Light Simulation

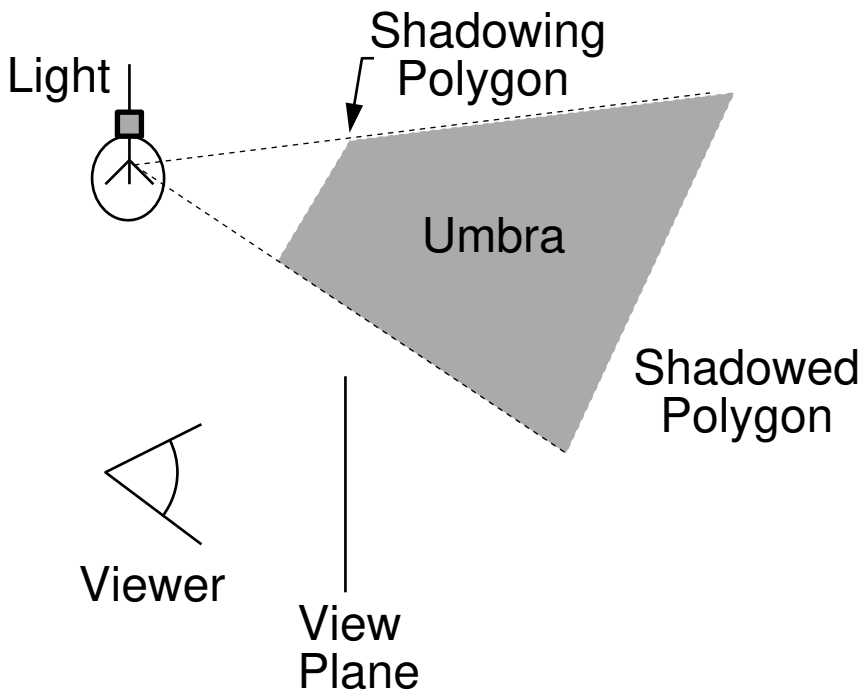
Thomas A. Funkhouser

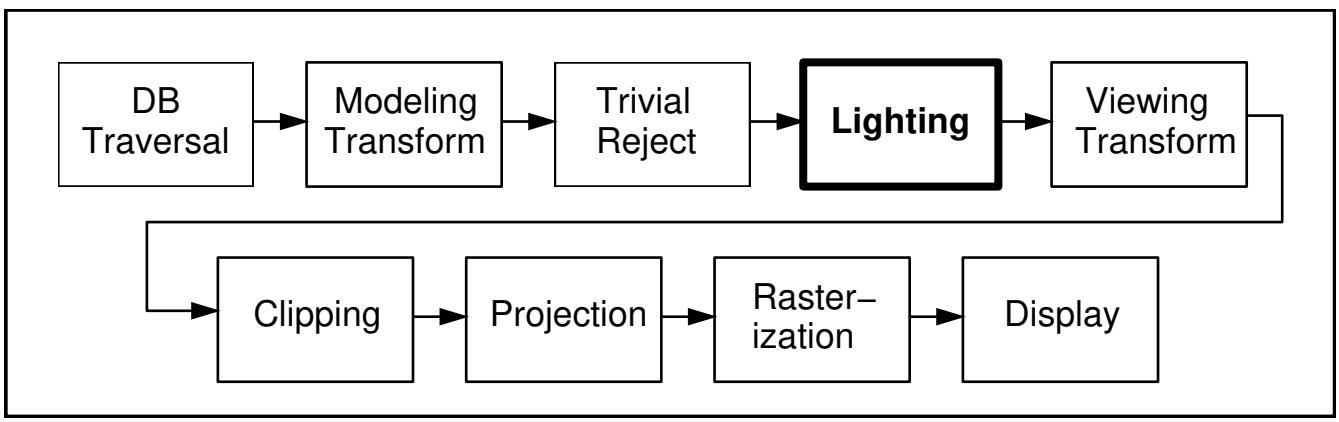
Overview

- **Shadows**
- **Radiosity**

Shadows

- **Goal**
  - Simulate shadows cast by one object onto another





Shadows

- **Direct illumination with shadows**
  - For each illumination computation, compute Si based on direct shadows:

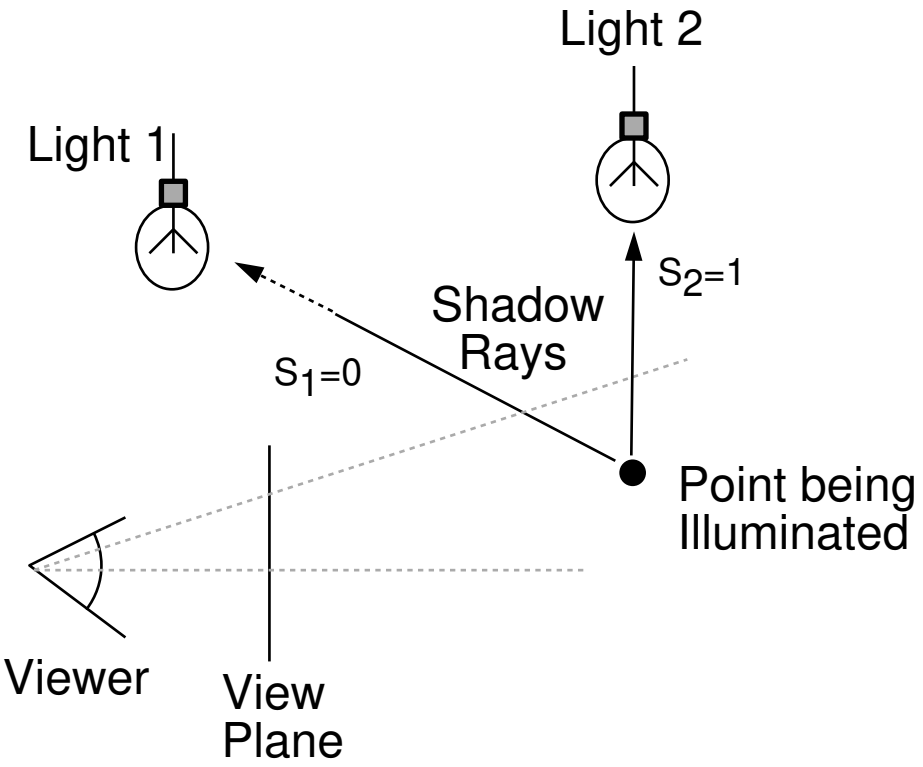
$$I = I_E + K_A I_{AL} + \sum_{L_i} (K_D (N \bullet L_i) + K_S (V \bullet R_i)^n) \mathbf{S_i} I_{L_i}$$

$$S_i = \begin{cases} 0, \text{ if light } i \text{ is blocked} \\ 1, \text{ if light } i \text{ is not blocked} \end{cases}$$

