Global Illumination and Radiosity

CS434

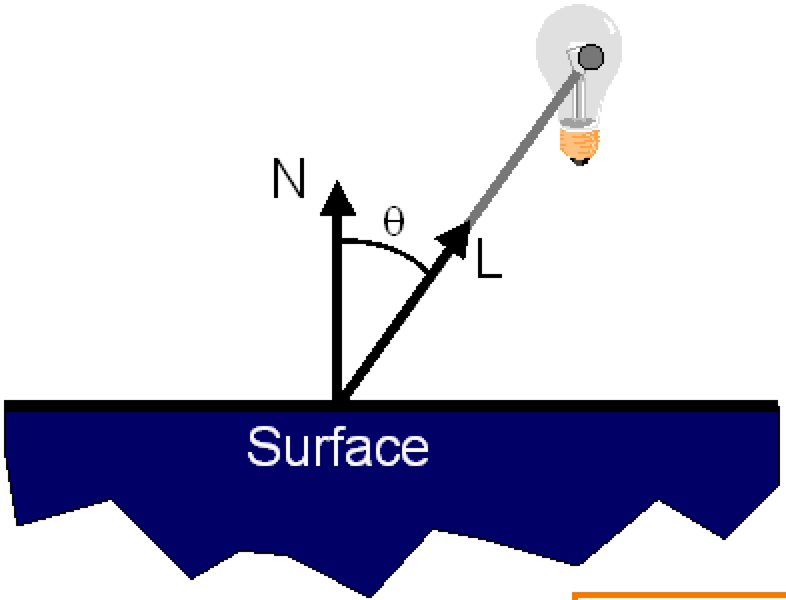
Daniel G. Aliaga
Department of Computer Science
Purdue University

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

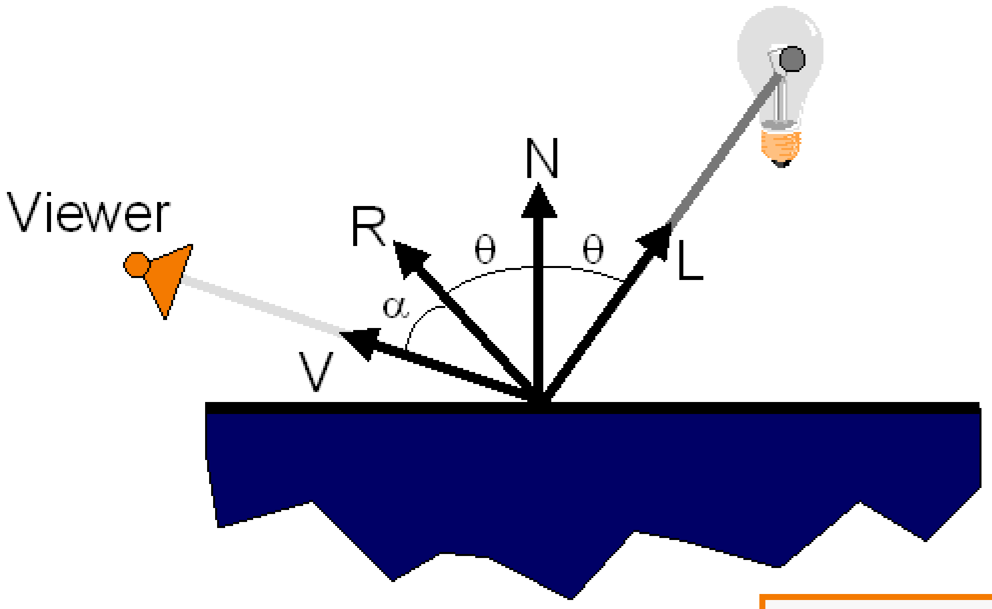
- Diffuse reflection
  - Lambertian model



