


# Illumination

Thomas Funkhouser  
Princeton University  
COS 426, Fall 2000

©Thomas Funkhouser 2000

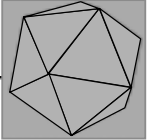


## Ray Casting

```


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;
}

```



Wireframe

©Thomas Funkhouser 2000

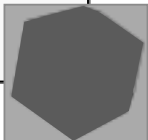


## Ray Casting

```


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;
}

```



Without Illumination

©Thomas Funkhouser 2000




## 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;
}

```



With Illumination

©Thomas Funkhouser 2000



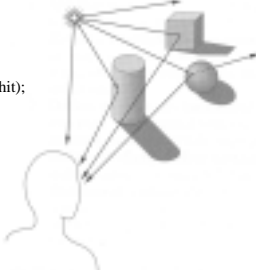
## Illumination

- How do we compute radiance for a sample ray?

```


image[i][j] = GetColor(scene, ray, hit);

```



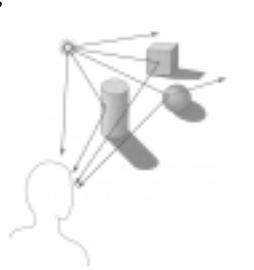
Angel Figure 6.2

©Thomas Funkhouser 2000



## Goal


- Must derive computer models for ...
  - Emission at light sources
  - Scattering at surfaces
  - Reception at the camera
- Desirable features ...
  - Concise
  - Efficient to compute
  - "Accurate"



©Thomas Funkhouser 2000

### Overview

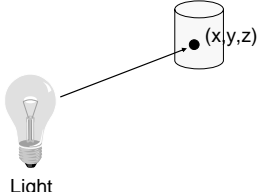
- Direct Illumination
  - Emission at light sources
  - Scattering at surfaces
- Global illumination
  - Shadows
  - Refractions
  - Inter-object reflections



Direct Illumination

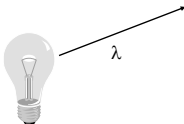
### Modeling Light Sources

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



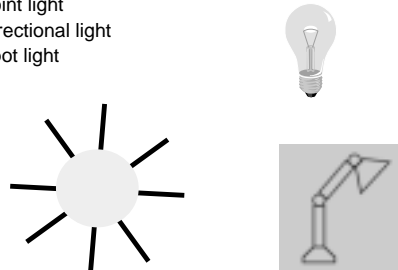
### Empirical Models

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



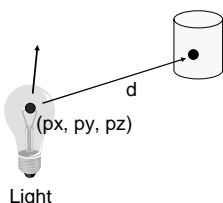
### OpenGL Light Source Models

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



### Point Light Source

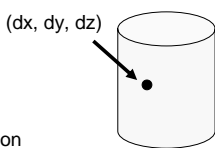
- Models omni-directional point source (e.g., bulb)
  - intensity ( $I_0$ ),
  - position  $(p_x, p_y, p_z)$ ,
  - factors  $(k_c, k_l, k_q)$  for attenuation with distance ( $d$ )



$$I_L = \frac{I_0}{k_c + k_l d + k_q d^2}$$

### Directional Light Source

- Models point light source at infinity (e.g., sun)
  - intensity ( $I_0$ ),
  - direction  $(dx, dy, dz)$



No attenuation with distance

$$I_L = I_0$$

### Spot Light Source

- Models point light source with direction (e.g., Luxo)
  - intensity ( $I_0$ ),
  - position ( $p_x, p_y, p_z$ ),
  - direction ( $dx, dy, dz$ )
  - attenuation

$$I_L = \frac{I_0(D \cdot L)}{k_c + k_l d + k_q d^2}$$

Light

### Overview

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

Direct Illumination

### 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

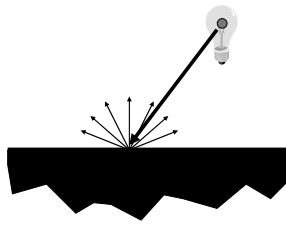
### OpenGL Reflectance Model

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

Based on model proposed by Phong

### Diffuse Reflection

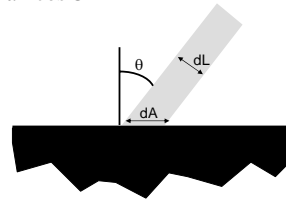
- Assume surface reflects equally in all directions
  - Examples: chalk, clay



©Thomas Fuchsauer 2000

### Diffuse Reflection

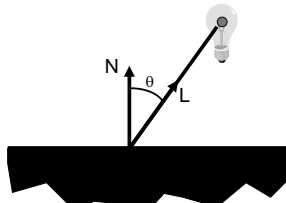
- How much light is reflected?
  - Depends on angle of incident light

$$dL = dA \cos \Theta$$


©Thomas Fuchsauer 2000

### Diffuse Reflection

- Lambertian model
  - cosine law (dot product)

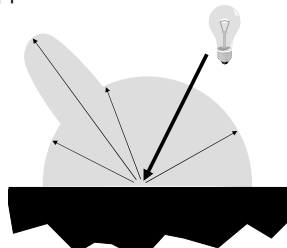


$$I_D = K_D (N \cdot L) I_L$$

©Thomas Fuchsauer 2000

### OpenGL Reflectance Model

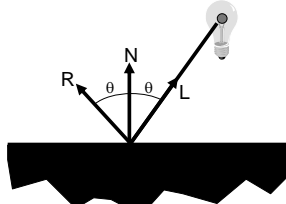
- Simple analytic model:
  - diffuse reflection +
  - specular reflection +
  - emission +
  - "ambient"



©Thomas Fuchsauer 2000

### Specular Reflection

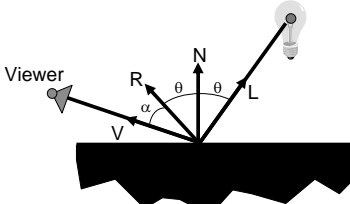
- Reflection is strongest near mirror angle
