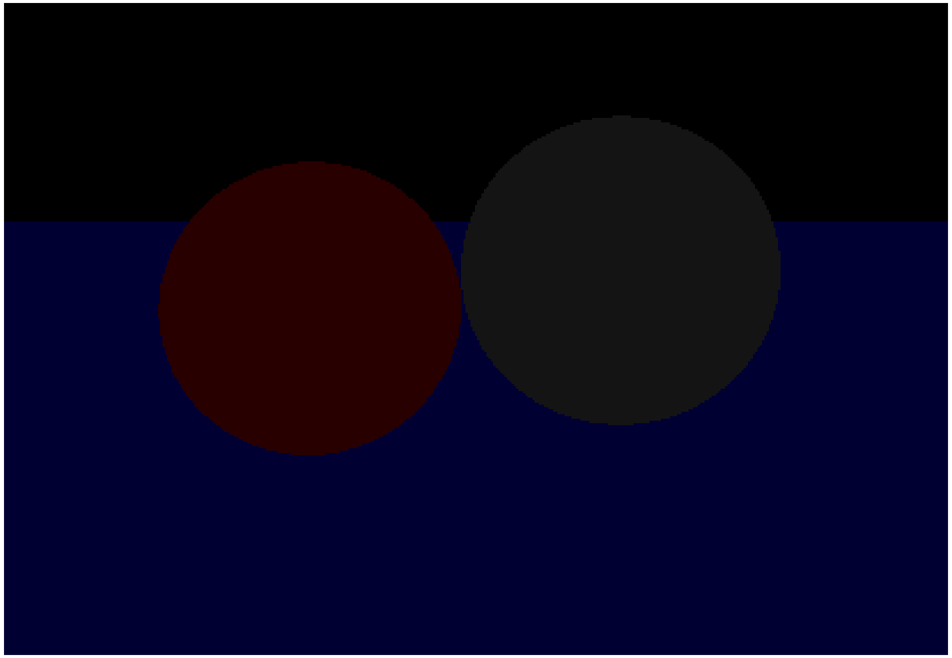


$$I = \frac{Light}{Area} = \frac{Beam\ Width \times I_{source}}{Surface\ Area} = I_{source}(L \cdot N)$$

$$\frac{Beam\ Width}{Surface\ Area} = \cos(\theta)$$

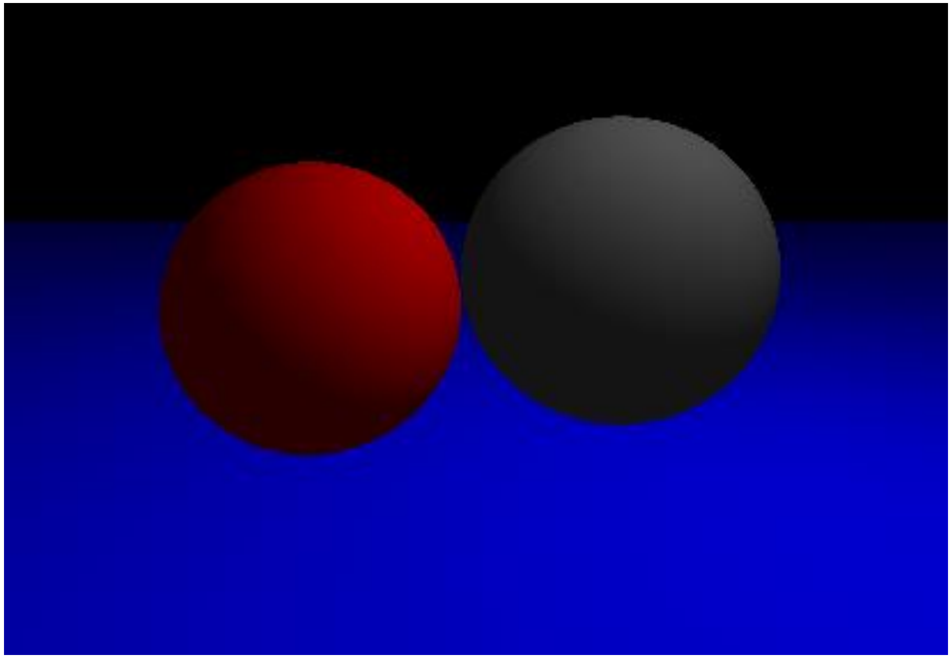
Total Illumination

$$I = k_a A$$



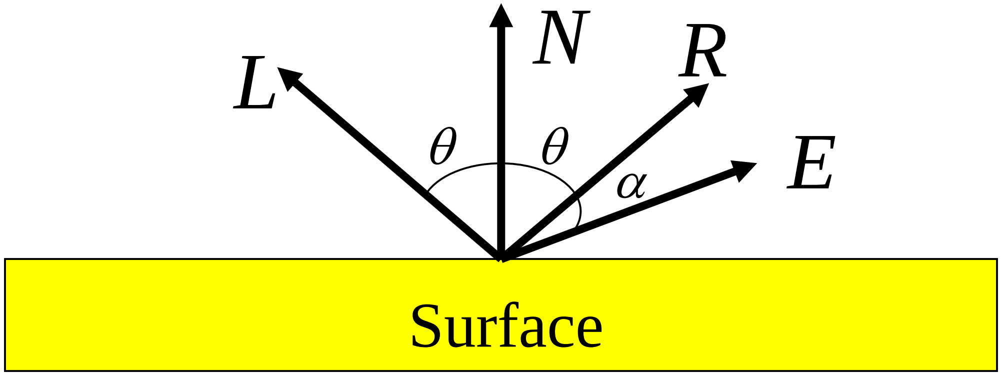
Total Illumination

$$I = k_a A + k_d C(L \cdot N)$$



Specular Light

- Perfect, mirror-like reflection of light from surface
- Forms highlights on shiny objects (metal, plastic)