$$I_D = K_D(N \bullet L)I_L$$

Recall: Lighting and Shading

- Specular reflection
  - Phong model



$$I_S = K_S (V \bullet R)^n I_L$$

Recall: Lighting and Shading

- Well....there is much more

For example…

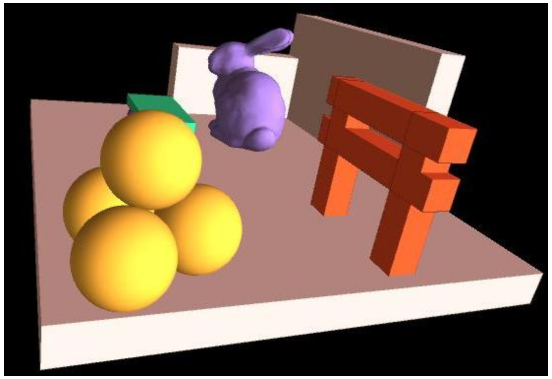
- Reflection -> Bidirectional Reflectance Distribution Functions (BRDF)
- Diffuse, Specular -> Diffuse Interreflection, Specular Interreflection
- Color bleeding
- Transparency, Refraction
- Scattering
  - Subsurface scattering
  - Through participating media
- And more!

Illumination Models

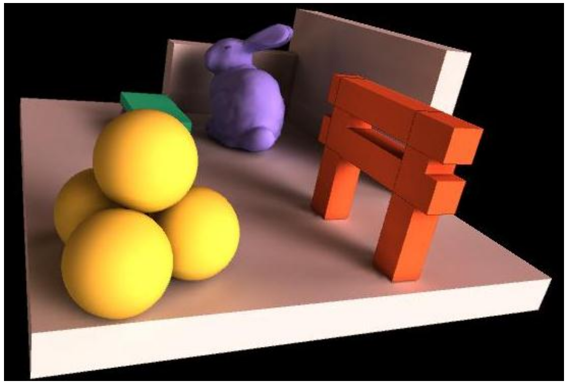
- So far, you considered mostly local (direct) illumination
  - Light directly from light sources to surface
  - No shadows (actually is a global effect)
- Global (indirect) illumination: multiple bounces of light
  - Hard and soft shadows
  - Reflections/refractions (you kinda saw already)
  - Diffuse and specular interreflections

Welcome to Global Illumination

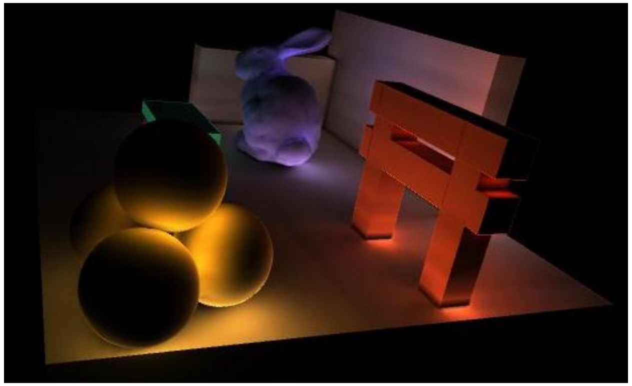
- *Direct illumination* + *indirect illumination*; e.g.
  – Direct = reflections, refractions, shadows, …
  – Indirect = diffuse and specular inter-reflection, …