- **Shadow algorithms**
  - Ray casting
  - Light buffer algorithm
  - Scan line algorithm
  - *Shadow volumes*
  - etc.

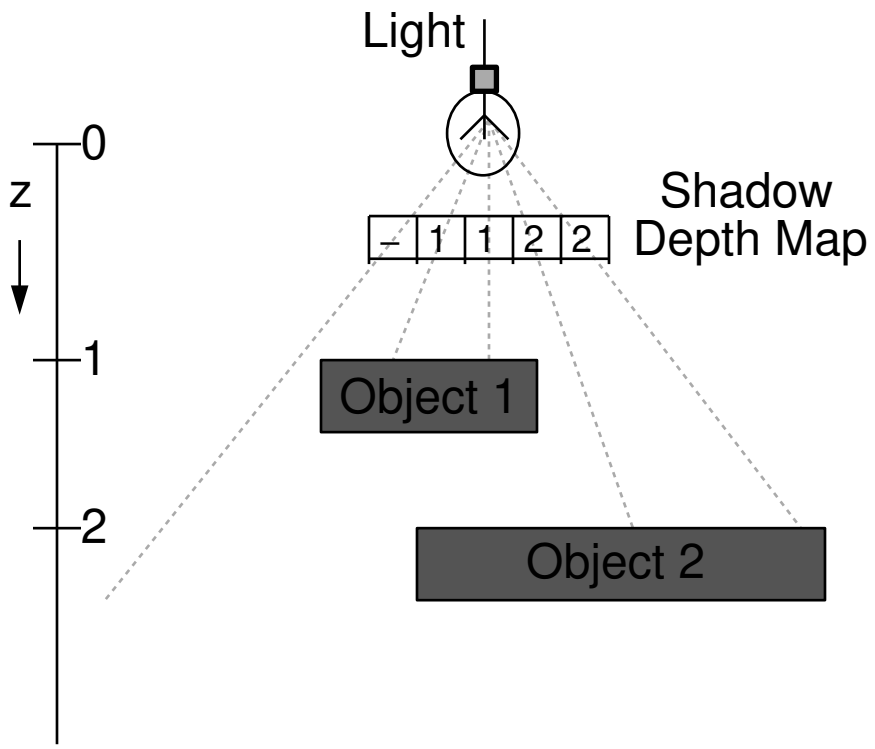
Shadow Algorithms

- **Ray Casting**
  - Cast ray to each light source
  - Check for intersections with other polygons to see if the light source is shadowed ($S_i = 0$) or not ($S_i = 1$)



Shadow Algorithms

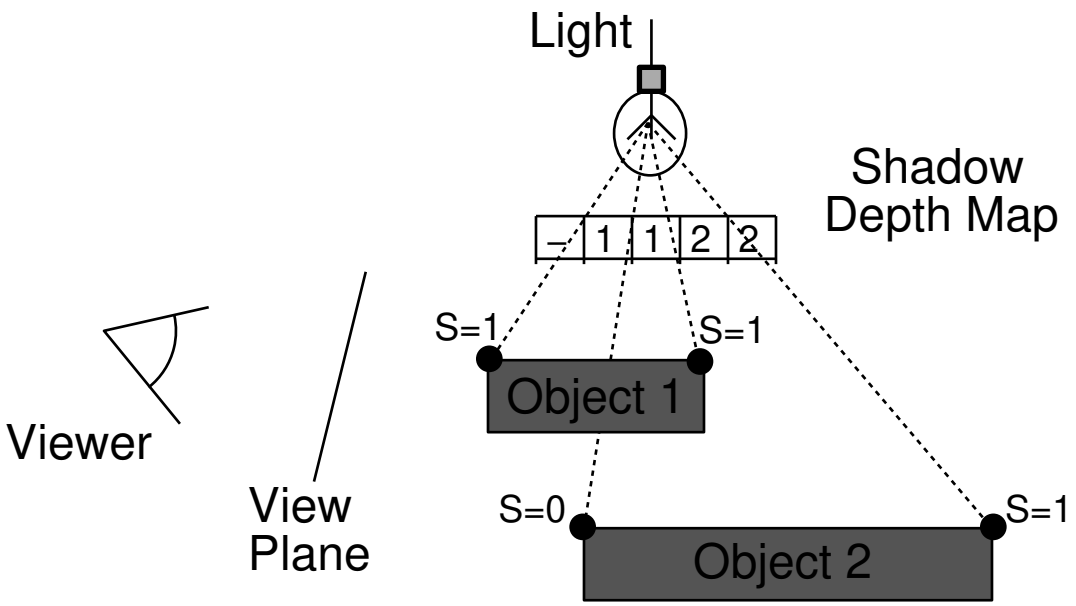
- **Shadow buffers**
  - At beginning, pre–render scene from viewpoint of light source and store ‘‘shadow depth map’’
    - SB(x,y) = minimum Z coord of any polygon projecting to (x,y) from light’s viewpoint



