### Lecture 2: Radiosity

**Fall 2004**  
**Kavita Bala**  
Computer Science  
Cornell University

### Information

- **Office Hours**  
  - Wed: 2-3 Upson 5142

- **Web-page**  
  - www.cs.cornell.edu/courses/cs665/2004fa/  
  - Tentative schedule  
  - Homeworks, lecture notes, will be on-line  
  - Check for updates and announcements

### Classic Ray Tracing

- **Image-based**

- **Gathering approach**  
  - from the light sources (direct illumination)  
  - from the reflected direction (perfect specular)  
  - from the refracted direction (perfect specular)

- **All other contributions are ignored!**  
  - Not a complete solution

### Whitted RT Assumptions

- **Light Source:** point light source  
  - Hard shadows  
  - Single shadow ray direction

- **Material:** Blinn-Phong model  
  - Diffuse with specular peak

- **Light Propagation**  
  - Occluding objects  
  - Specular interreflections only  
    - trace rays in mirror reflection direction only

### Other approaches

- **Classic ray tracing:**  
  - Only perfect specular and perfect refraction/reflection  
  - View-dependent

- **Radiosity (1984)**  
  - Pure diffuse  
  - View-independent

- **Monte Carlo Ray Tracing (1986)**  
  - Global Illumination for any environment

### Radiosity Advantages

- Physically based approach for diffuse environments

- Can model diffuse interactions, color bleeding, indirect lighting and penumbra (area light sources)

- Boundary element (finite element) problem

- Accounts for very high percentage of total energy transfer
Key Idea #1: diffuse only

- Radiance independent of direction
- Surface looks the same from any viewpoint
- No specular reflections

Diffuse surfaces

- Diffuse emitter
  \[ L(x \to \Theta) = \text{constant over } \Theta \]
- Diffuse reflector
  Reflectivity constant

Key Idea #2: “constant”polygons

- Radiosity solution is an approximation, due to discretization of scene into patches
- Subdivide scene into small polygons

Constant radiance approximation

- Radiance is constant over a surface element
  \[ L(x) = \text{constant over } x \]
- surface element \( i \): \[ L(x) = L_i \]

Radiosity Equation

\[ \text{Emitted radiosity} = \text{self-emitted radiosity} + \text{received & reflected radiosity} \]

\[ \text{Radiosity}_i = \text{Radiosity}_{\text{self}, i} + \sum_{j=1}^{N} a_{ij} \text{Radiosity}_j \]

Radiosity Equation

- Radiosity equation for each polygon \( i \)
  \[ \text{Radiosity}_i = \text{Radiosity}_{\text{self}, i} + \sum_{j=1}^{N} a_{ij} \text{Radiosity}_j \]
  ...  
  \[ \text{Radiosity}_n = \text{Radiosity}_{\text{self}, n} + \sum_{j=1}^{N} a_{jn} \text{Radiosity}_j \]
- \( N \) equations; \( N \) unknown variables
Radiosity algorithm

• Subdivide the scene into small polygons
• Compute for each polygon a constant illumination value
• Choose a viewpoint, and display the visible polygons

Radiosity algorithm

• Subdivide the scene into small polygons
• Compute for each polygon a constant illumination value
• Choose a viewpoint, and display the visible polygons
• Choose a new viewpoint ....
• ... and another ....
• ... and another ....

Radiosity: Typical Image

Energy Conservation Equation

Energy Conservation Equation

Compute Form Factors

Form Factors Invariant

\[ F(j \rightarrow i) = \frac{1}{A_i} \left( \frac{\cos \theta_i \cdot \cos \theta_j}{r_{ij}} \right) \int V(x,y) \, dA_j \cdot dA_i \]

\[ F(i \rightarrow j) = \frac{1}{A_j} \left( \frac{\cos \theta_i \cdot \cos \theta_j}{r_{ij}} \right) \int V(x,y) \, dA_i \cdot dA_j \]

\[ F(j \rightarrow i) \cdot A_i = F(j \rightarrow i) \cdot A_j \]

\[ \Phi_i = \Phi_{e,i} + \sum_{j=1}^{N} \Phi_j \cdot F(j \rightarrow i) \]
Radiosity Equation

