Ray Casting

RayCast(Camera camera, Scene scene, int width, int height)
{
    Image image = newImage(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;
}

Goal

- Must derive computer models for...
  o Emission at light sources
  o Scattering at surfaces
  o Reception at the camera

- Desirable features...
  o Concise
  o Efficient to compute
  o “Accurate”
### Overview

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

- **Global illumination**
  - Shadows
  - Refractions
  - Inter-object reflections

![Direct Illumination](image)

### Modeling Light Sources

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

![Light](image)

### Empirical Models

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

![Empirical Models](image)

### OpenGL Light Source Models

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

![OpenGL Light Source Models](image)

### Point Light Source

- Models omni-directional point source
  - intensity \(l_0\)
  - position \((px, py, pz)\)
  - factors \((k_c, k_l, k_q)\) for attenuation with distance \(d\)

\[
I_L = \frac{l_0}{k_c + k_l d + k_q d^2}
\]

![Point Light Source](image)

### Directional Light Source

- Models point light source at infinity
  - intensity \(l_0\)
  - direction \((dx, dy, dz)\)

\[
I_L = I_0
\]

![Directional Light Source](image)
Spot Light Source

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

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

Overview

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

- Global Illumination
  - Shadows
  - Refractions
  - 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
**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
  \[ \cos(\alpha)^n \]

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

OpenGL Reflectance Model

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

Emission

- Represents light emanating directly from polygon

\[ Emission \neq 0 \]

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"

Surface Illumination Calculation

- Single light source:

  \[ I = I_e + K_d I_{al} + K_s (N \cdot L) I_L + K_a (V \cdot R)^a I_d \]

Surface Illumination Calculation

- Multiple light sources:

  \[ I = I_e + K_d I_{al} + \sum (K_s (N \cdot L) I_L + K_a (V \cdot R)^a I_d) \]

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

$$I = I_L + K_r I_a + \sum_i (K_d (N \cdot L) + K_v (V \cdot R)) S_i I_i$$

Ray Casting

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

$$I = I_L + K_r I_a + \sum_i (K_d (N \cdot L) + K_v (V \cdot R')) S_i I_i$$

Recursive Ray Tracing

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

Mirror reflections

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

Transparency

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

$$I = I_L + K_r I_a + \sum_i (K_d (N \cdot L) + K_v (V \cdot R')) S_i I_i + K_r I_a + K_v I_i$$

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
Refractive Transparency

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

Refractive Transparency

- For solid objects, apply Snell’s law:
  \[ \eta_i \sin \theta_i = \eta_r \sin \theta_r \]

Recursive Ray Tracing

- Ray tree represents illumination computation

Recursive Ray Tracing

- Ray tree represents illumination computation

Recursive Ray Tracing

- GetColor calls RayTrace recursively

Recursive Ray Tracing

- GetColor calls RayTrace recursively

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/m²/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/m²)
    - e.g.: light hitting a surface along a

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