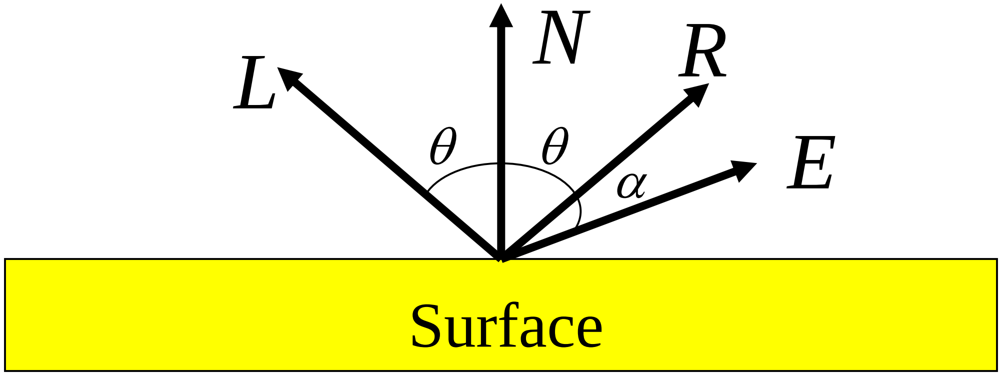


Specular Light

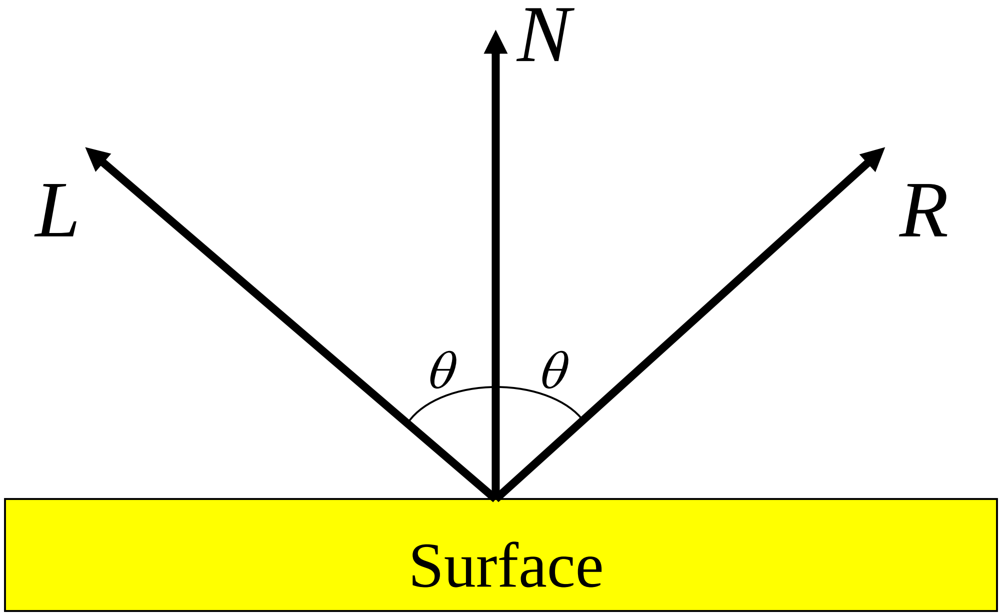
$$I = Ck_s \cos^n(\alpha) = Ck_s (R \cdot E)^n$$

- C = intensity of point light source
- k_s = specular reflection coefficient
- α = angle between reflected vector (R) and eye (E)
- n = specular exponent