direct illumination



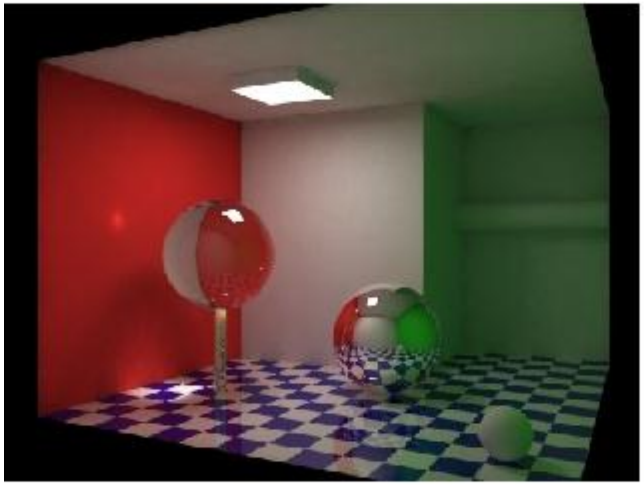
with global illumination



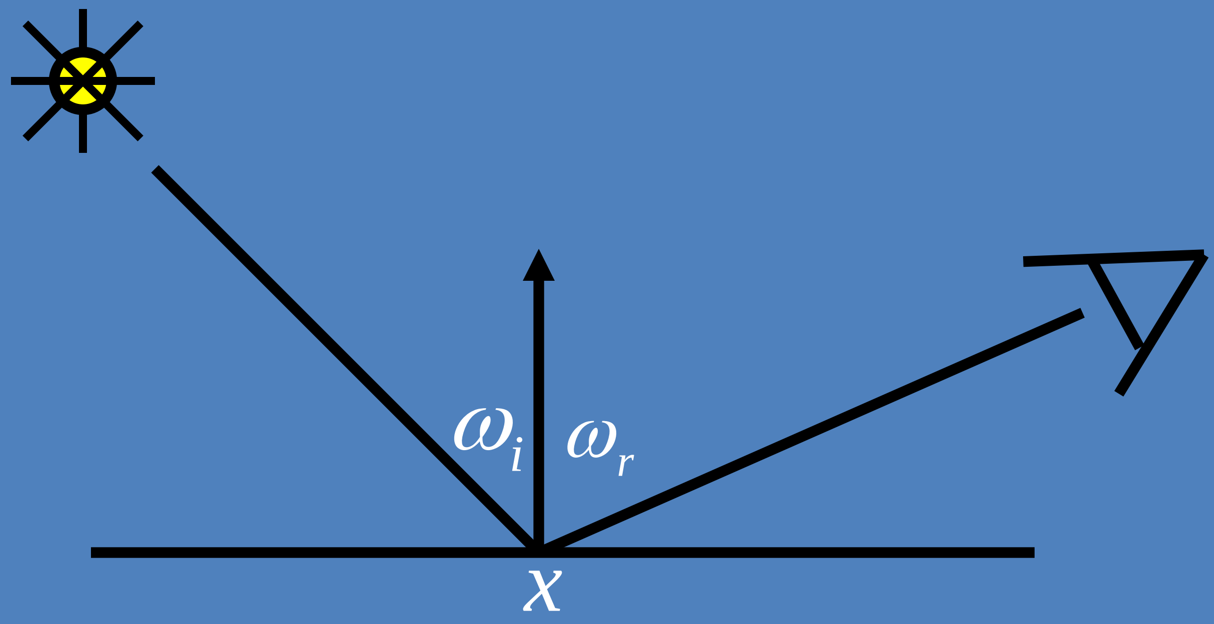
only diffuse inter-reflection

Global Illumination

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



Reflectance Equation

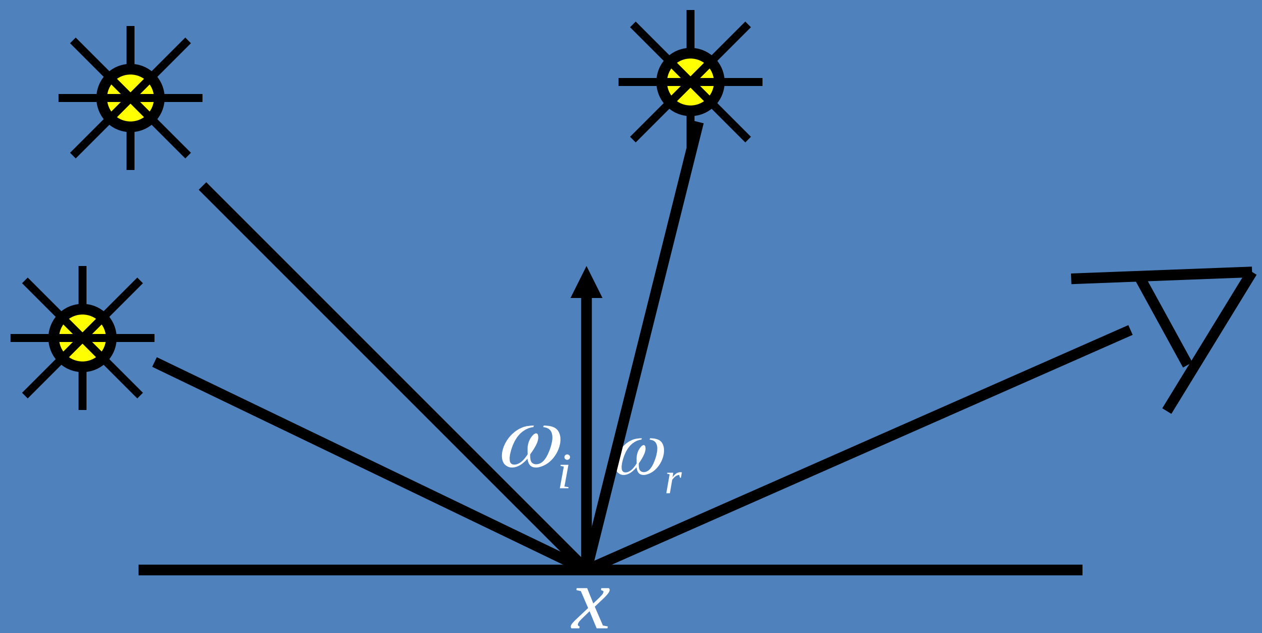


$$L_r(x,\omega_r) = L_e(x,\omega_r) + L_i(x,\omega_i)\, f(x,\omega_i,\omega_r)(\omega_i \bullet n)$$