Shadow Algorithms

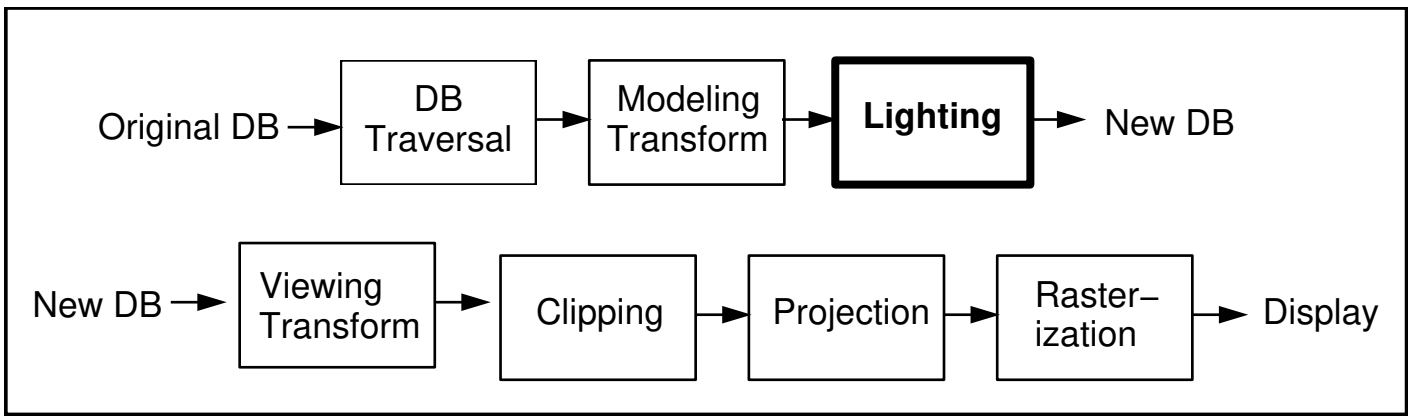
- **Shadow buffers (cont)**
  - When computing illumination for a point, transform (x,y,z) into (x',y',z') in the light's coordinate system.

$$Si = \begin{cases} 1, \text{ if } z' <= SB(x',y') \\ 0, \text{ if } z' > SB(x',y') \end{cases}$$



Shadow Algorithms

- **Shadow Volumes**
  - Split scene polygons according to regions in/out of shadow

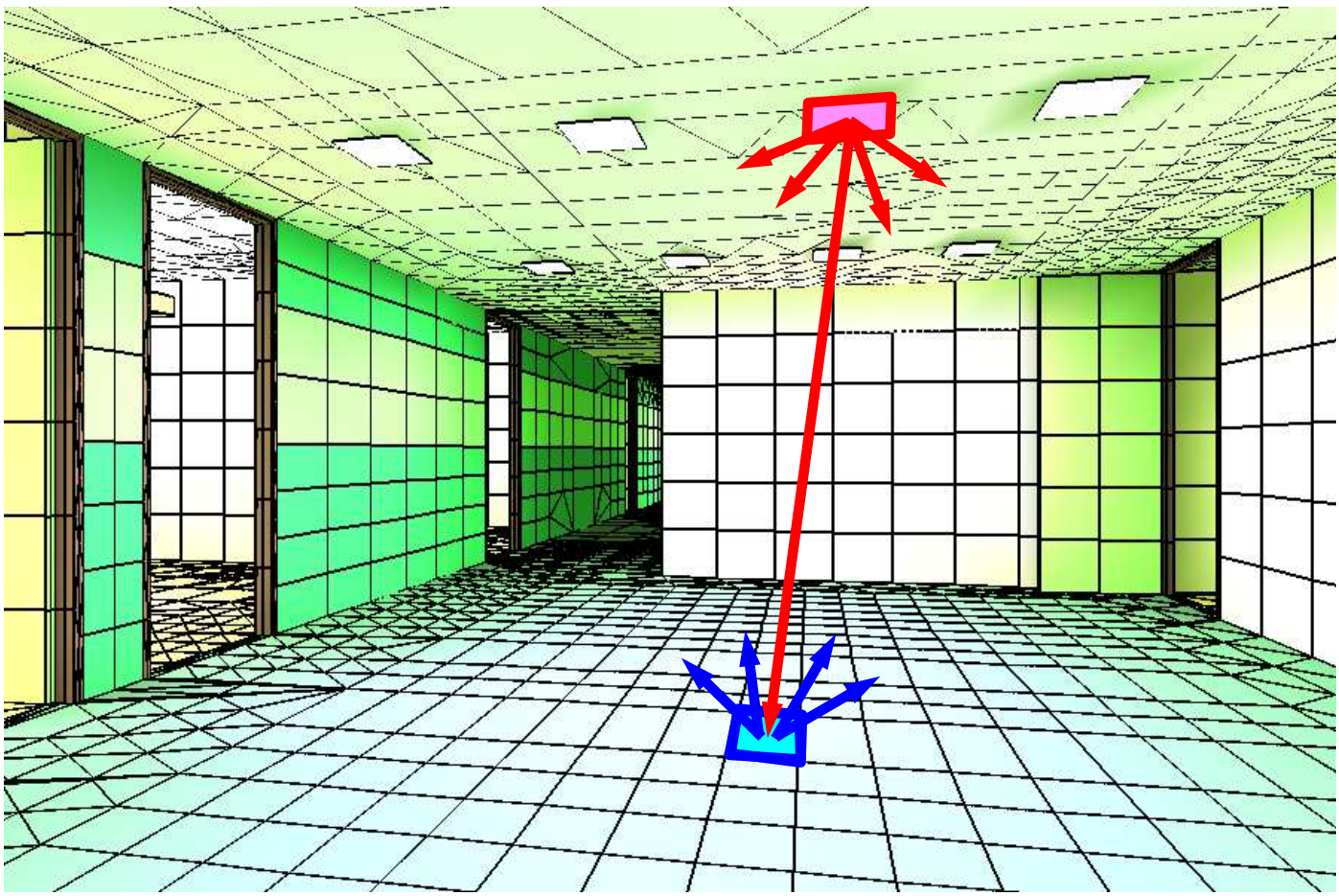


Radiosity

- **Goal**
  - Eliminate ‘‘ambient’’ term in illumination equation by more accurately modeling *diffuse* interobject reflection