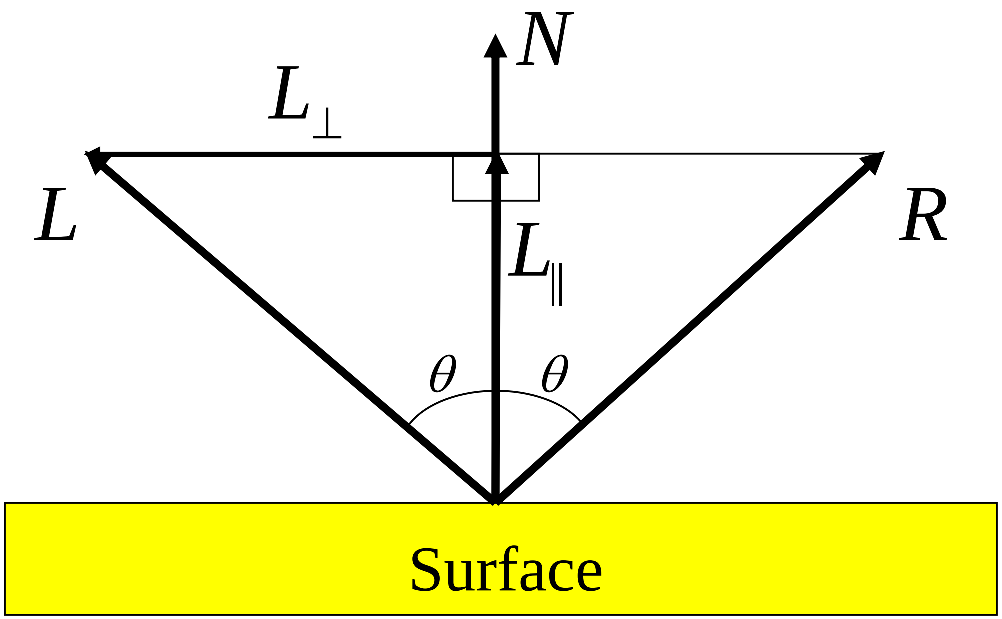
$$\cos(\alpha) = R \cdot E$$



Finding the Reflected Vector



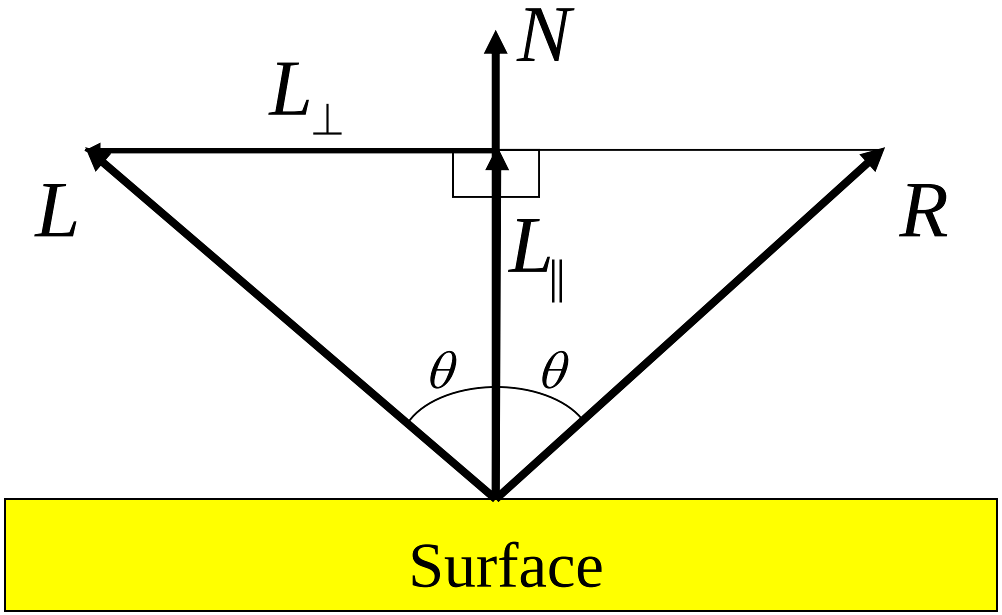
Finding the Reflected Vector



Finding the Reflected Vector

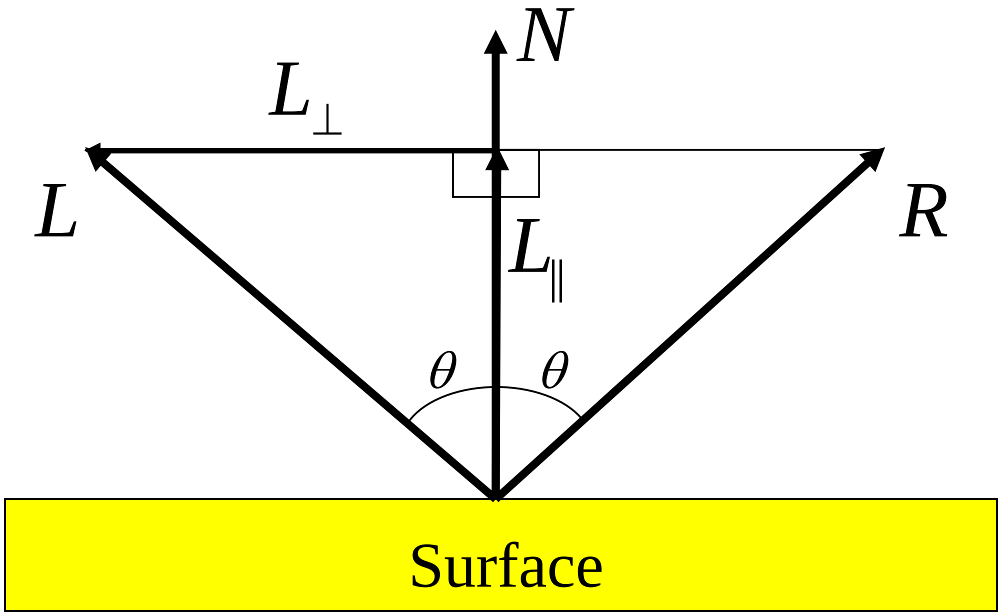
$$L_{\parallel} = N\cos(\theta) = N(L \cdot N)$$