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

[Slides with help from Pat Hanrahan and Henrik Jensen]

Reflectance Equation

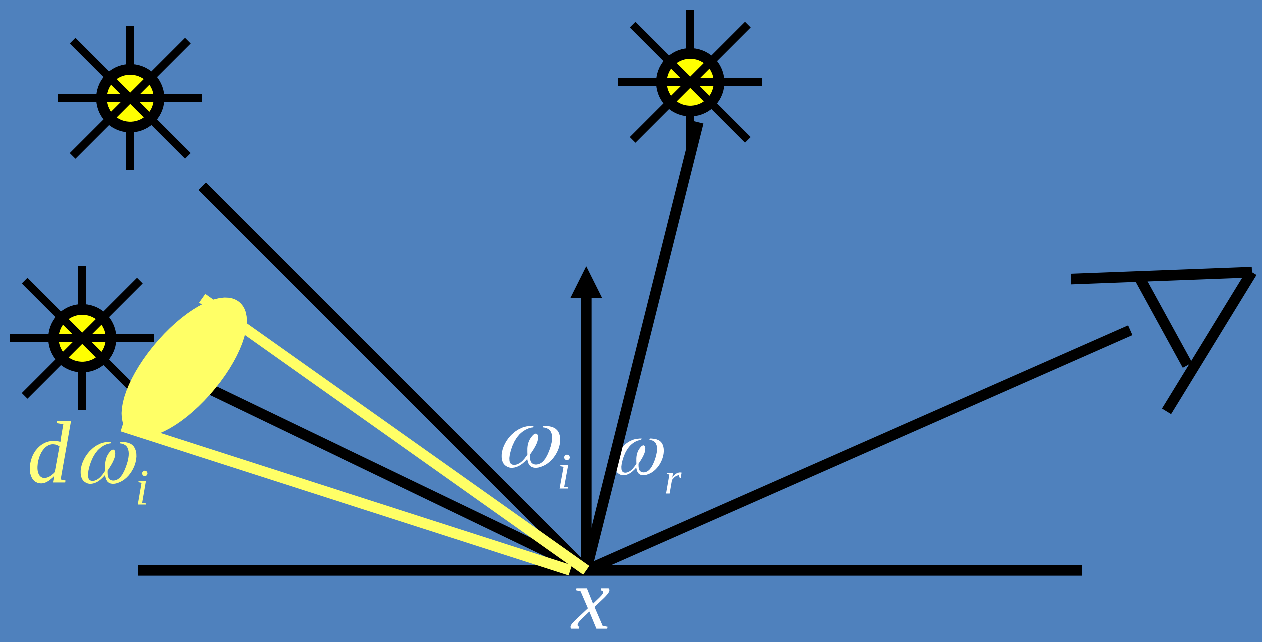


Sum over all light sources

$$L_r(x,\omega_r) = L_e(x,\omega_r) + \sum L_i(x,\omega_i)\, f(x,\omega_i,\omega_r)(\omega_i \bullet n)$$

