Illumination

Thomas Funkhouser
Princeton University
COS 426, Spring 2004

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;
}

Ray Casting

Without Illumination

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

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

Goal
Overview

- Direct Illumination
  - Emission at light sources
  - Scattering at surfaces
- Global illumination
  - Shadows
  - Refractions
  - Inter-object reflections

Empirical Models

- Ideally measure irradiant energy for "all" situations
  - Too much storage
  - Difficult in practice

Point Light Source

- Models omni-directional point source
  - intensity \( I_0 \)
  - position \((px, py, pz)\)
  - factors \(k_e, k_r, k_a\) for attenuation with distance \(d\)

\[
I_L = \frac{I_0}{k_e + k_rd + k_ud^2}
\]

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 \(\lambda\)

OpenGL Light Source Models

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

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 \( l_0 \)
  - position \((px, py, pz)\)
  - direction \((dx, dy, dz)\)
  - attenuation

\[
I_L = \frac{I_0 (D \cdot L)}{k_x + k_y d + k_z d^2}
\]

**Light Emitted from a Surface**
- Power per unit area per unit solid angle – Radiance (L)
  - Measured in \( W/m^2/\text{sr} \)
  - Projected area – perpendicular to given direction

\[
L = \frac{dA}{dA \, d\omega}
\]

**Light Falling on a Surface**
- Power per unit area – Irradiance \( E \)
  - Measured in \( W/m^2 \)
- Move surface away from light
  - Inverse square law: \( E \propto 1/r^2 \)
- Tilt surface away from light
  - Cosine law: \( E \propto n \cdot l \)

**Overview**
- Direct Illumination
  - Emission at light sources
  - Scattering at surfaces
- Global Illumination
  - Shadows
  - Refractions
  - Inter-object reflections

**Modeling Surface Reflectance**
- \( R_s(\theta, \phi, \psi, \lambda) \) ...
  - describes the amount of incident energy
  - arriving from direction \((\theta, \phi)\)
  - leaving in direction \((\psi, \lambda)\)
  - 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

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

\[ dL = dA \cos \theta \]

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!

**OpenGL Reflectance Model**

- Simple analytic model:
  - diffuse reflection +
  - specular reflection +
  - emission +
  - "ambient"

**Emission**

- Represents light emanating directly from polygon

$I_s = K_s (V \cdot R)^n I_L$
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

Surface Illumination Calculation

- Single light source:

\[ I = I_E + K_d I_{AL} + K_s (N \cdot L) I_E + K_a (V \cdot R)^* I_E \]
**Surface Illumination Calculation**

- Multiple light sources:

\[ I = I_E + K_d I_L + \sum (K_p (N \cdot L_i) I_i + K_r (V \cdot R_i) I_i) \]

**Overview**

- Direct illumination
  - Emission at light sources
  - Scattering at surfaces
- Global illumination
  - Shadows
  - Transmissions
  - Inter-object reflections

**Global Illumination**

![Image](example_image.png)

**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

\[ I = I_E + K_d I_L + \sum (K_p (N \cdot L) + K_r (V \cdot R) S_i I_i) \]

**Ray Casting (last lecture)**

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

\[ I = I_E + K_d I_L + \sum (K_p (N \cdot L) + K_r (V \cdot R) S_i I_i) \]

**Recursive Ray Tracing**

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

\[ I = I_E + K_d I_L + \sum (K_p (N \cdot L) + K_r (V \cdot R) S_i I_i) = K_i I_E + K_i I_L \]
Mirror reflections

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

\[ I = I_L + K_R I_R + \sum_i (K_{R_i}(N \cdot L) + K_{R_i}(V \cdot R_i)) S_{i} I_{i} + K_{B} I_{B} + K_{T} I_{T} \]

Transparency

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

\[ I = I_L + K_R I_R + \sum_i (K_{R_i}(N \cdot L) + K_{R_i}(V \cdot R_i)) S_{i} I_{i} + K_{B} I_{B} + K_{T} I_{T} \]

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

\[ I = I_L + K_R I_R + \sum_i (K_{R_i}(N \cdot L) + K_{R_i}(V \cdot R_i)) S_{i} I_{i} + K_{B} I_{B} + K_{T} I_{T} \]

Refractive Transparency

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

\[ T = -1 \]

Refractive Transparency

For solid objects, apply Snell’s law:

\[ \eta_i \sin \Theta_i = \eta_r \sin \Theta_r \]

\[ T = \left( \frac{\eta_r - \cos \Theta_i}{\eta_r} \right) N \cdot \hat{n} \cdot \hat{N} \]

Recursive Ray Tracing

- Ray tree represents illumination computation

\[ I = I_L + K_R I_R + \sum_i (K_{R_i}(N \cdot L) + K_{R_i}(V \cdot R_i)) S_{i} I_{i} + K_{B} I_{B} + K_{T} I_{T} \]
Recursive Ray Tracing

• Ray tree represents illumination computation

\[
I = I_e + K_r I_d + \sum_{i} (K_b(N \cdot L) + K_d(V \cdot R)S_i I_e + K_a I_d + K_I I_d)
\]

Ray traced through scene

Ray tree

Recursive Ray Tracing

• GetColor calls RayTrace recursively

```
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] (Φ)
  - 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/m²sr)
    - e.g.: light carried by a single ray (no inverse square law)

• Irradiance (E)
  - Incident flux density on a locally planar area (in Watts/m²)
    - e.g.: light hitting a surface at a point

• Radiosity (B)
  - Exitant flux density from a locally planar area (in Watts/ m²)