$$L_{\perp} = L - L_{\parallel}$$



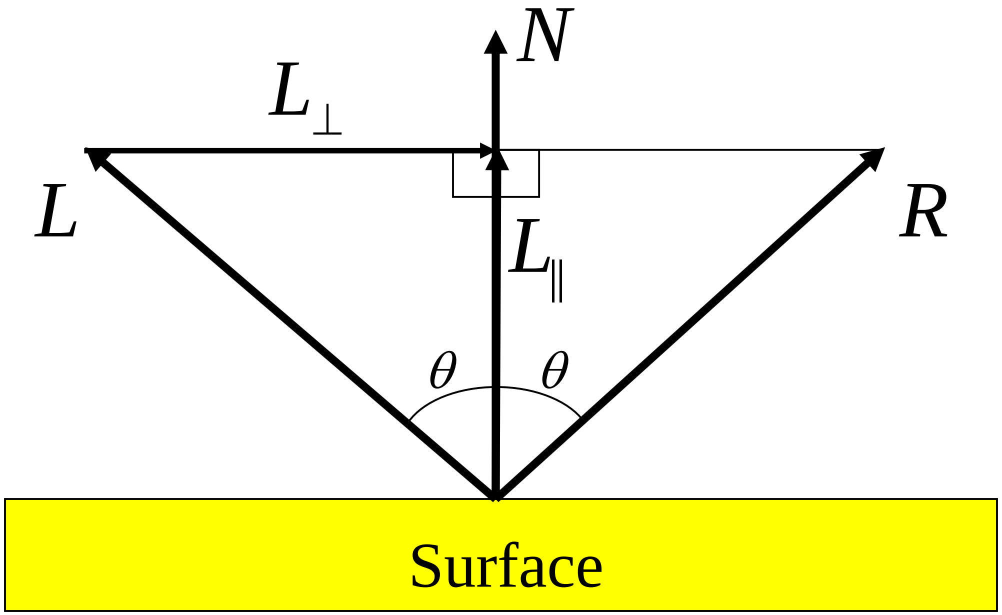
Finding the Reflected Vector

$$R = L_{\parallel} - L_{\perp}$$



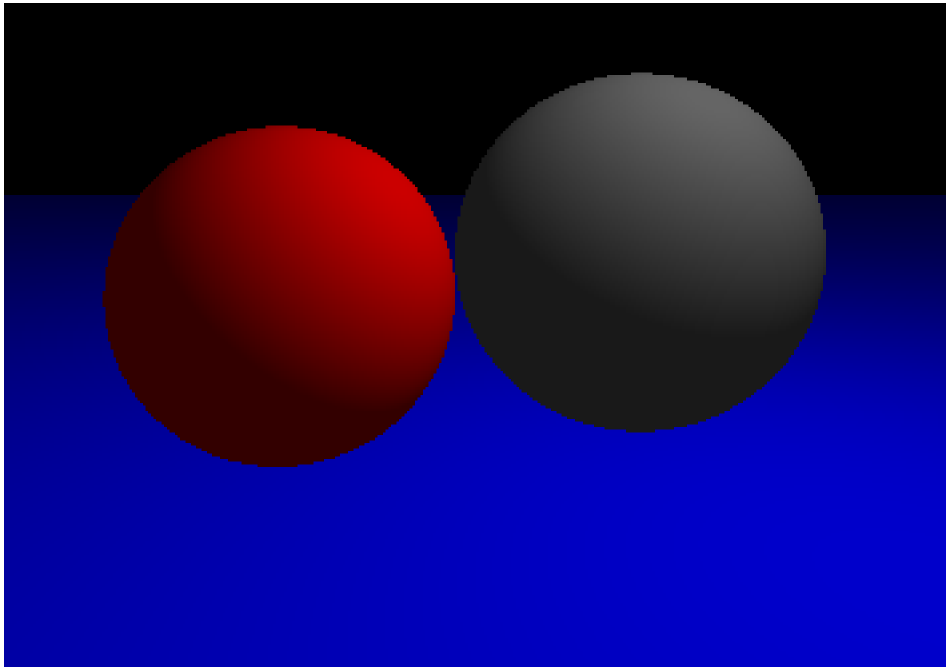
Finding the Reflected Vector

$$R = 2(L \cdot N)N - L$$



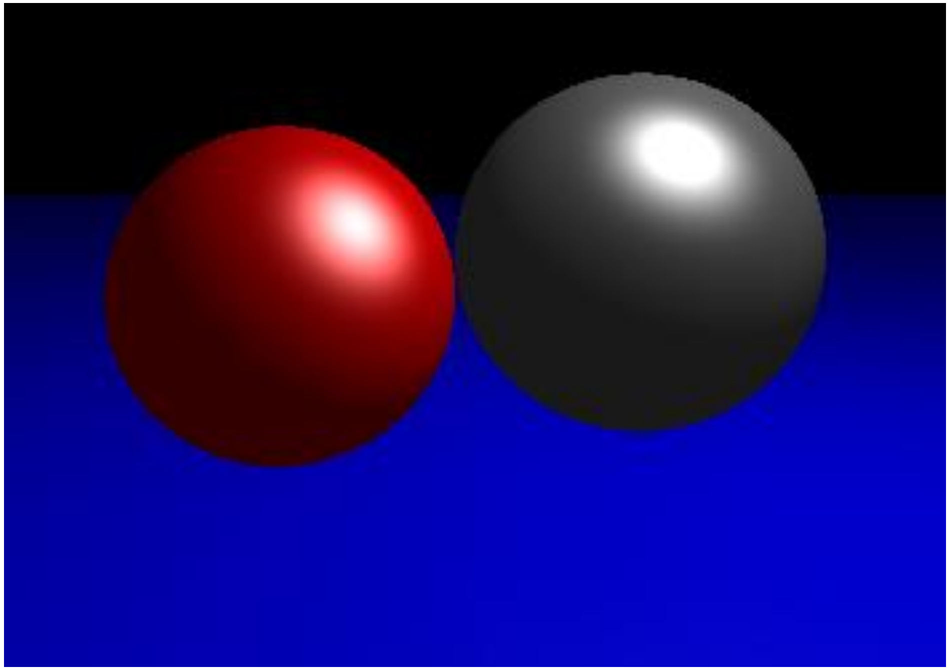
Total Illumination

$$I = k_a A + k_d C(L \cdot N)$$



Total Illumination

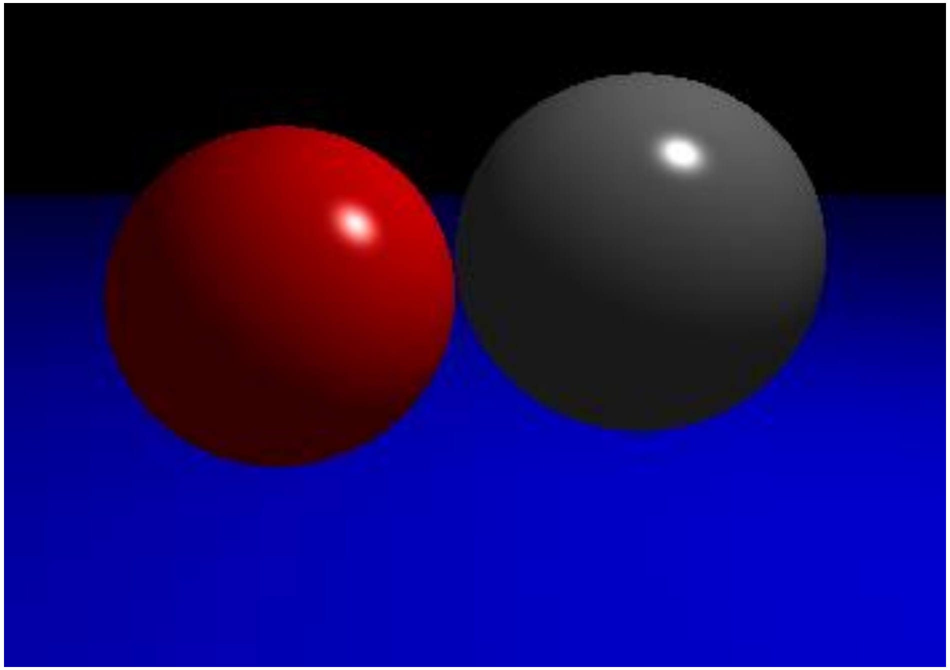
$$I = k_a A + C\left(k_d (L \cdot N) + k_s (R \cdot E)^n\right)$$



$n = 5$

Total Illumination

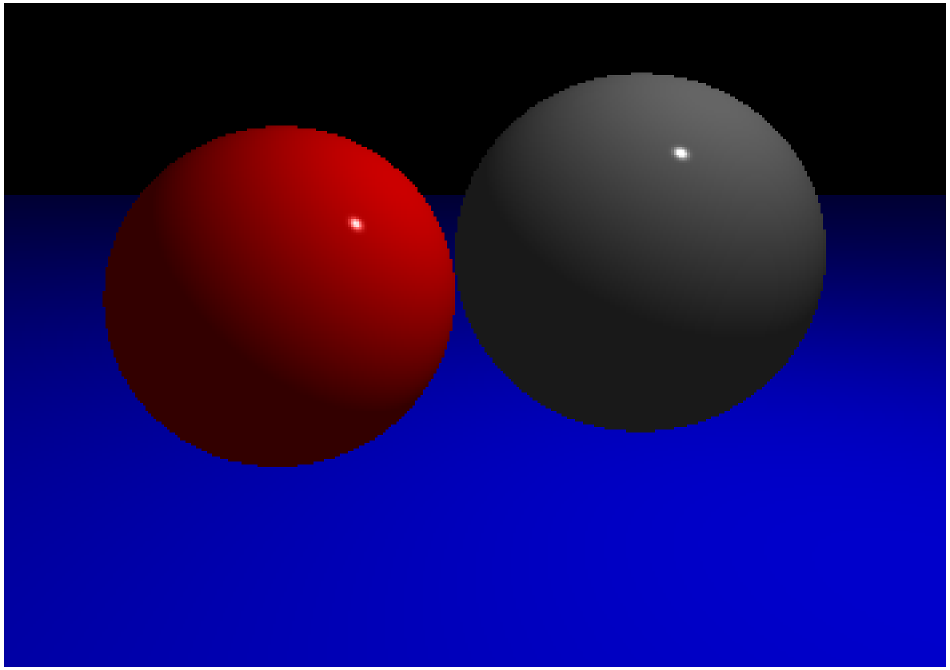
$$I = k_a A + C\left(k_d (L \cdot N) + k_s (R \cdot E)^n\right)$$



$$n = 50$$

Total Illumination

$$I = k_a A + C\left(k_d (L \cdot N) + k_s (R \cdot E)^n\right)$$



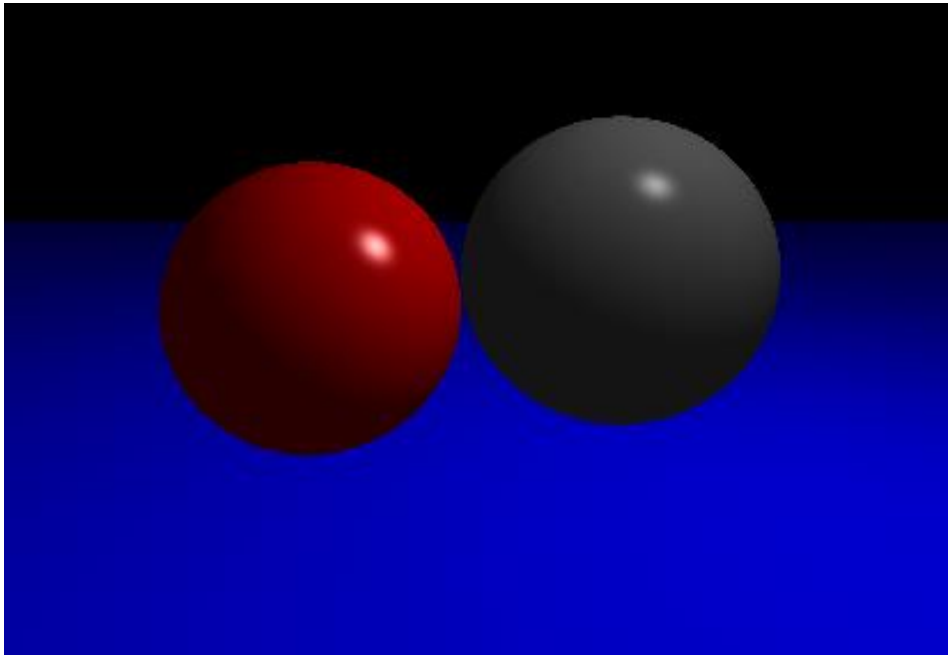
$$n = 500$$

Multiple Light Sources

- Only one ambient term no matter how many lights
- Light is additive; add contribution of multiple lights (diffuse/specular components)