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

Reflectance Equation

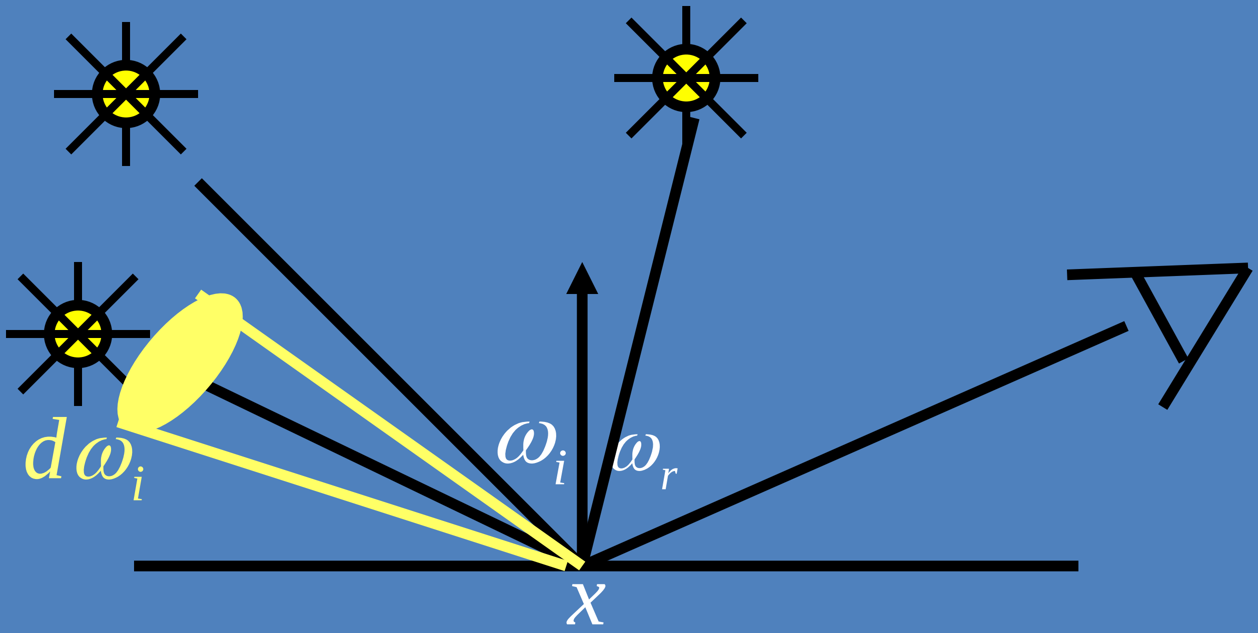


Replace sum with integral

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

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

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

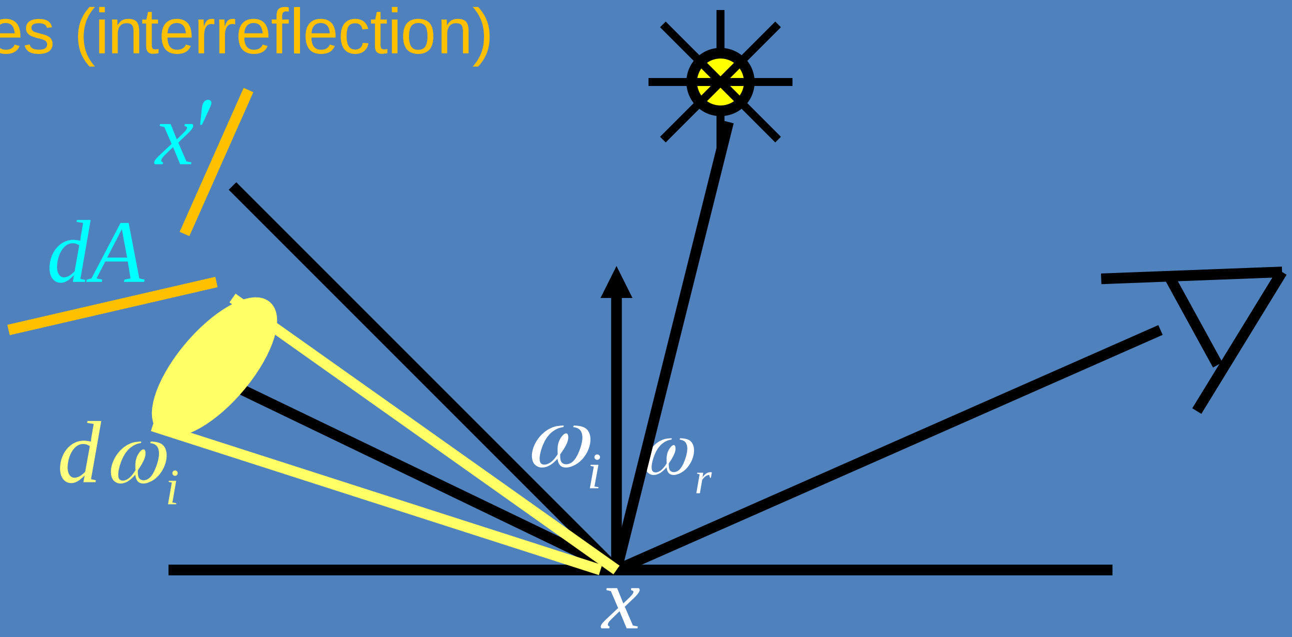
$$L_r(x,\omega_r) = L_e(x,\omega_r) + \int_{\Omega} L_i(x,\omega_i) f(x,\omega_i,\omega_r) \cos\theta_i d\omega_i$$

