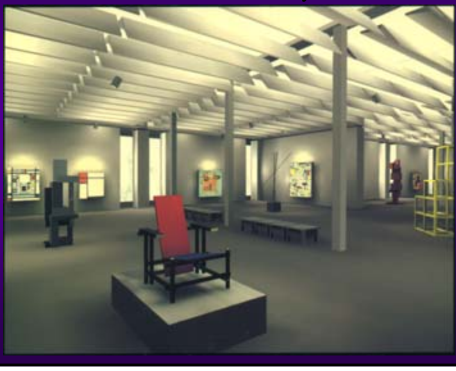
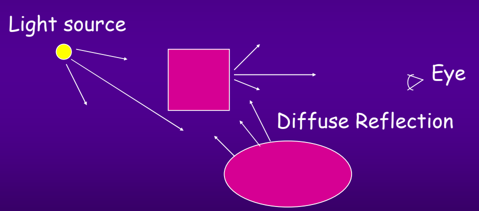
Radiosity



Radiosity

- Motivation: what is missing in ray-traced images?
  - Indirect illumination effects
  - Color bleeding
  - Soft shadows
- Radiosity is a physically-based illumination algorithm capable of simulating the above phenomena in a scene made of ideal diffuse surfaces.
- Books:
  - Cohen and Wallace, Radiosity and Realistic Image Synthesis, Academic Press Professional 1993.
  - Sillion and Puech, Radiosity and Global Illumination, Morgan-Kaufmann, 1994.

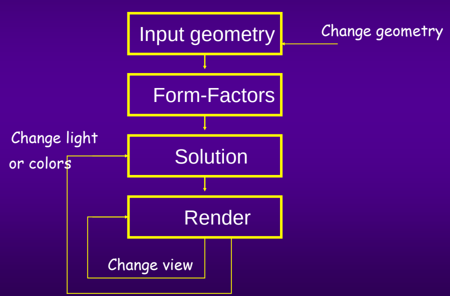
Indirect illumination effects



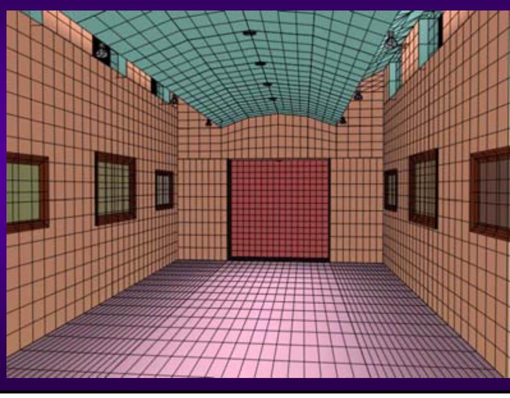
Radiosity in a Nutshell

- Break surfaces into many small elements
- Formulate and solve a linear system of equations that models the equilibrium of inter-reflected light in a scene.
- The solution gives us the amount of light leaving each point on each surface in the scene.
- Once solution is computed, the shaded elements can be quickly rendered from any viewpoint.

Radiosity



Meshing (partition into elements)



Radiometric quantities

- Radiant energy [J]
- Radiant power (flux): radiant energy per second [W]
- Irradiance (flux density): incident radiant power per unit area [W/m²]
- Radiosity (flux density): outgoing radiant power per unit area [W/m²]
- Radiance (angular flux density): radiant power per unit projected area per unit solid angle [W/(m² sr)]

The Radiosity Equation

- Assume that surfaces in the scene have been discretized into *n* small elements.
- Assume that each element emits/reflects light uniformly across its surface.
- Define the *radiosity B* as the total hemispherical flux density (W/m²) leaving a surface.
- Let's write down an expression describing the total flux (light power) leaving element *i* in the scene:

total flux = emitted flux + reflected flux

The Radiosity Equation

- Total flux leaving element i: $B_i A_i$
- Total flux emitted by element i: $E_i A_i$
- Total reflected flux:
  - (reflectance of element i)*(the total incoming flux)
  - total incoming flux = sum of contributions from all other elements in the scene $\rho_i \sum_j B_j A_j F_{ji}$
- The full radiosity equation is then:

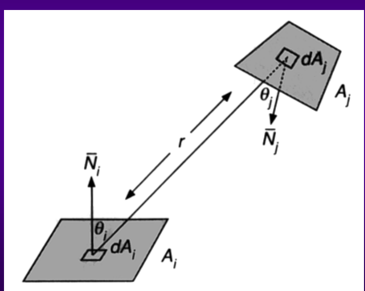
$$B_i A_i = E_i A_i + \rho_i \sum_{j=1}^{n} B_j A_j F_{ji}$$

The Form Factor

- The form factor F_{ji} tells us how much of the flux leaving element j actually reaches element i.

$$F_{ij} = \frac{1}{A_i} \int_{x \in A_i} \int_{y \in A_j} \frac{\cos\theta_x \cos\theta_y}{\pi \|x-y\|^2} V(x,y)\, dA_x\, dA_y$$

Form-Factor Computation



Properties of Form Factors

- Reciprocity: $A_i F_{ij} = A_j F_{ji}$
- Additivity: $F_{i(j \cup k)} = F_{ij} + F_{ik}$
- Conservation of energy in a closed environment:

$$\sum_{j=1}^{n} F_{ij} = 1$$

The Radiosity Equation

- The radiosity equation $B_i A_i = E_i A_i + \rho_i \sum_{j=1}^{n} B_j A_j F_{ji}$
- Divide equation by A_i: $B_i = E_i + \rho_i \sum_{j=1}^{n} B_j \frac{A_j}{A_i} F_{ji}$
- Apply form-factor reciprocity: $B_i = E_i + \rho_i \sum_{j=1}^{n} B_j F_{ij}$
- We can write this using matrix notation:

$$\begin{bmatrix} B_1 \\ \vdots \\ B_n \end{bmatrix} = \begin{bmatrix} E_1 \\ \vdots \\ E_n \end{bmatrix} + \begin{bmatrix} \quad & \quad & \quad \\ & & \\ & & \end{bmatrix} \begin{bmatrix} B_1 \\ \vdots \\ B_n \end{bmatrix}$$

Finally...

- A linear system of n equations in n unknowns:

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

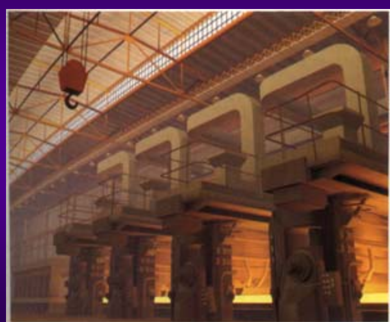
The Radiosity Method

- Take as input a geometric model of the scene, with emission and reflection properties of each surface
- Step 1 - Meshing: Discretize input surfaces into a mesh of small elements
- Step 2 - Setup: Compute the form factors F_{ij}
- Step 3 - Solution: Solve the resulting linear system of equations
- Step 4 - Display: Render shaded elements from any desired view point.
- These steps are often interleaved in practice.

Examples:

Examples:

Examples:



Solving the Equation

- The naive approach - Gaussian elimination
  - Requires $O(n^2)$ memory to store the matrix
  - Requires $O(n^3)$ time to solve the equation
- A better approach - iterative solution
  - Jacobi iteration
  - Gauss-Seidel iteration
  - Southwell relaxation (known as Progressive Radiosity)
- Due to special properties of the radiosity matrix, it is possible to prove that these iterative methods are guaranteed to converge to the correct solution.

Progressive Radiosity

- While not converged:
  - Select one element in the scene as the current light source
  - "Shoot" radiosity from the light source to the rest of the scene
- The solution process mimics the physical process of light propagation in the scene.
- Must take care not to shoot the same light more than once (keep track of "unshot radiosity")

```
B[i] = Unshot[i] = E[i]
while (not converged) {
  Choose i with largest Unshot[i]*A[i]
  Shoot(i)
}
```

Progressive Radiosity

```
B[i] = Unshot[i] = E[i]
while (not converged) {
  Choose i with largest Unshot[i]*A[i]
  Shoot(i)
}
```

```
Shoot(i):
  for j = 1..n {
    Compute the form factor FF[i,j]
    Delta[j] = ρ[j] FF[i,j] Unshot[i] A[i]/A[j]
    B[j] += Delta[j]
    Unshot[j] += Delta[j]
  }
  Unshot[i] = 0
```