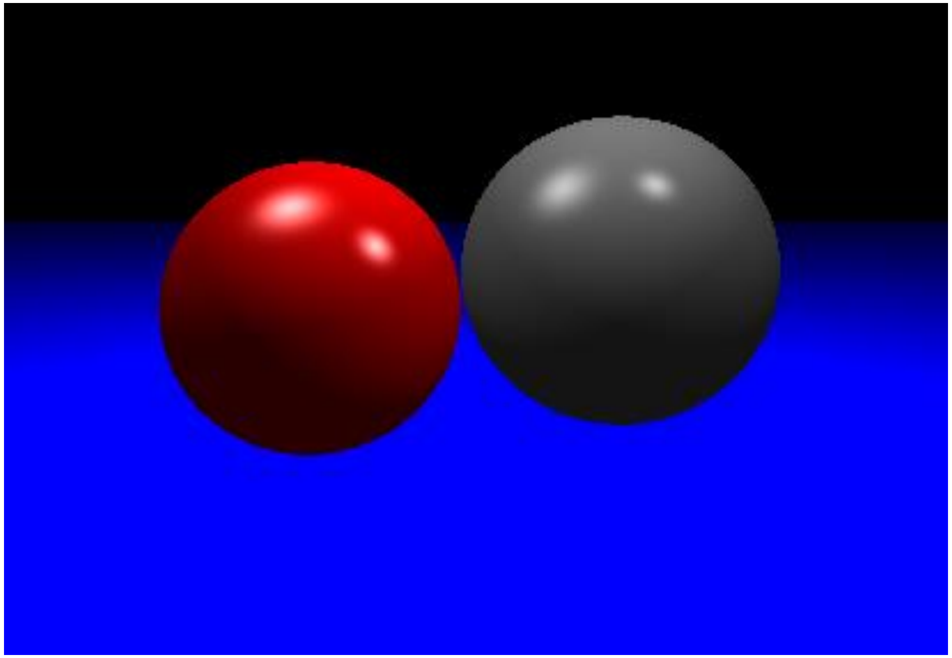
Total Illumination

$$I = k_a A + C\left(k_d (L \cdot N) + k_s (R \cdot E)^n\right)$$



Total Illumination

$$I = k_a A + \sum_i C_i \left(k_d (L_i \cdot N) + k_s (R_i \cdot E)^n \right)$$



Attenuation

- Decrease intensity with distance from light

- d = distance to light
- r = radius of attenuation for light

$$att(d,r) = \max(0, 1 - d/r)$$

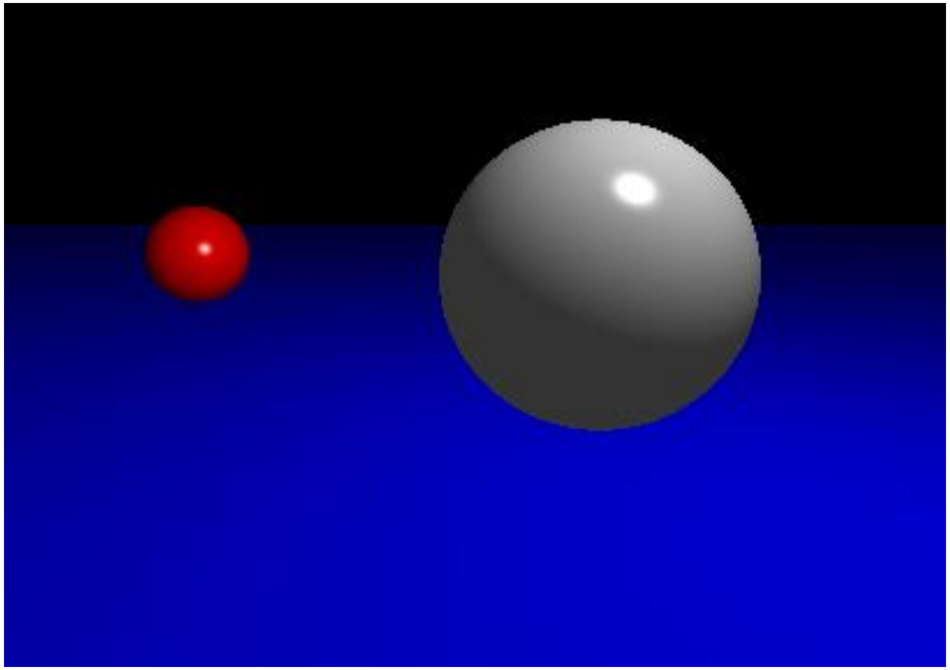
$$att(d,r) = \max(0, 1 - d^2/r^2)$$

$$att(d,r) = \max\left(0, \left(1 - d^2/r^2\right)^2\right)$$

$$att(d,r) = e^{-d^2/r^2}$$

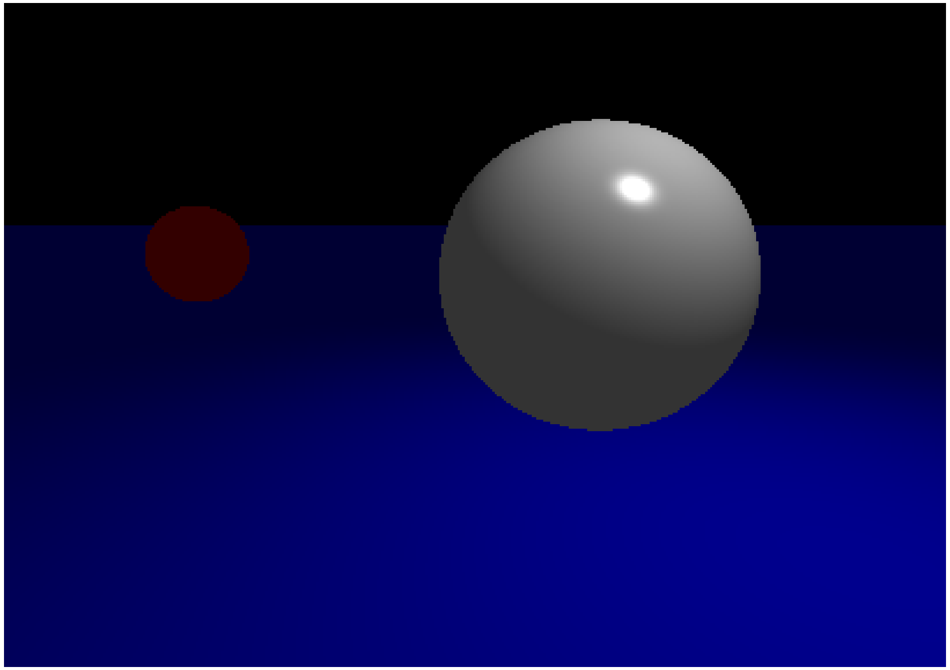
Attenuation

$$I = k_a A + \sum_i C_i \left(k_d (L_i \cdot N) + k_s (R_i \cdot E)^n \right) att(d, r_i)$$