Radiosity

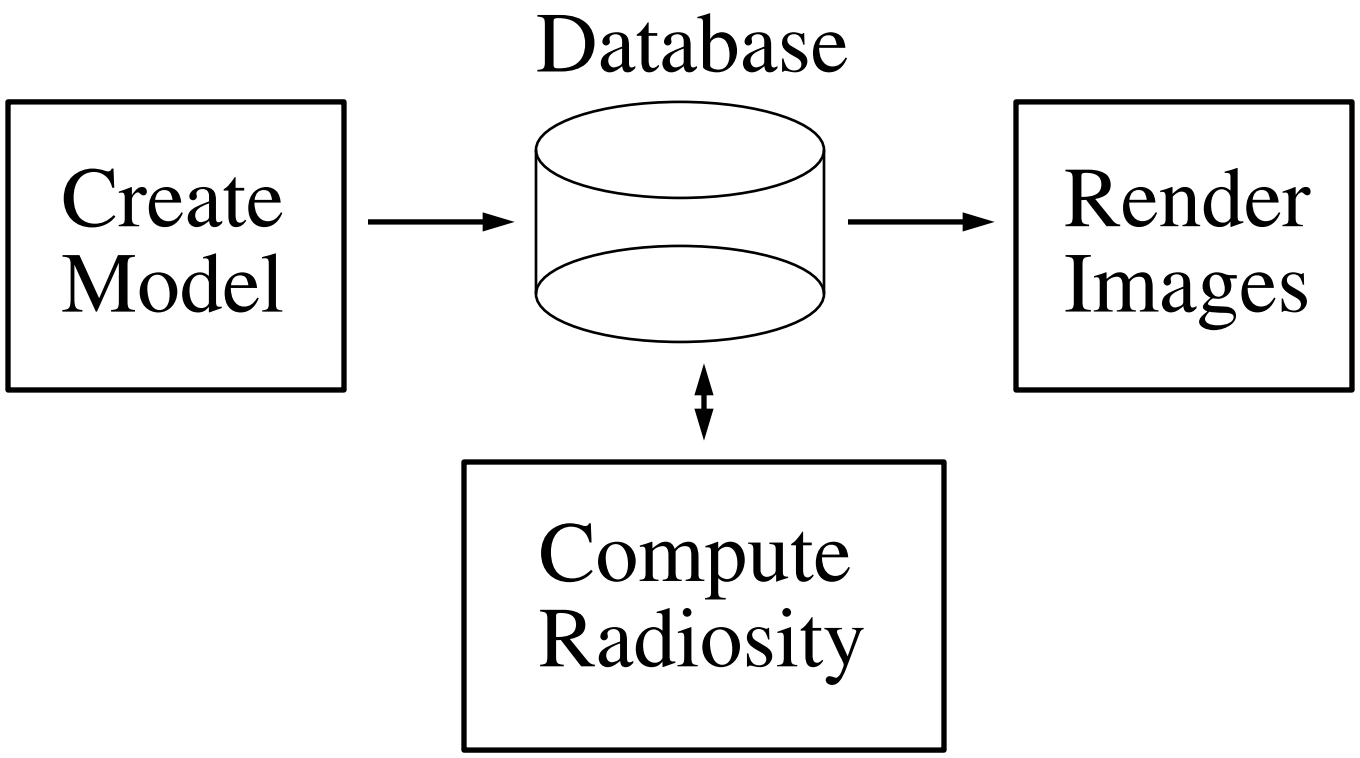
- **Basic idea**
  - Treat every polygon as light source

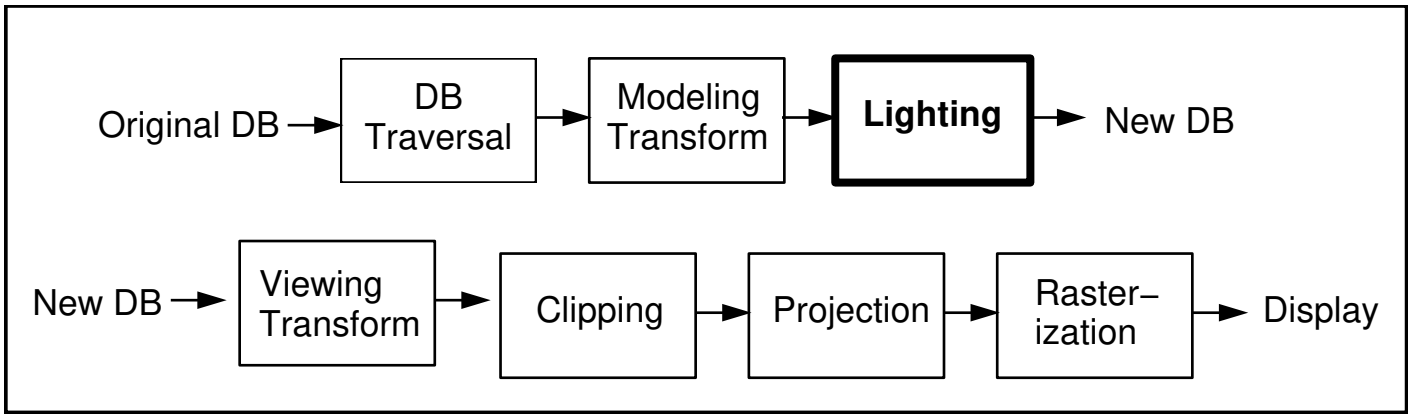


- **Advantages**
  - Physically models shadows and indirect diffuse illumination
  - Independent of any viewpoint

Radiosity

- **Two–Phase Rendering:**





Radiosity Equation

♦ Equation Formulation:

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

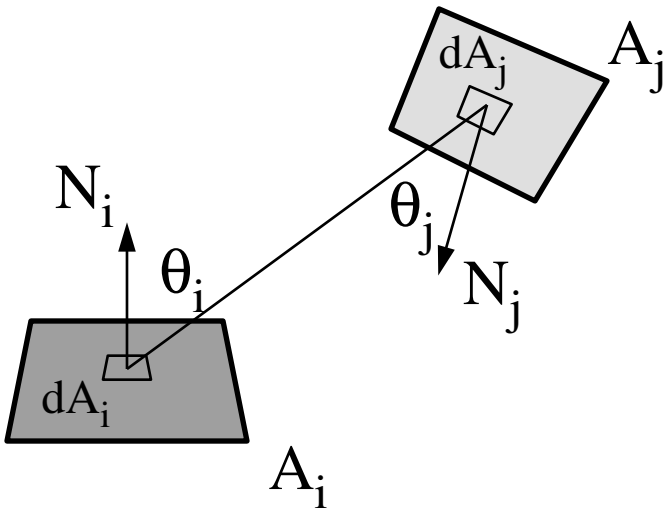
B_i = Radiosity of patch i
E_i = Emission of patch i
p_i = Reflectivity of patch i (K_D)
F_{ij} = Form–factor between patches i and j

♦ Matrix Formulation:

$$\begin{bmatrix} 1-\rho_1F_{11} & -\rho_1F_{12} & \cdots & -\rho_1F_{1n} \\ -\rho_2F_{21} & 1-\rho_2F_{22} & \cdots & -\rho_2F_{2n} \\ -\rho_3F_{31} & -\rho_3F_{32} & \cdots & -\rho_3F_{3n} \\ \vdots & \vdots & \vdots & \vdots \\ -\rho_nF_{n1} & -\rho_nF_{n2} & \cdots & 1-\rho_nF_{nn} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ B_3 \\ \\ B_n \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ E_3 \\ \vdots \\ E_n \end{bmatrix}$$

Form Factors

- **Definition:**
  - Fraction of energy leaving patch j that arrives at patch i



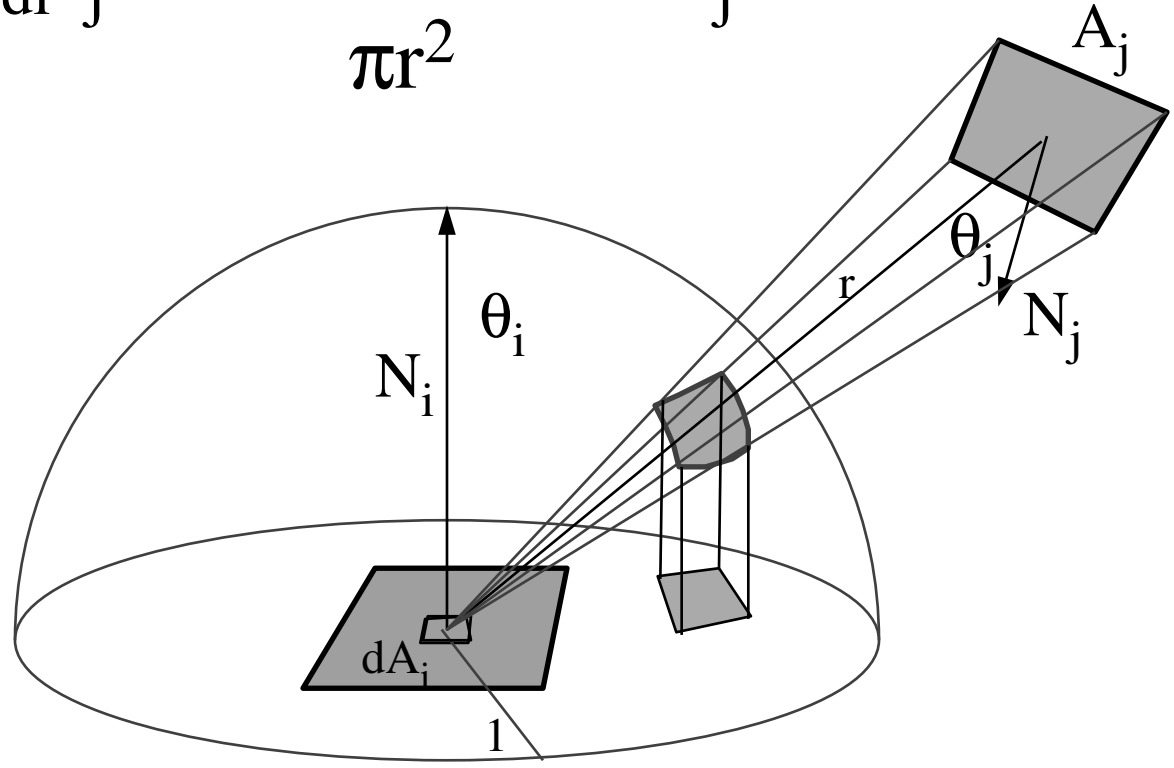
$$F_{di-dj} = \frac{\cos\theta_i \cos\theta_j}{\pi r^2} H_{ij}\, dA_j$$

$$F_{ij} = \frac{1}{A_i} \int_{A_i} \int_{A_j} \frac{\cos\theta_i \cos\theta_j}{\pi r^2} H_{ij}\, dA_j dA_i$$