- Computing reflectance equation requires knowing the incoming radiance from surfaces

- …But determining incoming radiance requires knowing the reflected radiance from surfaces

Global Illumination

Surfaces (interreflection)



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

Reflected Light (Output Image)

Emission

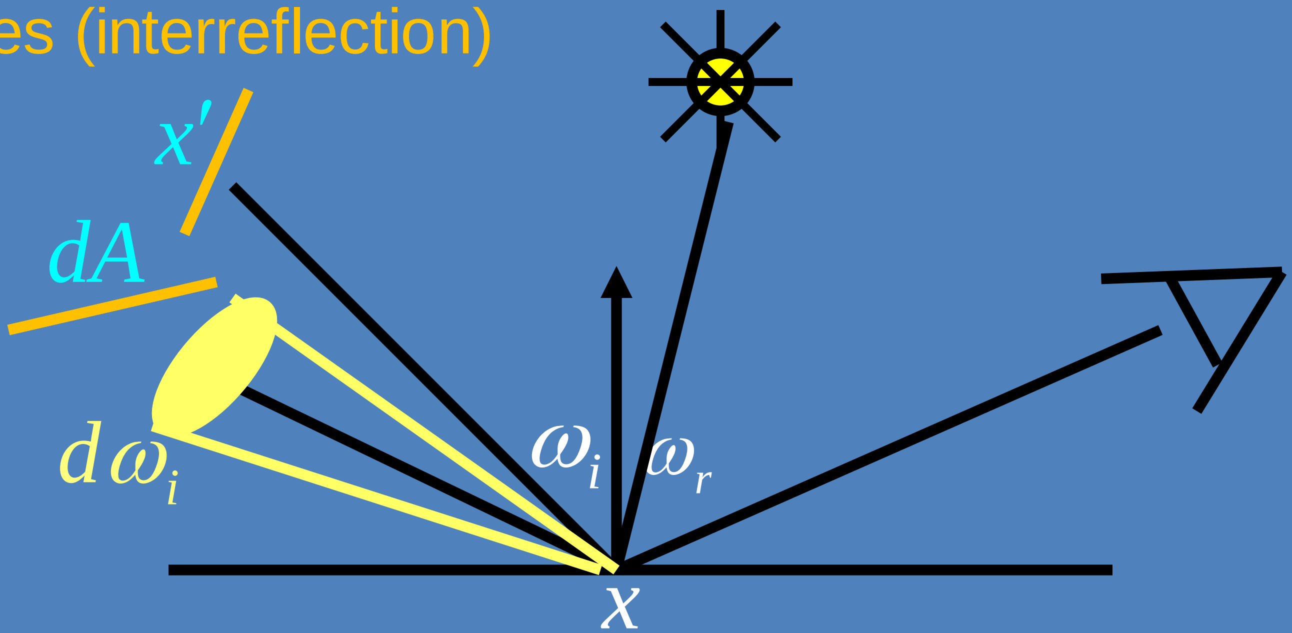
Reflected Light (from prev surface)

BRDF

Cosine of Incident angle

Rendering Equation

Surfaces (interreflection)



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

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

Rendering Equation (Kajiya 1986)

