Illumination

Courtesy of Adam Finkelstein, Princeton University
Ray Casting

```java
Image RayCast(Camera camera, Scene scene, int width, int height) {
    Image image = new Image(width, height);
    for (int i = 0; i < width; i++) {
        for (int j = 0; j < height; j++) {
            Ray ray = ConstructRayThroughPixel(camera, i, j);
            Intersection hit = FindIntersection(ray, scene);
            image[i][j] = GetColor(scene, ray, hit);
        }
    }
    return image;
}
```
Ray Casting

```java
Image RayCast(Camera camera, Scene scene, int width, int height) {
    Image image = new Image(width, height);
    for (int i = 0; i < width; i++) {
        for (int j = 0; j < height; j++) {
            Ray ray = ConstructRayThroughPixel(camera, i, j);
            Intersection hit = FindIntersection(ray, scene);
            image[i][j] = GetColor(scene, ray, hit);
        }
    }
    return image;
}
```

Without Illumination
Ray Casting

Image RayCast(Camera camera, Scene scene, int width, int height) {
    Image image = new Image(width, height);
    for (int i = 0; i < width; i++) {
        for (int j = 0; j < height; j++) {
            Ray ray = ConstructRayThroughPixel(camera, i, j);
            Intersection hit = FindIntersection(ray, scene);
            image[i][j] = GetColor(scene, ray, hit);
        }
    }
    return image;
}
Illumination

• How do we compute radiance for a sample ray?

\[ \text{image}[i][j] = \text{GetColor}(\text{scene}, \text{ray}, \text{hit}); \]
Goal

• Must derive computer models for ...
  ◦ Emission at light sources
  ◦ Scattering at surfaces
  ◦ Reception at the camera

• Desirable features ...
  ◦ Concise
  ◦ Efficient to compute
  ◦ “Accurate”
Overview

• **Direct Illumination**
  - Emission at light sources
  - Scattering at surfaces

• **Global illumination**
  - Shadows
  - Refractions
  - Inter-object reflections
Modeling Light Sources

- $I_L(x,y,z,\theta,\phi,\lambda)$ ...
  - describes the intensity of energy,
  - leaving a light source, ...
  - arriving at location $(x,y,z)$, ...
  - from direction $(\theta,\phi)$, ...
  - with wavelength $\lambda$
Empirical Models

- Ideally measure irradiant energy for “all” situations
  - Too much storage
  - Difficult in practice
OpenGL Light Source Models

- Simple mathematical models:
  - Point light
  - Directional light
  - Spot light
Point Light Source

• Models omni-directional point source
  ◦ intensity \( (I_0) \),
  ◦ position \((px, py, pz)\),
  ◦ factors \((k_c, k_l, k_q)\) for attenuation with distance \((d)\)

\[
I_L = \frac{I_0}{k_c + k_1d + k_qd^2}
\]
Directional Light Source

• Models point light source at infinity
  ○ intensity \( I_0 \),
  ○ direction \((dx, dy, dz)\)

\[ I_L = I_0 \]

No attenuation with distance
Spot Light Source

- Models point light source with direction
  - intensity ($I_0$),
  - position ($px, py, pz$),
  - direction ($dx, dy, dz$)
  - attenuation

\[
I_L = \frac{I_0 (D \bullet L)}{k_c + k_1 d + k_q d^2}
\]
Overview

• Direct Illumination
  o Emission at light sources
  o Scattering at surfaces

• Global illumination
  o Shadows
  o Refractions
  o Inter-object reflections
Modeling Surface Reflectance

- $R_s(\theta, \phi, \gamma, \psi, \lambda)$ ...  
  - describes the amount of incident energy,  
  - arriving from direction $(\theta, \phi)$, ...  
  - leaving in direction $(\gamma, \psi)$, ...  
  - with wavelength $\lambda$
Empirical Models

• Ideally measure radiant energy for “all” combinations of incident angles
  ◦ Too much storage
  ◦ Difficult in practice
OpenGL Reflectance Model

- Simple analytic model:
  - diffuse reflection +
  - specular reflection +
  - emission +
  - “ambient”

Based on model proposed by Phong
OpenGL Reflectance Model

- Simple analytic model:
  - diffuse reflection +
  - specular reflection +
  - emission +
  - “ambient”

Based on model proposed by Phong
Diffuse Reflection

• Assume surface reflects equally in all directions
  ◦ Examples: chalk, clay
Diffuse Reflection

• How much light is reflected?
  ◦ Depends on angle of incident light
Diffuse Reflection

• How much light is reflected?
  ◦ Depends on angle of incident light

\[ dL = dA \cos \Theta \]
Diffuse Reflection

- Lambertian model
  - cosine law (dot product)

\[ I_D = K_D (N \cdot L) I_L \]
OpenGL Reflectance Model

- Simple analytic model:
  - diffuse reflection +
  - specular reflection +
  - emission +
  - “ambient”
Specular Reflection

- Reflection is strongest near mirror angle
  - Examples: mirrors, metals
Specular Reflection

How much light is seen?