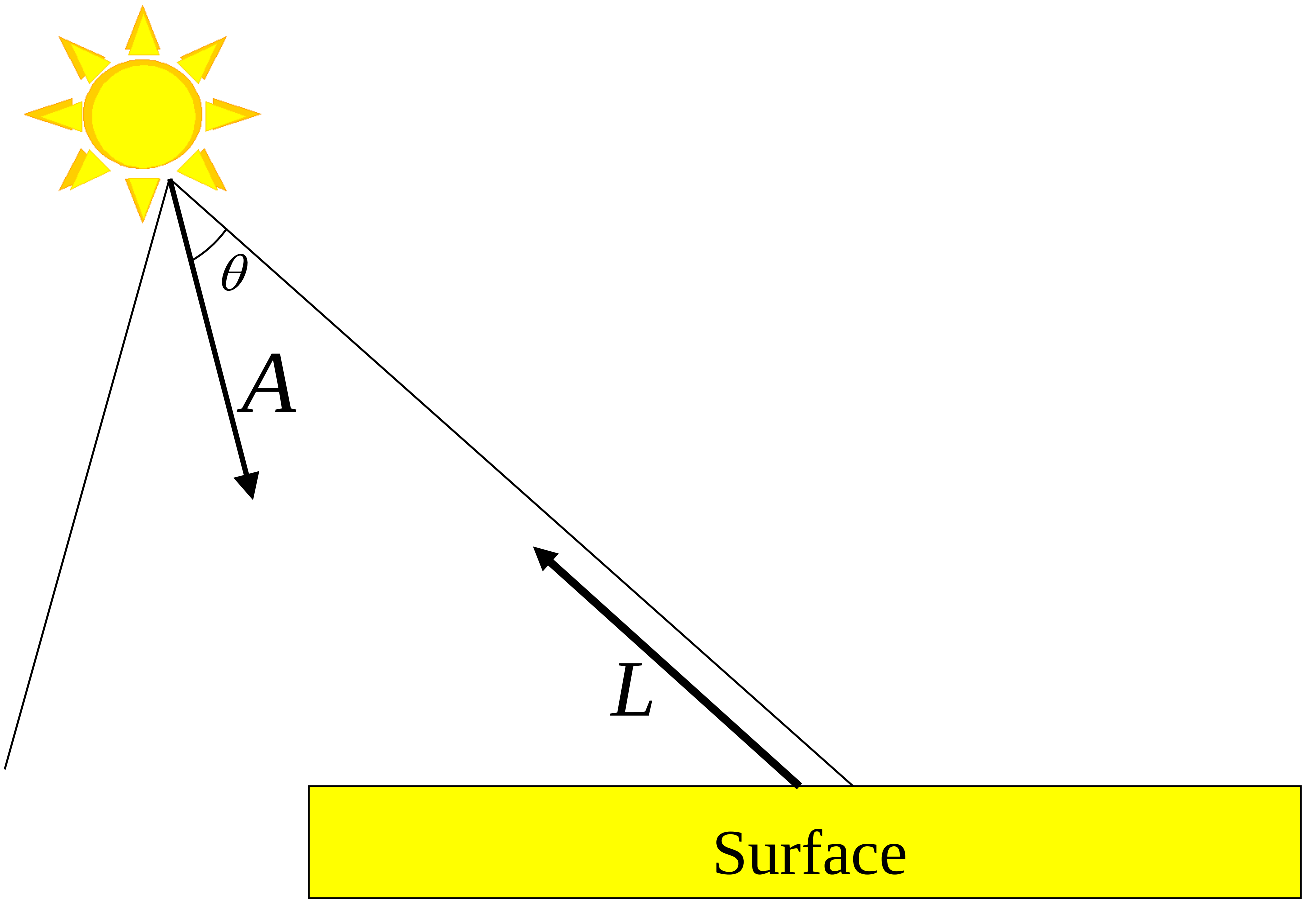
Attenuation

$$I = k_a A + \sum_i C_i \left(k_d (L_i \cdot N) + k_s (R_i \cdot E)^n \right) att(d, r_i)$$



Spot Lights

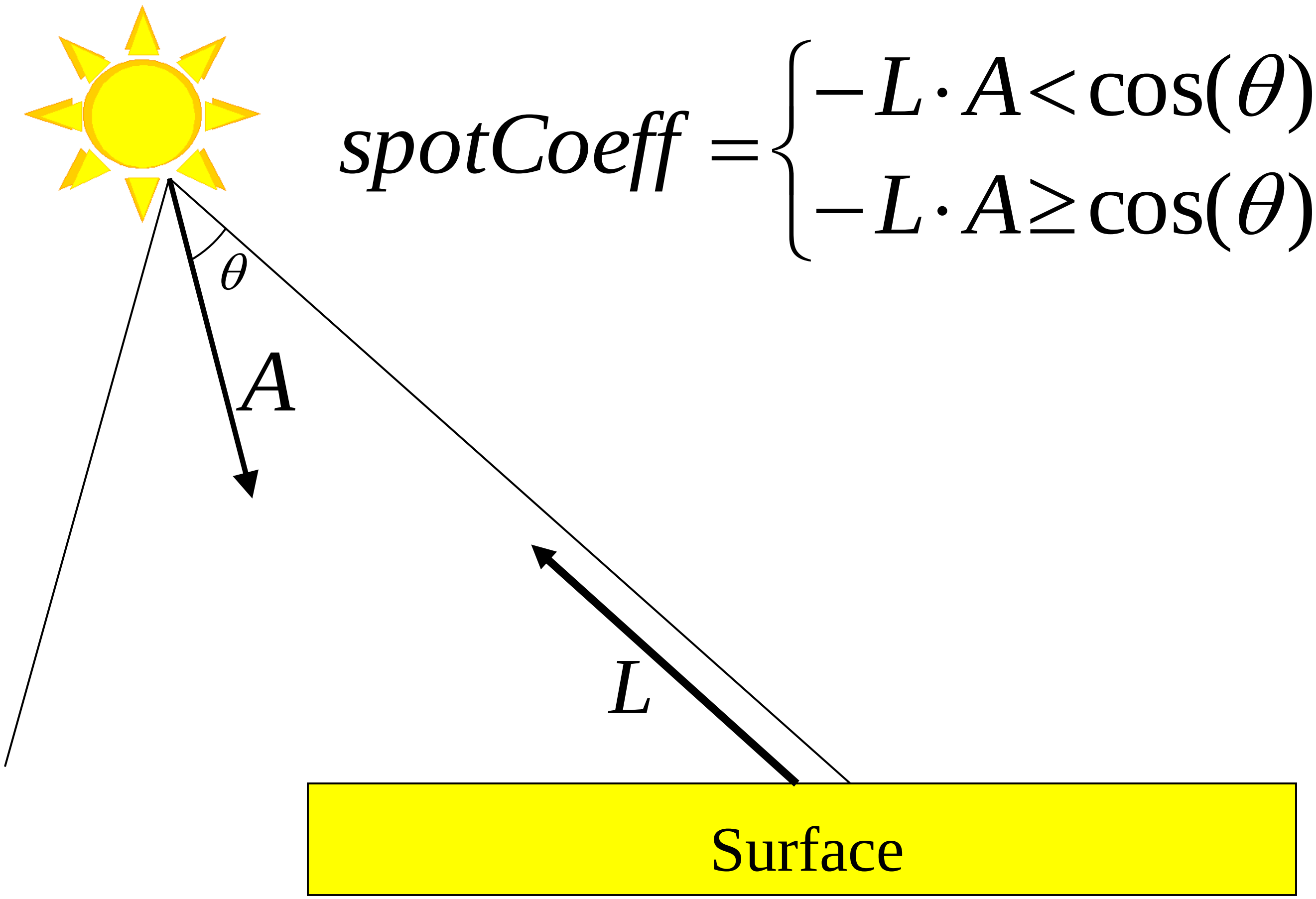
- Eliminate light contribution outside of a cone