- Radiosity for each polygon $i$
  \[ B_i = B_{e,i} + \rho \sum_{j} B_j F(i \rightarrow j) \]

- Linear system
  - $B_i$: radiosity of patch $i$ (unknown)
  - $B_{e,i}$: emission of patch $i$ (known)
  - $\rho$: reflectivity of patch $i$ (known)
  - $F(i \rightarrow j)$: form-factor (coefficients of matrix)

Linear System

\[
\begin{bmatrix}
1 - \rho F_{e,1} & -\rho F_{e,2} & \cdots & -\rho F_{e,n} & \mathbf{B}_1 \\
-\rho F_{e,1} & 1 - \rho F_{e,2} & \cdots & -\rho F_{e,n} & \mathbf{B}_2 \\
\vdots & \vdots & \ddots & \vdots & \mathbf{B}_n \\
-\rho F_{e,1} & -\rho F_{e,2} & \cdots & 1 - \rho F_{e,n} & \mathbf{B}_n
\end{bmatrix}
\begin{bmatrix}
B_1 \\
B_2 \\
\vdots \\
B_n
\end{bmatrix}
= \begin{bmatrix}
B_{e,1} \\
B_{e,2} \\
\vdots \\
B_{e,n}
\end{bmatrix}
\]

Radiosity Algorithm

- Subdivide scene in polygons
  - mesh that determines final solution
- Compute Form Factors
  - transfer of energy between polygons
- Solve linear system
  - results in power (color) per polygon
- Pick a viewpoint and display
  - loop

Form Factor

- $F_{i \rightarrow j}$: the fraction of power emitted by $j$, which is received by $i$
- **Area**
  - if $i$ is smaller, it receives less power
- **Orientation**
  - if $i$ faces $j$, it receives more power
- **Distance**
  - if $i$ is further away, it receives less power

Form Factors - how to compute?

- Closed Form
  - Analytic
- Hemicube
- Monte-Carlo
Form Factor – Analytical

\[ F(j \rightarrow i) = \frac{1}{A_i} \int \frac{\cos \theta_i \cdot \cos \theta_j}{\pi \cdot r_{ij}^2} \cdot V(x, y) \cdot dA_j \cdot dA_i \]

- Equations for special cases (polygons)
- In general hard problem
- Visibility makes it harder

Form Factors - Hemicube

- Project patch on hemicube
- Add hemicube cells to compute form factor

Form Factors - Hemicube

- Depth information per pixel evaluates visibility
- FFs for all polygons in scene
- Hardware rendering (Z-buffer)
- Severe aliasing: Small polygons “disappear”

FF - Monte Carlo

\[ \langle F(j \rightarrow i) \rangle = \frac{1}{N \cdot A_i} \sum \frac{\cos \theta_i \cdot \cos \theta_j}{\pi \cdot r_{ij}^2} \cdot V(x, y) \]

- Generate point on patch i
- Generate point on patch j
- Evaluate integrand
- Compute average

Form Factors

- Visibility checks are most expensive operation
- FFs are usually computed when needed
  - computationally expensive
  - memory \( O(N^2) \)

Radiosity Algorithm

- Subdivide scene in polygons
  - mesh that determines final solution
- Compute Form Factors
  - transfer of energy between polygons
- Solve linear system
  - results in power (color) per polygon
- Pick a viewpoint and display
  - loop
How To Solve Linear System

• Matrix Inversion

• Gathering methods
  – Jacobi iteration
  – Gauss-Seidel

• Shooting
  – Southwell iteration
  – Improved Southwell iteration

Matrix Inversion

\[
\begin{bmatrix}
B_1 \\
B_2 \\
\vdots \\
B_N
\end{bmatrix} =
\begin{bmatrix}
1 - \rho_i F_{i,e} & -\rho_i F_{i,e} & \ldots & -\rho_i F_{i,e} \\
-\rho_i F_{i,e} & 1 - \rho_i F_{i,e} & \ldots & -\rho_i F_{i,e} \\
\vdots & \vdots & \ddots & \vdots \\
-\rho_i F_{i,e} & -\rho_i F_{i,e} & \ldots & 1 - \rho_i F_{i,e}
\end{bmatrix}
\begin{bmatrix}
B_{1,e} \\
B_{2,e} \\
\vdots \\
B_{N,e}
\end{bmatrix}
\]