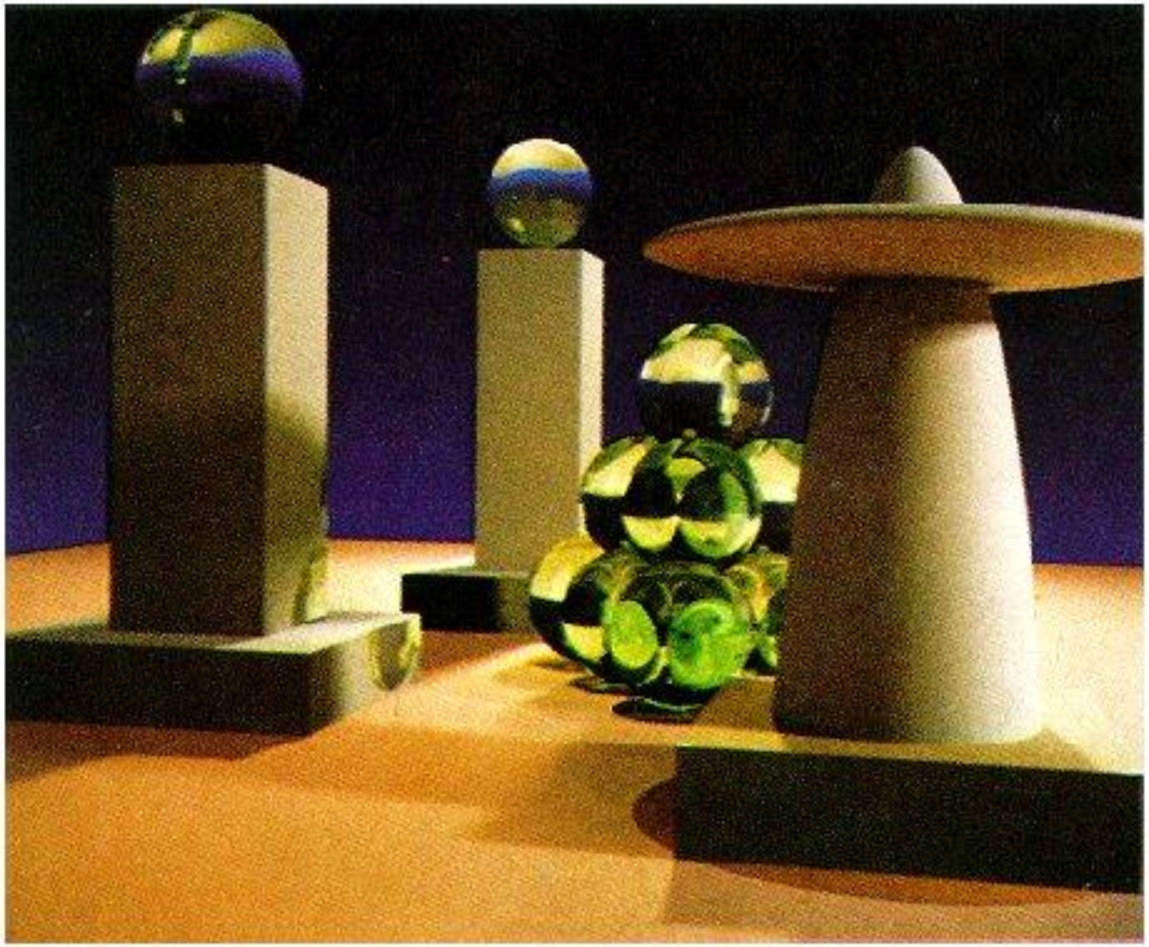


Figure 6. A sample image. All objects are neutral grey. Color on the objects is due to caustics from the green glass balls and color bleeding from the base polygon.

Rendering Equation as Integral 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

Is a Fredholm Integral Equation of second kind [extensively studied numerically] with canonical form

$$l(u) = e(u) + \int l(v)\, K(u,v)\, dv$$

Kernel of equation

Linear Operator Theory 101

Linear operators act on functions like matrices act on vectors or discrete representations

$$h(u) = (M \circ f)(u)$$

M is a linear operator.

f and h are functions of u

$$M \circ (af + bg) = a(M \circ f) + b(M \circ g)$$

a and b are scalars

f and g are functions

Basic linearity relations hold

$$(K \circ f)(u) = \int k(u,v) f(v) dv$$

$$(D \circ f)(u) = \frac{\partial f}{\partial u}(u)$$

(e.g., integration and differentiation)

Linear Operator Equation

$$l(u) = e(u) + \int l(v) \boxed{K(u,v) dv}$$

Kernel of equation

$$L = E + KL$$

which is effectively a simple matrix equation (or system of simultaneous linear equations) where

L, E are vectors,

K is the light transport matrix (more on this later!)