Spot Lights

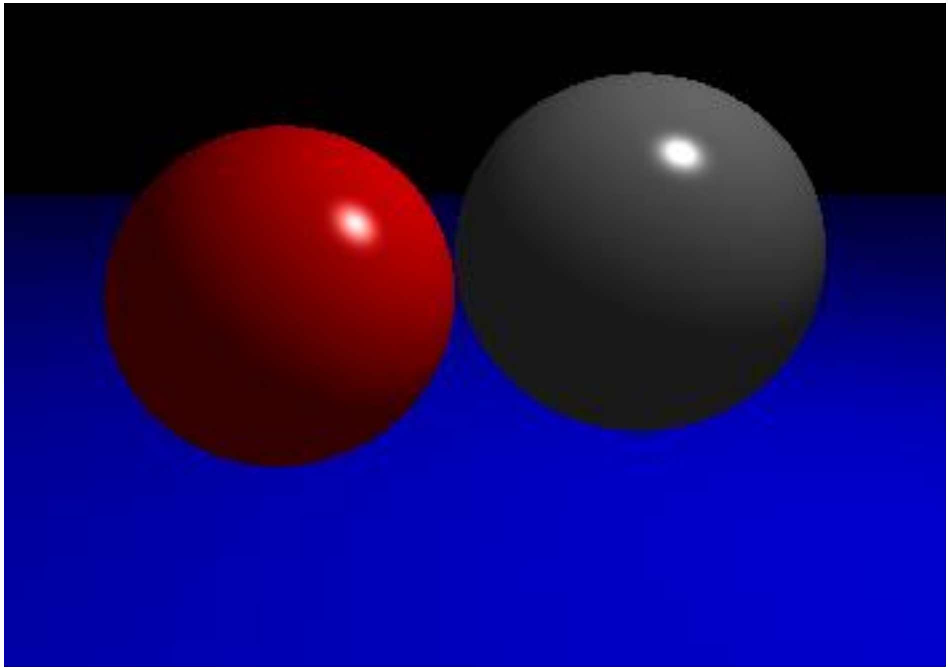
- Eliminate light contribution outside of a cone

$$spotCoeff = \begin{cases} -L \cdot A < \cos(\theta), & 0 \\ -L \cdot A \geq \cos(\theta), & (-L \cdot A)^{\alpha} \end{cases}$$



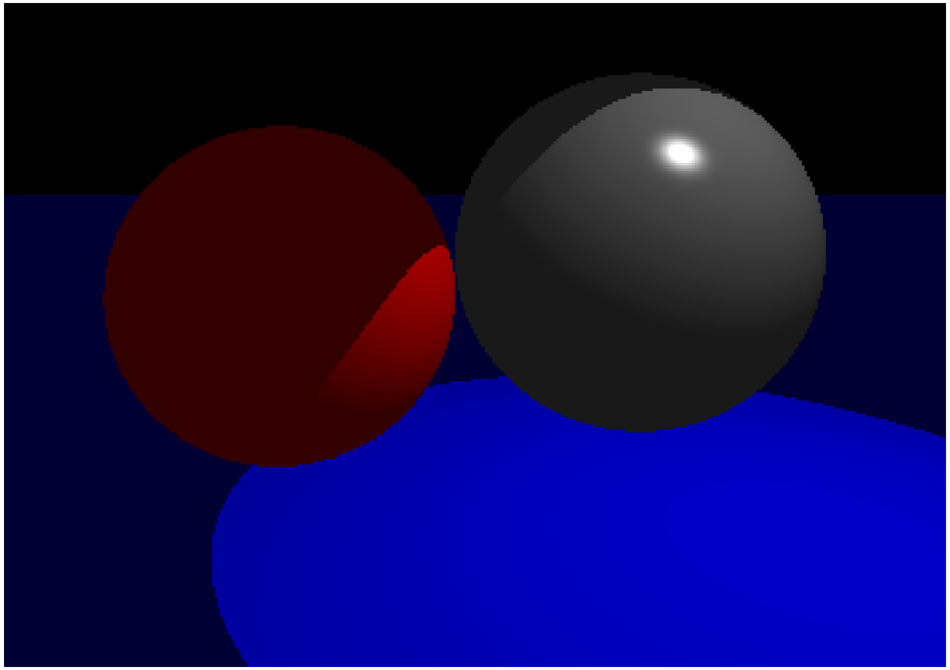
Spot Lights

$$I = k_a A + \sum_i C_i \left(k_d (L_i \cdot N) + k_s (R_i \cdot E)^n \right) spotCoeff_i$$



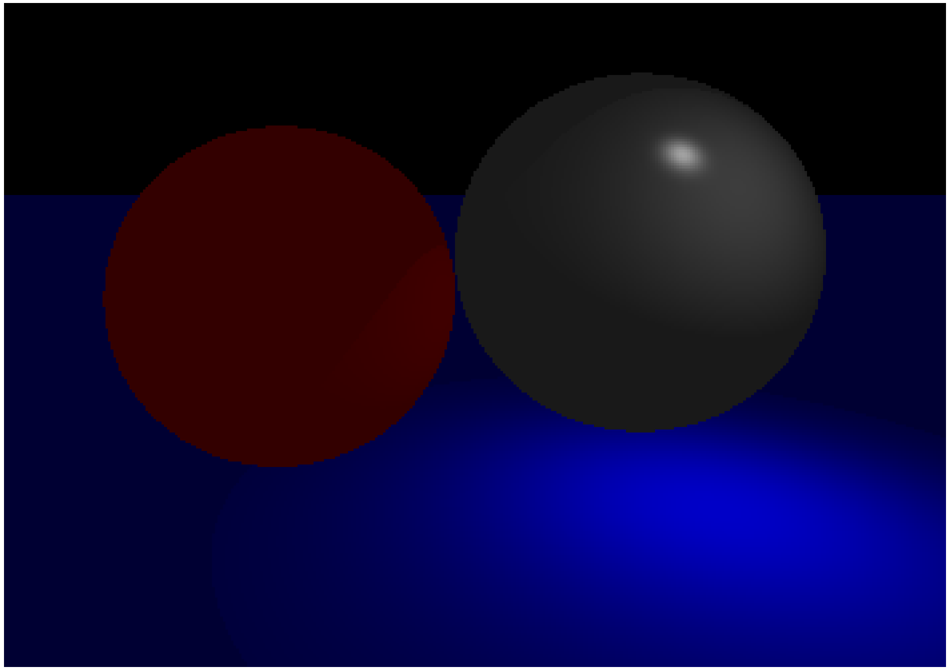
Spot Lights

$$I = k_a A + \sum_i C_i \left(k_d (L_i \cdot N) + k_s (R_i \cdot E)^n \right) spotCoeff_i$$