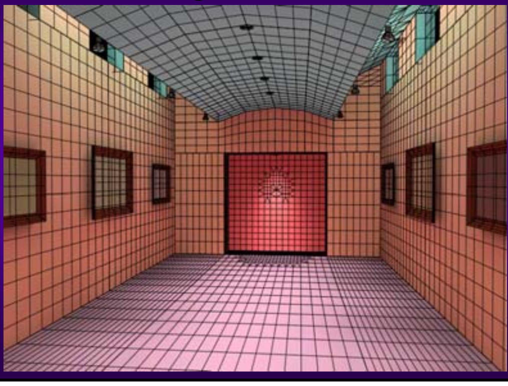
Progressive Radiosity

- In each iteration the algorithm computes n form factors on the fly, removing the $O(n^2)$ storage complexity.
- Choosing the "brightest" shooter at each iteration makes the solution to converge rapidly during the first iterations.
- It is possible to display the solution after each iteration, resulting in a progressive sequence of images.
- Typically, there is no need to run until complete convergence. The process can be stopped after relatively few iterations.

Progressive Radiosity Example: 0, 8, 16, 25, 50, 100 iterations

Progressive Radiosity + Ambient 0, 8, 16, 25, 50, 100 iterations

Progressive Radiosity with Adaptive Meshing (0,8,16,50,100)



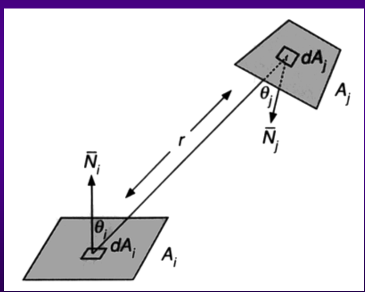
Form-Factor Computation

- Analytic as well as approximate formulas exist for various configurations.
- A closed form expression for a form factor between two polygons (Schroder 93):
  - extremely complicated formula
  - does not take into account occlusion
- A common approximation is to assume that the inner integral is constant for all locations x in element i:

$$F_{ij} = \frac{1}{A_i} \int_{x \in A_i} \int_{y \in A_j} \frac{\cos\theta_x \cos\theta_y}{\pi r^2} V(x,y)\, dA_x\, dA_y$$

$$\approx \int_{y \in A_j} \frac{\cos\theta_x \cos\theta_y}{\pi r^2} V(x,y)\, dA_y$$

Form-Factor Computation



Form-Factor Computation

- The remaining integral is approximated as a finite sum:

$$F_{ij} \approx \int_{y \in A_j} \frac{\cos\theta_x \cos\theta_y}{\pi r^2} V(x,y)\, dA_y$$

$$\approx \sum \frac{\cos\theta_x \cos\theta_y}{\pi r^2} V(x,y)\, \Delta A_y$$

- This approximation works well so long as the elements i and j are *well-separated* from each other (the distance between them is large relative to their sizes)

Nusselt's Method

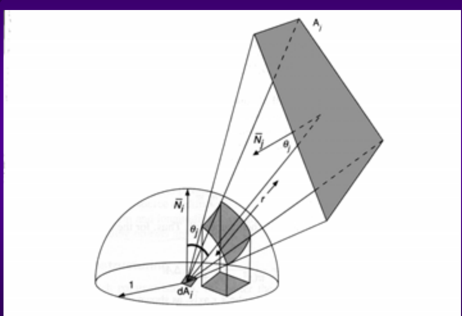


Fig. 16.66 Determining the form factor between a differential area and a patch using Nusselt's method. The ratio of the area projected onto the hemisphere's base to the area of the entire base is the form factor. (After [SIEG81].)

Hemicube

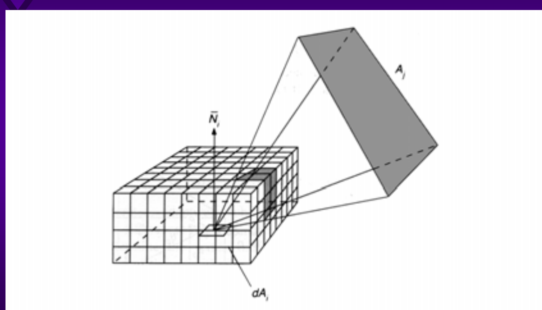


Fig. 16.67 The hemicube is the upper half of a cube centered about the patch. (After [COHE85].)