Global Illumination

CS334

Daniel G. Aliaga
Department of Computer Science
Purdue University
Recall: Lighting and Shading

• Light sources
  – Point light
    • Models an omnidirectional light source (e.g., a bulb)
  – Directional light
    • Models an omnidirectional light source at infinity
  – Spot light
    • Models a point light with direction

• Light model
  – Ambient light
  – Diffuse reflection
  – Specular reflection
Recall: Lighting and Shading

- Diffuse reflection
  - Lambertian model

\[ I_D = K_D (N \cdot L) I_L \]
Recall: Lighting and Shading

- Specular reflection
  - Phong model

\[ I_S = K_S (V \cdot R)^n I_L \]
Recall: Lighting and Shading

• Well....there is much more
For example...

- Reflection -> Bidirectional Reflectance Distribution Functions (BRDF)
- Diffuse, Specular -> Diffuse Interreflection, Specular Interreflection
- Color bleeding
- Transparency, Refraction
- Scattering
  - Subsurface scattering
  - Through participating media
- And more!
Illumination Models

• So far, you considered mostly local (direct) illumination
  – Light directly from light sources to surface
  – No shadows (actually is a global effect)

• Global (indirect) illumination: multiple bounces of light
  – Hard and soft shadows
  – Reflections/refractions (you kinda saw already)
  – Diffuse and specular interreflections
Welcome to Global Illumination

• Direct illumination + indirect illumination; e.g.
  – Direct = reflections, refractions, shadows, ...
  – Indirect = diffuse and specular inter-reflection, ...

- direct illumination
- with global illumination
- only diffuse inter-reflection
Global Illumination

- *Direct illumination + indirect illumination*; e.g.
  - Direct = reflections, refractions, shadows, ...
  - Indirect = diffuse and specular inter-reflection, ...
Reflectance Equation

• Lets start with the diffuse illumination equation and generalize...
• Define the all encompassing reflectance equation...
• Then specialize to the subset called the rendering equation...
Reflectance Equation

diffuse_illumination = 0 + I_L \cdot K_D \cdot l \cdot n
Reflectance Equation

diffuse_illumination = 0 + I_L K_D l \cdot n
Reflectance Equation

\[ L_r(x, \omega_r) = L_e(x, \omega_r) + L_i(x, \omega_i) f(x, \omega_i, \omega_r)(\omega_i \cdot n) \]

diffuse_illumination = \[ 0 \] + \[ I_L \] \[ K_D \] \[ l \cdot n \]

[Slides with help from Pat Hanrahan and Henrik Jensen]
The reflectance equation is given by:

\[ L_r(x, \omega_r) = L_e(x, \omega_r) + L_i(x, \omega_i) f(x, \omega_i, \omega_r)(\omega_i \cdot n) \]

where:
- \( L_r \) is the reflected light (Output Image)
- \( L_e \) is the emission
- \( L_i \) is the incident light (from light source)
- \( f \) is the BRDF
- \( \omega_i \) is the incident angle
- \( \omega_r \) is the reflected angle
- \( n \) is the normal vector
Reflectance Equation

\[
L_r(x, \omega_r) = L_e(x, \omega_r) + \sum L_i(x, \omega_i)f(x, \omega_i, \omega_r)(\omega_i \cdot \hat{n})
\]

Reflected Light (Output Image)  Emission  Incident Light (from light source)  BRDF  Cosine of Incident angle
Reflectance Equation

\[ L_r(x, \omega_r) = L_e(x, \omega_r) + \int_{\Omega} L_i(x, \omega_i) f(x, \omega_i, \omega_r) \cos \theta_i \, d\omega_i \]

Reflected Light (Output Image)
Emission
Incident Light (from light source)
BRDF
Cosine of Incident angle

Replace sum with integral
Reflectance Equation

\[ L_r(x, \omega_r) = L_e(x, \omega_r) + \int_{\Omega} L_i(x, \omega_i) f(x, \omega_i, \omega_r) \cos \theta_i d\omega_i \]
The Challenge

\[ L_r(x, \omega_r) = L_e(x, \omega_r) + \int_{\Omega} L_i(x, \omega_i) f(x, \omega_i, \omega_r) \cos \theta_i \, d\omega_i \]