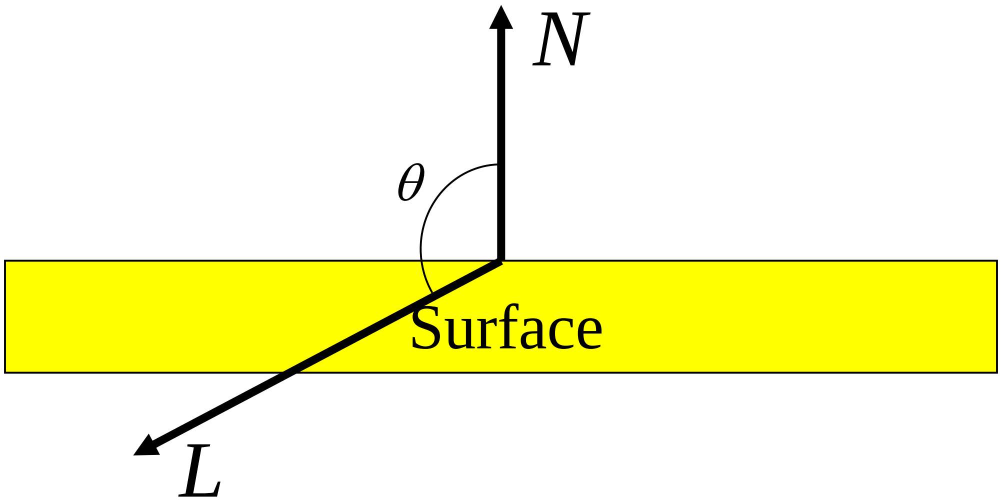
Spot Lights

$$I = k_a A + \sum_i C_i \left(k_d (L_i \cdot N) + k_s (R_i \cdot E)^n \right) spotCoeff_i$$



Implementation Considerations

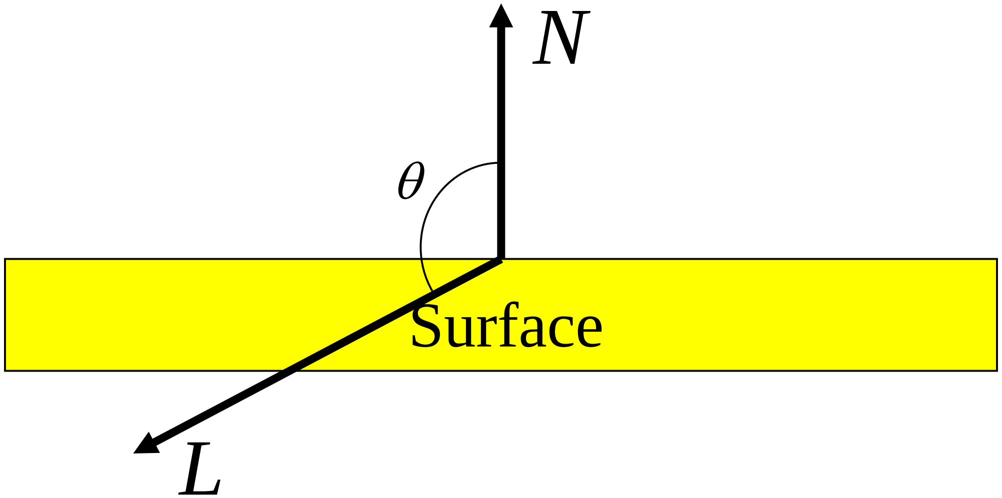
$$I = k_a A + C\left(k_d (L \cdot N) + k_s (R \cdot E)^n\right)$$



Implementation Considerations

$$I = k_a A + C\left(k_d (L \cdot N) + k_s (R \cdot E)^n\right)$$

- Two options:
  - 2-sided: negate N for back-facing polygons
  - 1-sided: if $L \cdot N \leq 0$, $I = k_a A$ // light on back of surface
    else $I = k_a A + C\left(k_d (L \cdot N) + k_s \max(0, R \cdot E)^n\right)$



Implementation Considerations

$$I = k_a A + \sum_i C_i \left(k_d (L_i \cdot N) + k_s (R_i \cdot E)^n \right)$$

- Typically choose $k_a + k_d + k_s \leq 1$
- Clamp each color component to [0,1]

OpenGL and Lighting

- Specify normals for geometry
- Create/position lights
- Specify material properties
- Select lighting model

OpenGL and Lighting

- Specify normals for geometry
- Create/position lights
- Specify material properties
- Select lighting model

OpenGL and Lighting

```
glBegin(GL_TRIANGLES);
…
glNormal3f(nx,ny,nz);
glVertex3f(x,y,z);
…
glEnd();
```

OpenGL and Lighting

- Specify normals for geometry
- Create/position lights
- Specify material properties
- Select lighting model

OpenGL and Lighting

```
float light_position[] = {0, -10, 0, 1};
float light_ambient[] = {.1, .1, .1, 1};
float light_diffuse[] = {.9, .9, .9, 1};
float light_specular[] = {1, 1, 1, 1};

glLightfv(GL_LIGHT0, GL_POSITION, light_position);
glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
glEnable(GL_LIGHT0);
glEnable(GL_LIGHTING);
```

OpenGL and Lighting

- Specify normals for geometry
- Create/position lights
- Specify material properties
- Select lighting model

OpenGL and Lighting

```
float mat_ambient[] = {1, 0, 0, 1};
float mat_diffuse[] = {1, 0, 0, 1};
float mat_specular[] = {1, 1, 1, 1};
float mat_shiny[] = {50};

glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
glMaterialfv(GL_FRONT, GL_SHININESS, mat_shiny);
```

OpenGL and Lighting

- Specify normals for geometry
- Create/position lights
- Specify material properties
- Select lighting model

OpenGL and Lighting

```
glLightModelfv(GL_LIGHT_MODEL_LOCAL_VIEWER,
    GL_TRUE);
glLightModelfv(GL_LIGHT_MODEL_TWO_SIDE,
    GL_FALSE);
```