Computing Form Factors

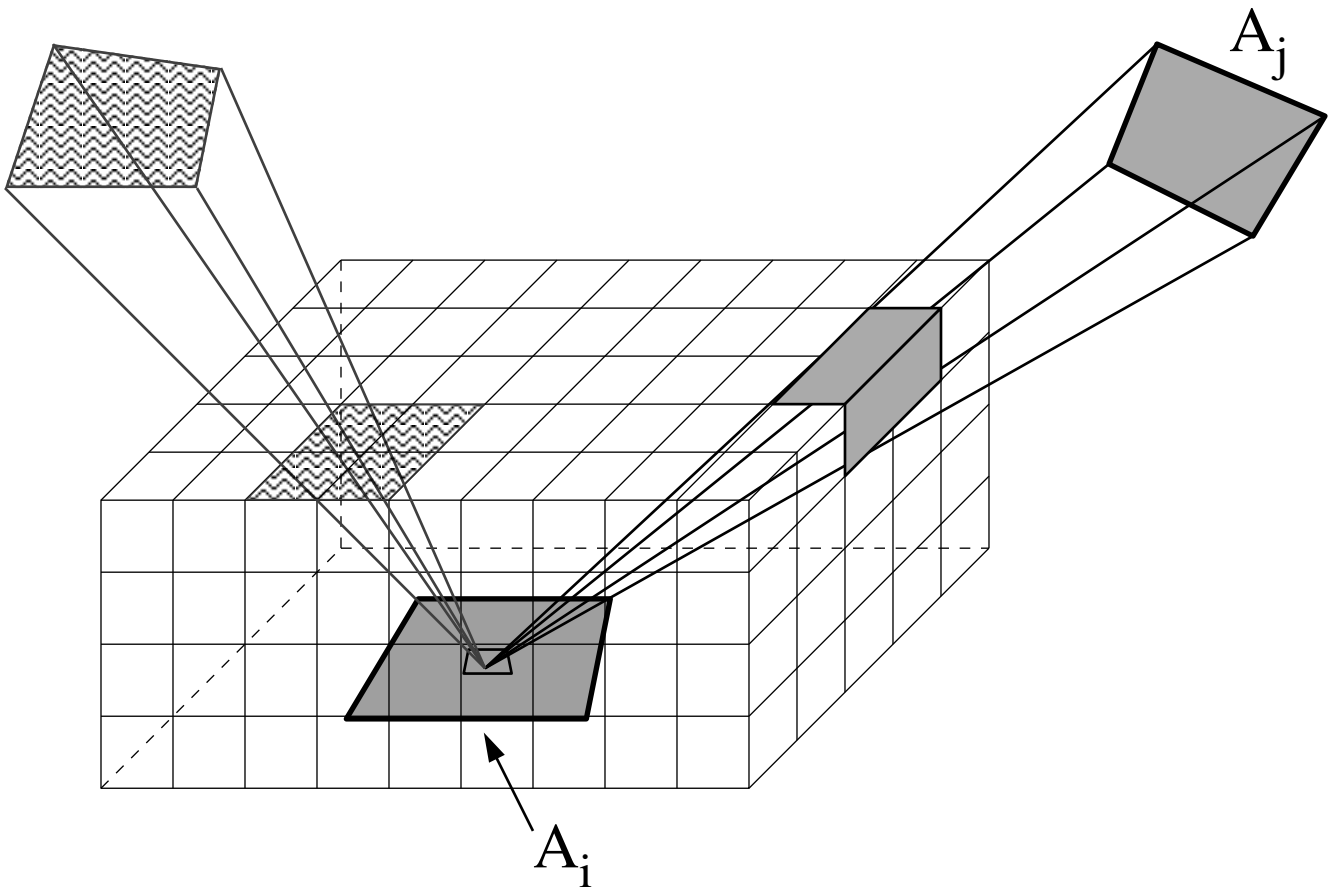
- **Method:**
  - Project onto unit hemisphere ($A_j \cos \theta_j / r^2$)
  - Project onto unit circle base ($\cos \theta_i$)
  - Divide by area of circle ($1/\pi$)

$$F_{di-j} = \frac{\cos \theta_i \cos \theta_j}{\pi r^2} A_j$$



Hemi–Cube Form Factors

- **Method:**
  - Project scene onto hemi–cube positioned at centroid of patch i
  - Count pixel coverage to determine form factors F_{i*}
  - Can use hardware z–buffer
  - Image precision (aliasing)