- Computing reflectance equation requires knowing the incoming radiance from surfaces
- ...But determining incoming radiance requires knowing the reflected radiance from surfaces
In Global Illumination, the reflected light $(L_r(x, \omega_r))$ at a point $x$ is given by the emission $(L_e(x, \omega_r))$ plus the integral of the BRDF (Bidirectional Reflectance Distribution Function)

$$L_r(x, \omega_r) = L_e(x, \omega_r) + \int_{\Omega} L_r(x', -\omega_i) f(x, \omega_i, \omega_r) \cos \theta_i d\omega_i$$

- **Reflected Light (Output Image)**
- **Emission**
- **Reflected Light (from prev surface)**
- **BRDF**
- **Cosine of Incident angle**
Figure 6. A sample image. All objects are neutral grey. Color on the objects is due to caustics from the green glass balls and color bleeding from the base polygon.
Rendering Equation

$$L_r(x, \omega_r) = L_e(x, \omega_r) + \int_{\Omega} L_r(x', -\omega_i) f(x, \omega_i, \omega_r) \cos \theta_i d\omega_i$$

Reflected Light (Output Image)
Emission
Reflected Light
BRDF
Cosine of Incident angle

UNKNOWN
KNOWN
UNKNOWN
KNOWN
KNOWN
After applying to simple math and simplifications, it turns we can approximately express the above as

\[
L = E + KL
\]

L, E are vectors, 
K is the light transport matrix
Rendering as a Linear Operator...

\[ L = E + KE + K^2E + K^3E + \ldots \]

- Emission directly from light sources
- Direct Illumination on surfaces
- Global Illumination (One bounce indirect) [Mirrors, Refraction]
- (Two bounce indirect) [Caustics, etc...]
Ray Tracing

\[ L = E + KE + K^2E + K^3E + \ldots \]

- Emission directly from light sources
- Direct Illumination on surfaces
- Global Illumination (One bounce indirect) [Mirs, Refraction]
- (Two bounce indirect) [Caustics, etc…]

OpenGL Shading
Rendering Equation and Global Illumination Topics

• Local-approximations to Global Illumination
  – Diffuse/Specular
  – Ambient Occlusion

• Global Illumination Algorithms
  – Ray tracing
  – Path tracing
  – Radiosity

• Bidirectional Reflectance Distribution Functions (BRDF)
Rendering Equation and Global Illumination Topics

• Local-approximations to Global Illumination
  – Diffuse/Specular
  – Ambient Occlusion

• Global Illumination Algorithms
  – Ray tracing
  – Path tracing
  – Radiosity

• Bidirectional Reflectance Distribution Functions (BRDF)
Ambient Occlusion

• It is a lighting technique to increase the realism of a 3D scene by a “cheap” imitation of global illumination
History

• In 1998, Zhukov introduced *obscurances* in the paper “An Ambient Light Illumination Model.”
• The effect of obscurances: we just need to evaluate the *hiddenness* or occlusion of the point by considering the objects around it.
Occlusion Factor/Map

- Shooting rays outwards
- Determine the occlusion factor at p as a percentage; e.g., \( \text{occ}(p) \in [0,1] \)
Ambient Occlusion in a Phong Illumination Model

\[ I = I_a + I_d + I_s \]
\[ I_a = IA \cdot \text{occ}(p) \]

Modulate the intensity by an occlusion factor

Constant ambient intensity rendering
Inside-Looking-Out Approach: Ray Casting

• Cast rays from \( p \) in uniform pattern across the hemisphere.
• Each surface point is shaded by a ratio of ray intersections to number of original samples.
• Subtracting this ratio from 1 gives us dark areas in the occluded portions of the surface.

\[ \text{occ}(p) = \text{?} \]

e.g.: Cast 13 rays 9 intersections, so \( \text{occ}(p) = \text{?} \)
Inside-Looking-Out Approach: Ray Casting

- Cast rays from $p$ in uniform pattern across the hemisphere.
- Each surface point is shaded by a ratio of ray intersections to number of original samples.
- Subtracting this ratio from 1 gives us dark areas in the occluded portions of the surface.

e.g.: Cast 13 rays
9 intersections, so $\text{occ}(p)=4/13$;
$\Rightarrow$ Color $\times$ 4/13
Inside-Looking-Out Approach: Hardware Rendering

- Render the view at low-res from $p$ toward normal $N$
- Rasterize black geometry against a white background
- Take the (cosine-weighted) average of rasterized fragments.