  - Examples: mirrors, metals



©Thomas Fuchsauer 2000

### Specular Reflection

- How much light is seen?
  - Depends on angle of incident light and angle to viewer



©Thomas Fuchsauer 2000

### Specular Reflection

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

$$I_S = K_S (V \cdot R)^n I_L$$

©Thomas Funkhouser 2000

### OpenGL Reflectance Model

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

©Thomas Funkhouser 2000

### Emission

- Represents light emanating directly from polygon

Emission  $\neq 0$

©Thomas Funkhouser 2000

### OpenGL Reflectance Model

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

©Thomas Funkhouser 2000

### Ambient Term

- Represents reflection of all indirect illumination

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

©Thomas Funkhouser 2000

### OpenGL Reflectance Model

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

©Thomas Funkhouser 2000

### OpenGL Reflectance Model

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

©Thomas Fuchs/2000

### OpenGL Reflectance Model

- Sum diffuse, specular, emission, and ambient

Material	Specular	Diffuse	Emission
0.0			
0.5			
1.0			

Leonard McMillan, MIT

©Thomas Fuchs/2000

### Surface Illumination Calculation

- Single light source:

$$I = I_E + K_A I_{AL} + K_D (N \cdot L) I_L + K_S (V \cdot R)^n I_L$$

©Thomas Fuchs/2000

### Surface Illumination Calculation

- Multiple light sources:

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

©Thomas Fuchs/2000

### Overview

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

Global Illumination

©Thomas Fuchs/2000

### Global Illumination

Greg Larson

©Thomas Fuchs/2000

### Shadows

- Shadow terms tell which light sources are blocked
  - Cast ray towards each light source  $L_i$
  - $S_i = 0$  if ray is blocked,  $S_i = 1$  otherwise

Shadow Term

$$I = I_E + K_A I_A + \sum_L (K_D(N \cdot L) + K_S(V \cdot R)^n) S_L I_L$$

Angel Figure 6.44

### Ray Casting

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

$$I = I_E + K_A I_A + \sum_L (K_D(N \cdot L) + K_S(V \cdot R)^n) S_L I_L$$

### Recursive Ray Tracing

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

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

### Mirror reflections

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

Radiance for mirror reflection ray

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

### Transparency

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

Radiance for refraction ray

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

### Transparency

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

Transparency Coefficient

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

### Refractive Transparency

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

$T \approx -L$

### Refractive Transparency

For solid objects, apply Snell's law:

$$\eta_r \sin \Theta_r = \eta_i \sin \Theta_i$$

$$T = \left( \frac{\eta_i}{\eta_r} \cos \Theta_i - \cos \Theta_r \right) N - \frac{\eta_i}{\eta_r} L$$

### 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

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

### Recursive Ray Tracing

- 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] ( $\Phi$ )
  - 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<sup>2</sup>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<sup>2</sup>)
    - e.g.: light hitting a surface along a
- Radiosity ( $B$ )
  - Exitant flux density from a locally planar area (in Watts/ m<sup>2</sup>)

© Thomas Frutkiner 2000