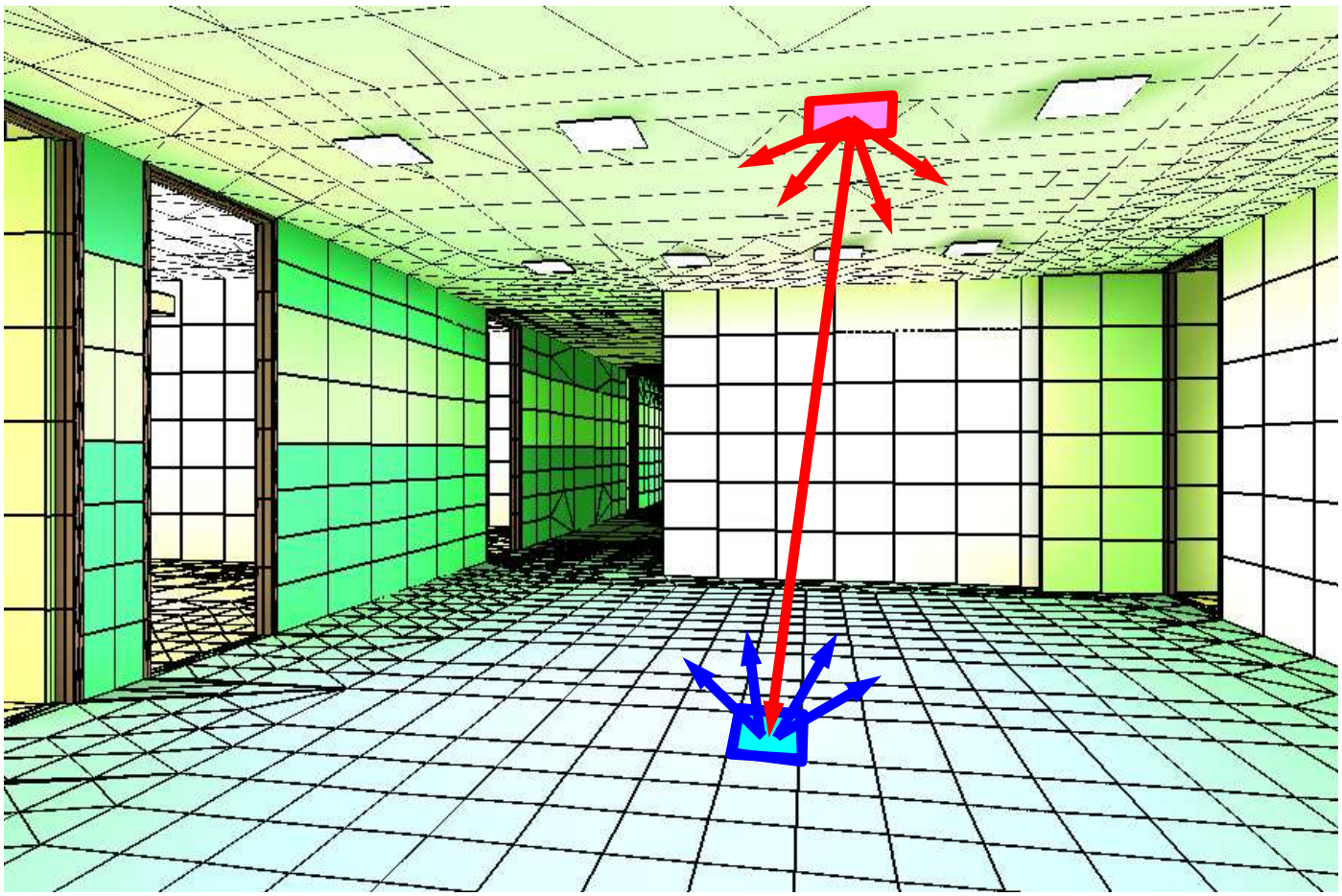
Matrix Solution Methods

- **Invert the Matrix:**
  - Gaussian elimination
  - $O(n^3)$
- **Gauss–Seidel Iteration**
  - Iterate gather radiosity to patches
  - Relax rows of variables
  - $O(n^2)$

$$\begin{bmatrix} 1-\rho_1 F_{11} & -\rho_1 F_{12} & \cdots & -\rho_1 F_{1n} \\ -\rho_2 F_{21} & 1-\rho_2 F_{22} & \cdots & -\rho_2 F_{2n} \\ -\rho_3 F_{31} & -\rho_3 F_{32} & \cdots & -\rho_3 F_{3n} \\ \vdots & \vdots & \vdots & \vdots \\ -\rho_n F_{n1} & -\rho_n F_{n2} & \cdots & 1-\rho_n F_{nn} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ B_3 \\ \\ B_n \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ E_3 \\ \vdots \\ E_n \end{bmatrix}$$

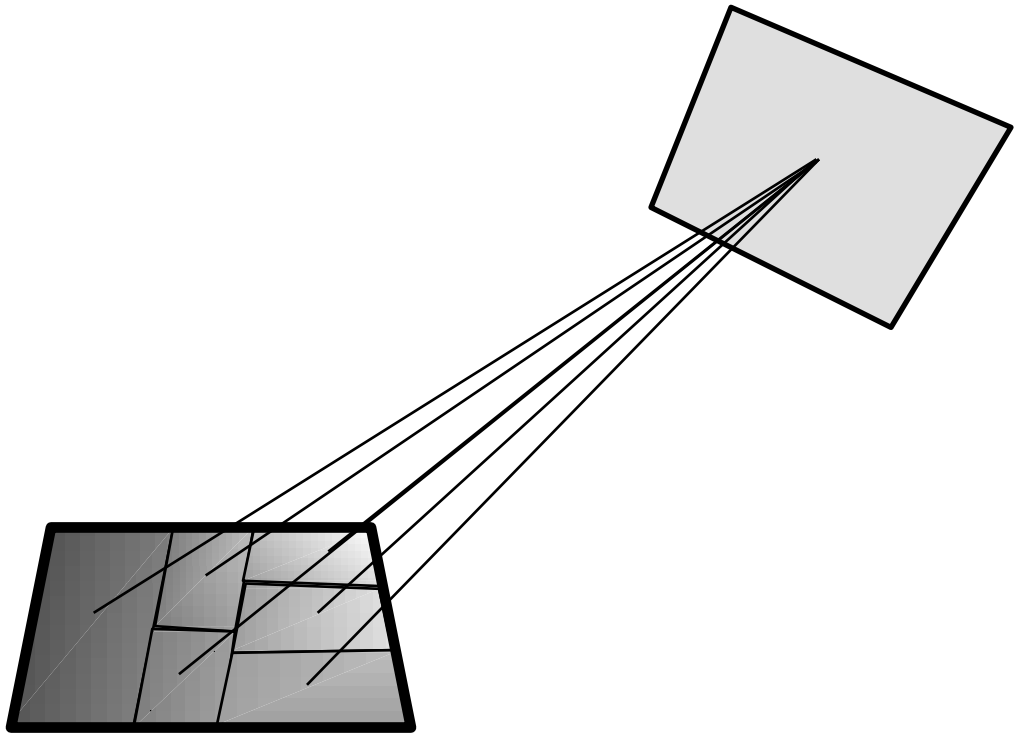
Matrix Solution Methods

- **Progressive refinement**
  - Iterate shoot radiosity from patches
  - Relax columns of variables
  - Select patches to shoot in order of unshot radiosity
  - $O(n^2)$ computation, but get pretty good solutions more quickly