11 black fragments
\[ \Rightarrow \text{Color} \times \frac{14}{25} \]
Comments

• Potentially huge pre-computation time per scene
• Stores occlusion factor as vertex attributes
  – Thus needs a dense sampling of vertices
• Variations on sampling method
  – “Inside-out” algorithm
  – “outside-in” alternative (not explained)
Outside-Looking-In Approach

• What would you do?
Outside-Looking-In: One option is [Sattler et. al 2004]

\[ c_i = \sum_{j=1}^{k} M_{ij} I_j \]
enable orthographic projection
disable framebuffer
for all light directions \( j \) do
    set camera at light direction \( l_j \)
    render object into depth buffer with polygon offset
for all vertices \( i \) do
    begin query \( i \)
    render vertex \( i \)
    end query \( i \)
end for
for all vertices \( i \) do
    retrieve result from query \( i \)
    if result is "visible" then
        \[ M_{ij} = n_i \cdot l_j \]
    end if
end for
end for

\[
M_{ij} = \begin{cases} 
    n_i \cdot l_j & : \text{vertex visible} \\
    0 & : \text{vertex invisible}
\end{cases}
\]

\[
c_i = \sum_{j=1}^{k} M_{ij} I_j
\]
[Sattler et al. 2004]

- For each light on the light sphere
- Take the depth map (for occlusion query)
- Use occlusion query to determine the visibility matrix
Another option: Screen-Based AO

Screen-Based AO
Screen-Based AO

• What would you do?
Rendering Equation and Global Illumination Topics

• Local-approximations to Global Illumination
  – Diffuse/Specular
  – Ambient Occlusion

• Global Illumination Algorithms
  – Ray tracing
  – Path tracing
    – Radiosity

• Bidirectional Reflectance Distribution Functions (BRDF)
Radiosity

- Radiosity, inspired by ideas from heat transfer, is an application of a finite element method to solving the rendering equation for scenes with purely diffuse surfaces.

- The main idea of the method is to store illumination values on the surfaces of the objects, as the light is propagated starting at the light sources.

[Radiosity slides heavily based on Dr. Mario Costa Sousa, Dept. of of CS, U. Of Calgary]
Radiosity

- Calculating the overall light propagation within a scene, for short **global illumination** is a very difficult problem.

- With a standard ray tracing algorithm, this is a very time consuming task, since a huge number of rays have to be shot.
Radiosity (Computer Graphics)

• **Assumption #1:** surfaces are diffuse emitters and reflectors of energy, emitting and reflecting energy uniformly over their entire area.

• **Assumption #2:** an equilibrium solution can be reached; that all of the energy in an environment is accounted for, through absorption and reflection.

• Also **viewpoint independent:** the solution will be the same regardless of the viewpoint of the image.
Radiosity

• Equation:

\[ B_i = E_i + \rho_i \sum B_j F_{ij} \]
Ray Tracing
The Radiosity Equation

\[ B_i = E_i + \rho_i \sum B_j F_{ij} \]

- **Radiosity of surface** \( i \)
- **Emissivity of surface** \( i \)
- **Reflectivity of surface** \( i \)
- **Radiosity of surface** \( j \)
- **Form Factor** of surface \( j \) relative to surface \( i \)

accounts for the physical relationship between the two surfaces

will absorb a certain percentage of light energy which strikes the surface
The Radiosity Equation

\[ B_i = E_i + \rho_i \sum B_j F_{ij} \]

