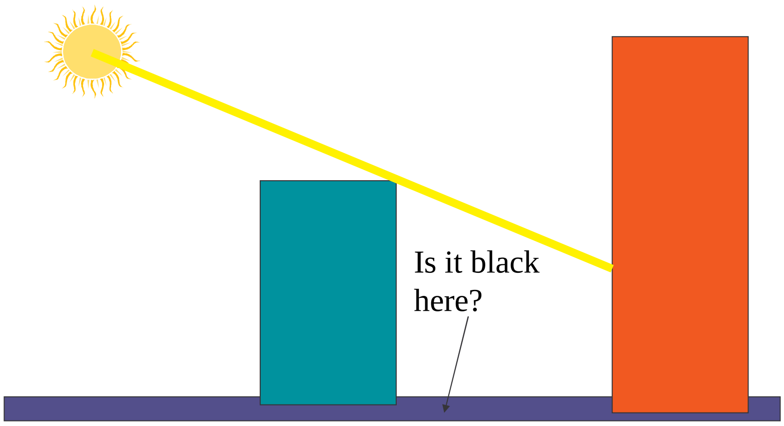
Radiosity

- Local Lighting (OpenGL).
  - Captures interaction of light source and surface, but nothing else in environment.
- Shadow maps
  - Light source gated by environment.
- Ray tracing
  - Captures light rays striking surface from light source *or* from specular or refractive direction.
- What is missing?
  - Light striking diffuse material from non-sources.

Does this matter?



(Wikipedia)

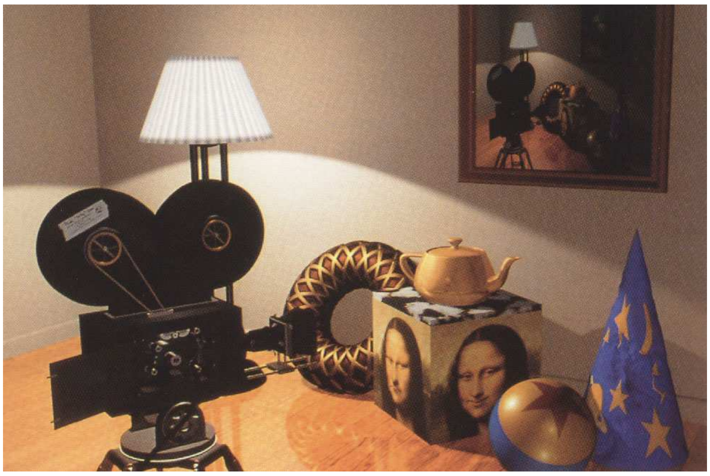
Radiosity

- Global illumination for diffuse surfaces
- Models view-independent illumination
- Diffuse/soft shadows, color bleeding

Local Illumination

Slide 5 Lecture 19

Ray-tracing-like Illumination



Slide 6 Lecture 19

Radiosity-like Illumination

Slide 7 Lecture 19

Radiosity Illumination

Image: Lightscape Inc.

Slide 8 Lecture 19

Radiosity

- Light energy per unit time per unit area
- Based on conservation of light energy
- Assumes area light sources

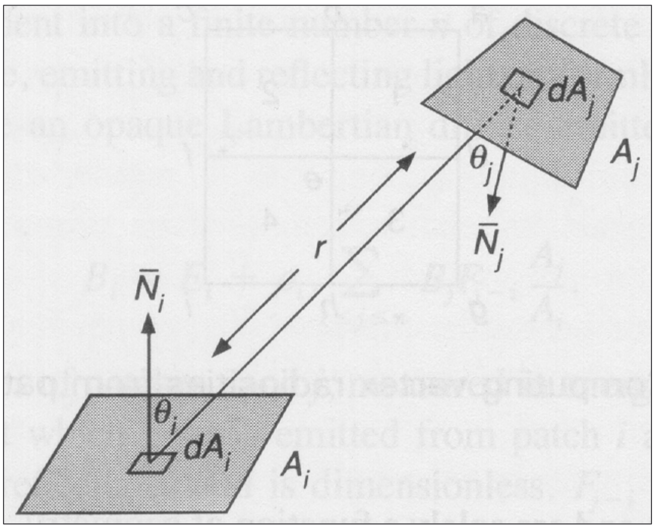
Radiosity Equation

For every surface patch *i :*

$B_i = E_i + \rho_i H_i$, where B_i is the radiosity, E_i is the emissivity, ρ_i is the coefficient of reflectivity, and H_i is the incident energy per unit time per unit area for patch *i* .

$E_i = 0$ for all surfaces that are not light sources

Modeling Light Reflections



Slide 11 Lecture 19

Form Factor

- F_{ji} = Fraction of light energy leaving surface j and arriving at surface i
- $A_i F_{ij} = A_j F_{ji}$
- Sum of F_{ij} for all j's is 1

Slide 12 Lecture 19

Radiosity Equation

Assuming *n* surface patches in the environment:

$H_i = \Sigma_{j=1 \text{ to } n}(F_{ji} B_j A_j/A_i) = \Sigma_{j=1 \text{ to } n}(F_{ij}B_j)$

Therefore, the radiosity equation is:

$B_i = E_i + \rho_i H_i \; = E_i + \rho_i \Sigma_{j=1 \text{ to } n}(F_{ij}B_j)$, or

$B_i - \rho_i \Sigma_{j=1 \text{ to } n}(F_{ij}B_j) = E_i$

Slide 13 Lecture 19

Radiosity Equation

Assuming *n* surface patches in the environment:

$H_i = \Sigma_{j=1 \text{ to } n}(F_{ij}B_j)$

Therefore, the radiosity equation is:

$B_i = E_i + \rho_i H_i \; = E_i + \rho_i \Sigma_{j=1 \text{ to } n}(F_{ij}B_j)$, or

$B_i - \rho_i \Sigma_{j=1 \text{ to } n}(F_{ij}B_j) = E_i$

Slide 14 Lecture 19

Radiosity Equation

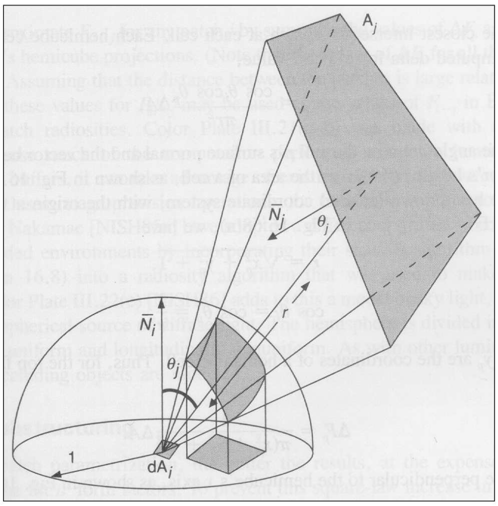
$$\begin{pmatrix} 1-\rho_1 F_{11} & -\rho_1 F_{12} & \dots & -\rho_1 F_{1n} \\ -\rho_2 F_2 & 1-\rho_2 F_{22} & \dots & -\rho_2 F_{2n} \\ : & : & & : \\ : & : & & : \\ -\rho_n F_{n1} & -\rho_n F_{n2} & \dots & 1-\rho_n F_{nn} \end{pmatrix} \begin{bmatrix} B_1 \\ B_2 \\ : \\ : \\ B_n \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ : \\ : \\ E_n \end{bmatrix}$$

F_{ij}'s are the form factors that are dependent on scene geometry and can be determined ($F_{ii} = 0$ for planar polygons).

Radiosity Equation

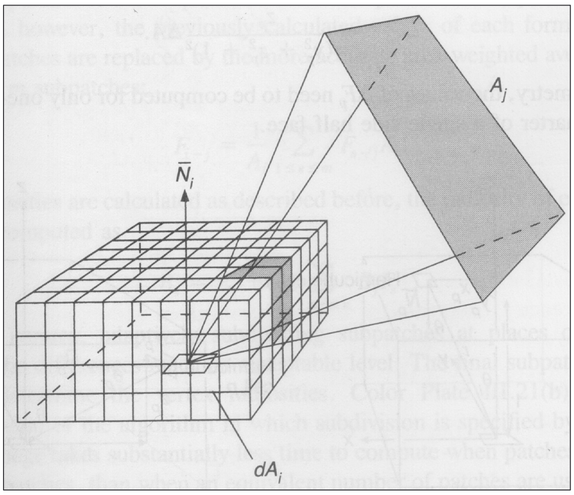
- Directly solving the linear system of equations on the previous slide: **A b** = **e** is expensive
  - – Gaussian Elimination method takes $O(n^3)$ time
- In practice we take advantage of the structure of **A** (strictly diagonally dominant) to efficiently solve this system

But first … let us see how to compute Form Factors



Slide 17 Lecture 19

Hemi-Cube Form Factors



Slide 18 Lecture 19

Progressive Refinement Radiosity

- This approach computes an approximate solution of the system of linear equations **A b** = **e**
- The basic approach is to identify the *brightest* patch in the environment and *shoot* (distribute) its energy to the other patches that can *see* it.
- This is equivalent to computing only those rows of the form-factor matrix **A** that correspond to the brightest patches.
- In practice, this approach results in a fast convergence to the solution without computing all the rows of **A**

Progressive Refinement Radiosity

Initially, $\Delta B_i = E_i$

while **max**(ΔB_i) > ε

{ select patch *i* with maximum unshot energy ΔB_i

for each patch *j* do

{ $\Delta R = \rho_j \Delta B_i F_{ij} A_i / A_j$

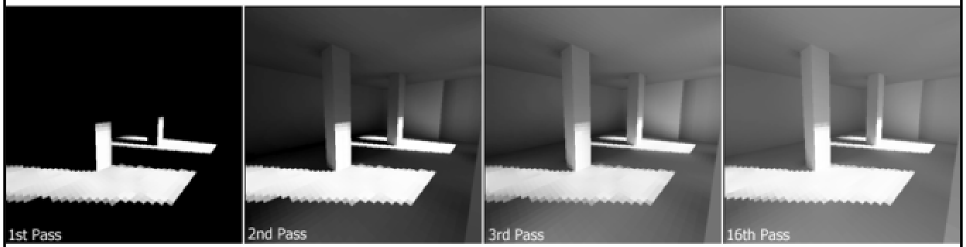
$\Delta B_j = \Delta B_j + \Delta R$ (unshot energy for patch *j*)

$B_j = B_j + \Delta R$ (accumulated energy)

}

$\Delta B_i = 0$

}



(Wikipedia)

Radiosity Artefacts

- Light Leaks
- Hemicube aliasing
- Light distribution at patches, whereas vertices are used for shading
- Mesh discretization (solution: adaptive meshing)
- Need a separate specular pass to capture specular highlights