Mesh Substructuring

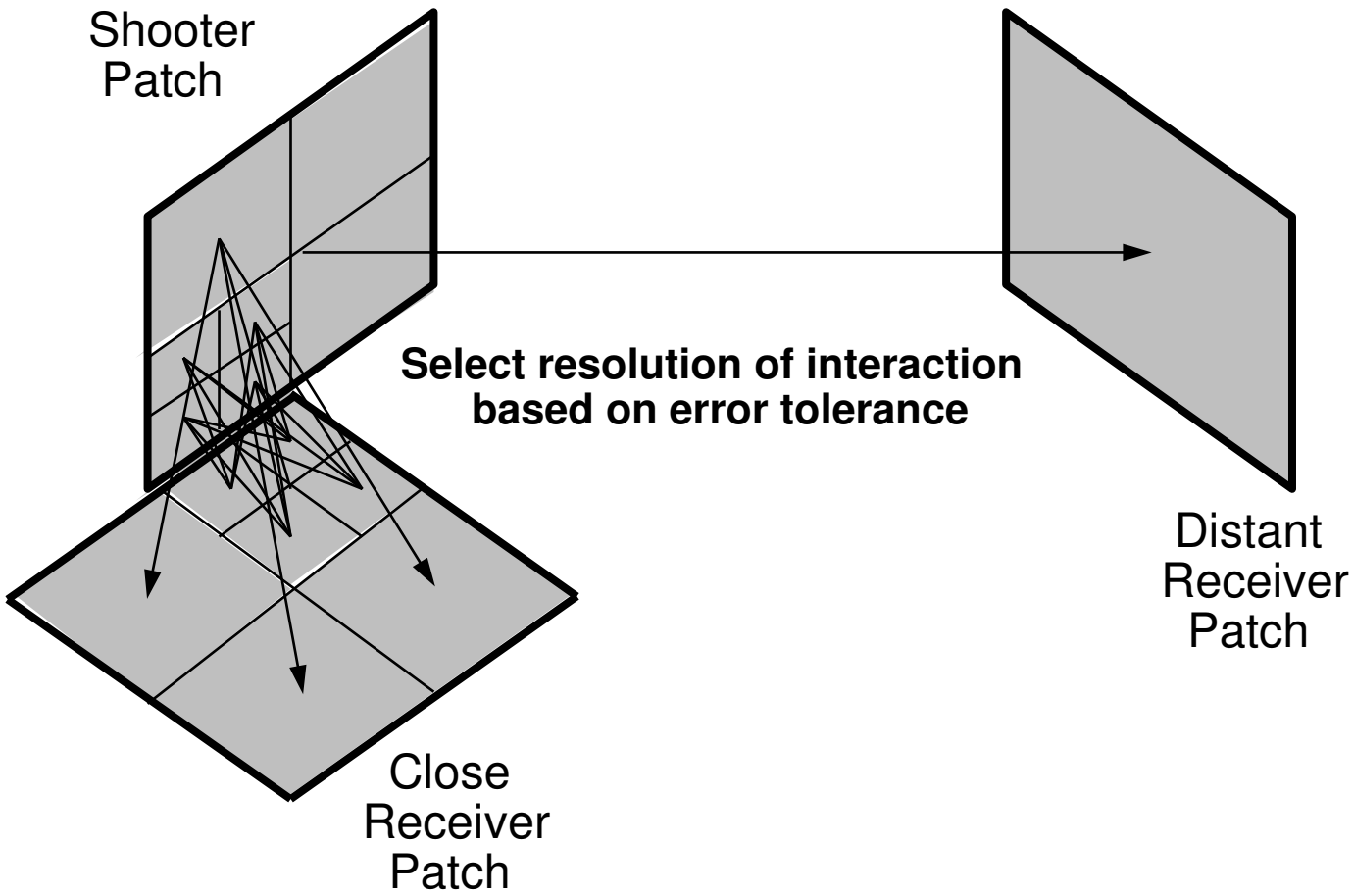
- **Adaptive subdivision**
  - Partition each patch into multiple ‘‘elements’’ (along gradients of radiosity)
  - O(np) if only gather to elements from patches



Patch partitioned into 6 elements

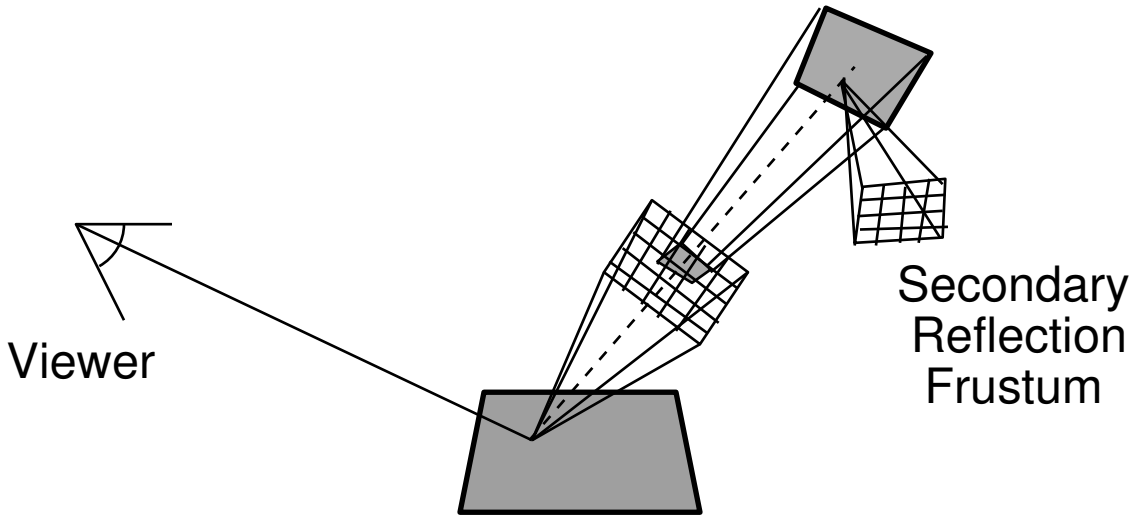
Hierarchical Radiosity

- **Multiresolution Computation**
  - Substructure patches into quad–tree
  - Transfer energy using lower resolution mesh elements if can do so within error tolerance
  - O(n) computation



Ray Tracing and Radiosity

- **Dilemma:**
  - Radiosity is good at diffuse interobject reflection
  - Ray tracing is good at specular interobject reflection
- **Combine them**
  - Example: compute diffuse interobject reflections in a ray tracer
  - Monte Carlo methods



Summary

- **Precomputation of lighting**
  - Shadows
  - Radiosity