Energy emitted by surface i
The Radiosity Equation

\[ B_i = E_i + \rho_i \sum B_j F_{ij} \]

Energy reaching surface \( i \) from other surfaces
The Radiosity Equation

\[ B_i = E_i + \rho_i \sum B_j F_{ij} \]

Energy reflected by surface i
Classic Radiosity Algorithm

1. Mesh Surfaces into Elements
2. Compute Form Factors Between Elements
3. Solve Linear System for Radiosities
4. Reconstruct and Display Solution
Radiosity Matrix

\[ B_i = E_i + \rho_i \sum_{j=1}^{n} F_{ij} B_j \]

What is the matrix form? (like “Ax=b”)

\[ B_i - \rho_i \sum_{j=1}^{n} F_{ij} B_j = E_i \]
Radiosity Matrix

\[
B_i = E_i + \rho_i \sum_{j=1}^{n} F_{ij} B_j
\]

\[
B_i - \rho_i \sum_{j=1}^{n} F_{ij} B_j = E_i
\]

\[
\begin{bmatrix}
1-\rho_1 F_{11} & -\rho_1 F_{12} & \cdots & -\rho_1 F_{1n} \\
-\rho_2 F_{21} & 1-\rho_2 F_{22} & \cdots & -\rho_2 F_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
-\rho_n F_{n1} & -\rho_n F_{n2} & \cdots & 1-\rho_n F_{nn}
\end{bmatrix}
\begin{bmatrix}
B_1 \\
B_2 \\
\vdots \\
B_n
\end{bmatrix}
= 
\begin{bmatrix}
E_1 \\
E_2 \\
\vdots \\
E_n
\end{bmatrix}
\]
Radiosity Matrix

- The "full matrix" radiosity solution calculates the form factors between each pair of surfaces in the environment, then forms a series of simultaneous linear equations.

\[
\begin{bmatrix}
1 - \rho_1 F_{11} & -\rho_1 F_{12} & \cdots & -\rho_1 F_{1n} \\
-\rho_2 F_{21} & 1 - \rho_2 F_{22} & \cdots & -\rho_2 F_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
-\rho_n F_{n1} & -\rho_n F_{n2} & \cdots & 1 - \rho_n F_{nn}
\end{bmatrix}
\begin{bmatrix}
B_1 \\
B_2 \\
\vdots \\
B_n
\end{bmatrix}
= 
\begin{bmatrix}
E_1 \\
E_2 \\
\vdots \\
E_n
\end{bmatrix}
\]

- This matrix equation is solved for the "B" values, which can be used as the final intensity (or color) value of each surface.
Artifacts

A. Blocky shadows
B. Missing features
C. Mach bands
D. Inappropriate shading discontinuities
E. Unresolved discontinuities
What can you do?
Increase Resolution
Adaptively Mesh
e.g., Discontinuity Meshing
More examples...
Rendering Equation and Global Illumination Topics

• Local-approximations to Global Illumination
  – Diffuse/Specular
  – Ambient Occlusion

• Global Illumination Algorithms
  – Ray tracing
  – Path tracing
  – Radiosity

• Bidirectional Reflectance Distribution Functions (BRDF)
Measuring BRDFs

- BRDF is 4-dimensional, though simpler measurements (0D/1D/2D/3D) are often useful
Measuring Reflectance

$0^\circ/45^\circ$
Diffuse Measurement

$45^\circ/45^\circ$
Specular Measurement
Gloss Measurements

- “Haze” is the width of a specular peak
BRDF Measurements

- Next step up: measure over a 1- or 2-D space
Gonioreflectometers

- Or a 4D space
Image-Based BRDF Measurement

• A camera acquires with each picture a 2D image of sampled measurements
  – Requires mapping light angles to camera pixels
Ward’s BRDF Measurement Setup

Half-silvered dome

Light source

Camera

Sample
Ward’s BRDF Measurement Setup

- Each picture captures light from a hemisphere of angles
Measurement

- 20-80 million reflectance measurements per material
- Each tabulated BRDF entails $90 \times 90 \times 180 \times 3 = 4,374,000$ measurement bins