\(O(n^3)\)

Iterative approaches

• Jacobi iteration
  • Start with initial guess for energy distribution (light sources)
  • Update radiosity/power of all patches based on the previous guess

\[
B_i = B_{i,j} + \rho_i \sum_{j=1}^{N} B_j F(i \to j)
\]

new value old values

• Repeat until converged

Jacobi

• For all patches \(i = 1 \ldots N\) : \(B_i^{(0)} = B_{e,i}\)

• while not converged:
  – for all patches \(i = 1 \ldots N\)

\[
B_i^{(\ell)} = B_{i,j} + \rho_i \sum_{j=1}^{N} B_j^{(\ell-1)} F(i \to j)
\]

update of 1 patch requires evaluation of \(N\) FFs

Improved Gathering

• Jacobi iteration only uses values of previous iterations to compute new values

• Gauss-Seidel iteration
  – New values used immediately
  – Slightly better convergence
Gauss-Seidel

- For all patches i (i=1...N) : $B_i^{(0)} = B_{e,i}$
- while not converged:
  - for all patches i (i=1...N)

  $$B_i^{(g)} = B_{ij} + \rho \sum_{j=1}^{i-1} B_j^{(g)} F(i \rightarrow j) + \rho \sum_{j=i}^{N} B_j^{(g-1)} F(i \rightarrow j)$$

Example

Progressive Radiosity

- Gathering: $O(n^2)$/iteration
  - Still too slow
- Can use “shooting” as opposed to “gathering” approach
- 1-2% of all emitting and reflecting surfaces can account for very high percentage of energy

Southwell Iteration

- “Shooting” method
- Start with initial guess for light distribution (light sources)
- Select patch and distribute its energy over all polygons

Southwell Iteration (Wrong)

- For all patches i (i=1..N) :
  - $B_i^{(0)} = B_{e,i}$
- while not converged:
  - select shooting patch k with $B_k^{(g-1)} \neq 0$
  - for all patches i (i=1..N)

  with n FF evaluations, n patches are updated!

Southwell Iteration

- Keep record of “unshot” radiosity/energy per patch
- Repeat shooting of unshot energy until converged
**Southwell Iteration (Correct)**

- For all patches $i$ ($i=1..N$):
  - $B_i(0) = B_{e,i}$
  - $\Delta B_i(0) = B_{e,i}$
- while not converged:
  - select shooting patch $k$ with $\Delta B_k(g-1) \neq 0$
  - for all patches $i$ ($i=1..N$)
    - $\Delta B_k(g) = 0$

  with $n$ FF evaluations, $n$ patches are updated!

**Progressive Radiosity**

- Solution time is fast: $O(n)$ for first results
- Can monotonically approach the complete diffuse radiosity solution

**Progressive Refinement**

- Southwell selects shooting patches in no particular order
- Progressive refinement radiosity selects patch with largest unshot energy
- First image is generated fairly quickly!

**PR + Ambient term**

- PR gives an estimate for each radiosity value that is smaller than the real value
- Estimate can be improved by using ambient term
  - Add all unshot energy
  - Distribute total unshot energy equally over all patches
- Solution has improved energy distribution

**Gathering vs. Shooting**

- Gathering
  - Jacobi
  - Gauss-Seidel

- Shooting
  - Southwell
  - Progressive Radiosity

**Comparison**

- Gauss-Seidel
- Southwell + sorting
- Southwell + ambient
Radiosity Algorithm

- Subdivide scene in polygons
  - mesh that determines final solution
- Compute Form Factors
  - transfer of energy between polygons
- Solve linear system
  - results in power (color) per polygon
- Pick a viewpoint and display
  - Loop over different viewpoints