Depends on:
- angle of incident light
- angle to viewer
Specular Reflection

- Phong Model
  - \( \cos(\alpha)^n \)

This is a physically-motivated hack!

\[
I_S = K_S (V \cdot R)^n I_L
\]
OpenGL Reflectance Model

• Simple analytic model:
  ◦ diffuse reflection +
  ◦ specular reflection +
  ◦ emission +
  ◦ “ambient”
Emission

- Represents light emanating directly from polygon

Emission $\neq 0$
OpenGL Reflectance Model

• Simple analytic model:
  - diffuse reflection +
  - specular reflection +
  - emission +
  - “ambient”
Ambient Term

• Represents reflection of all indirect illumination

This is a total hack (avoids complexity of global illumination)!
OpenGL Reflectance Model

• Simple analytic model:
  ○ diffuse reflection +
  ○ specular reflection +
  ○ emission +
  ○ “ambient”
OpenGL Reflectance Model

- Simple analytic model:
  - diffuse reflection +
  - specular reflection +
  - emission +
  - “ambient”
OpenGL Reflectance Model

- Sum diffuse, specular, emission, and ambient

<table>
<thead>
<tr>
<th>Phong</th>
<th>$\rho_{\text{ambient}}$</th>
<th>$\rho_{\text{diffuse}}$</th>
<th>$\rho_{\text{specular}}$</th>
<th>$\rho_{\text{total}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi_i = 60^\circ$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\phi_i = 25^\circ$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\phi_i = 0^\circ$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Leonard McMillan, MIT
Surface Illumination Calculation

- Single light source:

\[ I = I_E + K_A I_{AL} + K_D (N \cdot L)I_L + K_S (V \cdot R)^n I_L \]
Surface Illumination Calculation

- Multiple light sources:

\[ I = I_E + K_A I_{AL} + \sum_i (K_D (N \cdot L_i) I_i + K_S (V \cdot R_i)^n I_i) \]
Overview

• Direct Illumination
  ◦ Emission at light sources
  ◦ Scattering at surfaces

• Global illumination
  ◦ Shadows
  ◦ Transmissions
  ◦ Inter-object reflections
Global Illumination

Greg Larson
Shadows

- Shadow term tells if light sources are blocked
  - Cast ray towards each light source \( L_i \)
  - \( S_i = 0 \) if ray is blocked, \( S_i = 1 \) otherwise

\[
I = I_E + K_A I_A + \sum_L \left( K_D (N \cdot L) + K_S (V \cdot R)^n \right) S_L I_L
\]
Ray Casting

- Trace primary rays from camera
  - Direct illumination from unblocked lights only

\[ I = I_E + K_A I_A + \sum_L \left( K_D (N \cdot L) + K_S (V \cdot R)^n \right) S_L I_L \]
Recursive Ray Tracing

• Also trace secondary rays from hit surfaces
  ◦ Global illumination from mirror reflection and transparency

\[
I = I_E + K_A I_A + \sum_L \left( K_D (N \cdot L) + K_S (V \cdot R)^n \right) S_L I_L + K_S I_R + K_T I_T
\]
Mirror reflections

- Trace secondary ray in mirror direction
  - Evaluate radiance along secondary ray and include it into illumination model

\[
I = I_E + K_A I_A + \sum_L \left( K_D (N \cdot L) + K_S (V \cdot R)'' \right) S_L L_L + K_S I_R + K_T I_T
\]
Transparency

- Trace secondary ray in direction of refraction
  - Evaluate radiance along secondary ray and include it into illumination model

\[ I = I_E + K_A I_A + \sum L \left( K_D (N \cdot L) + K_S (V \cdot R)^n \right) S_L I_L + K_S I_R + K_T I_T \]
Transparency

- Transparency coefficient is fraction transmitted
  - $K_T = 1$ for translucent object, $K_T = 0$ for opaque
  - $0 < K_T < 1$ for object that is semi-translucent

\[ I = I_E + K_A I_A + \sum_L \left( K_D (N \cdot L) + K_S (V \cdot R)^n \right) S_L I_L + K_S I_R + K_T I_T \]
Refractive Transparency

• For thin surfaces, can ignore change in direction
  ◦ Assume light travels straight through surface

\[ \eta_i - T \approx -L \]
Refractive Transparency

For solid objects, apply Snell’s law:

\[ \eta_r \sin \Theta_r = \eta_i \sin \Theta_i \]
Recursive Ray Tracing

- Ray tree represents illumination computation

\[
I = I_E + K_A I_A + \sum_L (K_D (N \cdot L) + K_S (V \cdot R)^n) S_L I_L + K_S I_R + K_T I_T
\]
Recursive Ray Tracing

- Ray tree represents illumination computation

\[ I = I_E + K_A I_A + \sum_L \left( K_D (N \cdot L) + K_S (V \cdot R)^{n} \right) S_L I_L + K_S I_R + K_T I_T \]
Recursive Ray Tracing

- GetColor calls RayTrace recursively

```java
Image RayTrace(Camera camera, Scene scene, int width, int height)
{
    Image image = new Image(width, height);
    for (int i = 0; i < width; i++) {
        for (int j = 0; j < height; j++) {
            Ray ray = ConstructRayThroughPixel(camera, i, j);
            Intersection hit = FindIntersection(ray, scene);
            image[i][j] = GetColor(scene, ray, hit);
        }
    }
    return image;
}
```
Summary

• Ray casting (direct Illumination)
  ◦ Usually use simple analytic approximations for light source emission and surface reflectance

• Recursive ray tracing (global illumination)
  ◦ Incorporate shadows, mirror reflections, and pure refractions

All of this is an approximation so that it is practical to compute

More on global illumination later!
Illumination Terminology

- **Radiant power [flux] ($\Phi$)**
  - Rate at which light energy is transmitted (in Watts).

- **Radiant Intensity (I)**
  - Power radiated onto a unit solid angle in direction (in Watts/sr)
    - e.g.: energy distribution of a light source (inverse square law)

- **Radiance (L)**
  - Radiant intensity per unit projected surface area (in Watts/m$^2$sr)
    - e.g.: light carried by a single ray (no inverse square law)

- **Irradiance (E)**
  - Incident flux density on a locally planar area (in Watts/m$^2$)
    - e.g.: light hitting a surface along a

- **Radiosity (B)**
  - Exitant flux density from a locally planar area (in Watts/m$^2$)