Solving the Rendering Equation (=how to compute L?)

- In general, too hard for analytic solution
- But there are approximations and some nice observations…

Solving the Rendering Equation (=how to compute L?)

$$L = E + KL$$

$$IL - KL = E$$

$$(I - K)L = E$$

$$L = (I - K)^{-1} E$$

(using Binomial Theorem)

$$L = (I + K + K^2 + K^3 + \ldots)E$$

$$L = E + KE + K^2 E + K^3 E + \ldots$$

where term n corresponds to n-th bounces of light

Ray Tracing

$$L = E + KE + K^2E + K^3E + \ldots$$

- E: Emission directly From light sources
- KE: Direct Illumination on surfaces
- K^2E: Global Illumination (One bounce indirect) [Mirrors, Refraction]
- K^3E: (Two bounce indirect) [Caustics, etc…]

Ray Tracing

$$L = E + KE + K^2E + K^3E + \ldots$$

Emission directly
From light sources

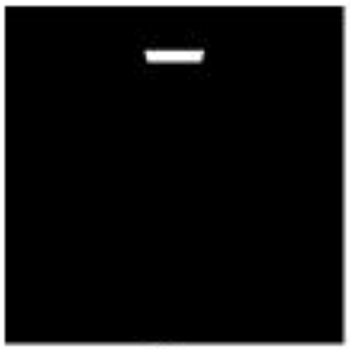
Direct Illumination
on surfaces

Global Illumination
(One bounce indirect)
[Mirrors, Refraction]

(Two bounce indirect)
[Caustics, etc…]

OpenGL
Shading

Successive Approximation



L_e

$K \circ L_e$

$K \circ K \circ L_e$

$K \circ K \circ K \circ L_e$

L_e

$L_e + K \circ L_e$

$L_e + \cdots K^2 \circ L_e$

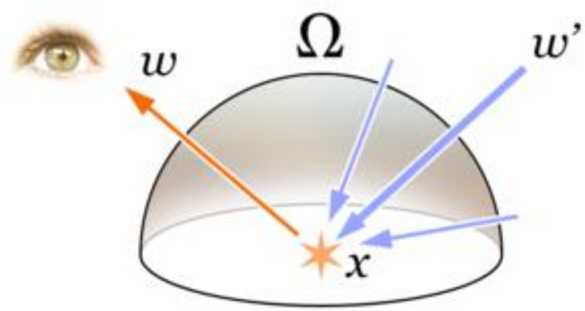
$L_e + \cdots K^3 \circ L_e$

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

$$L_o(\mathbf{x},\omega,\lambda,t) = L_e(\mathbf{x},\omega,\lambda,t) + \int_\Omega f_r(\mathbf{x},\omega',\omega,\lambda,t) L_i(\mathbf{x},\omega',\lambda,t)(-\omega'\cdot\mathbf{n})d\omega'$$

(rendering equation)



[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

- For this reason, the radiosity method was invented.

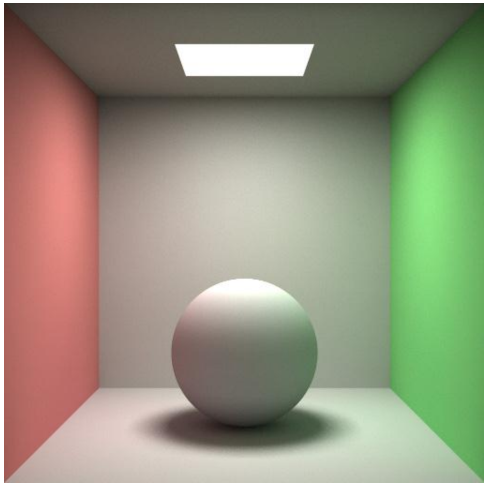
- The main idea of the method is

  to store illumination values on the surfaces of the objects, as the light is propagated starting at the light sources.

Radiosity

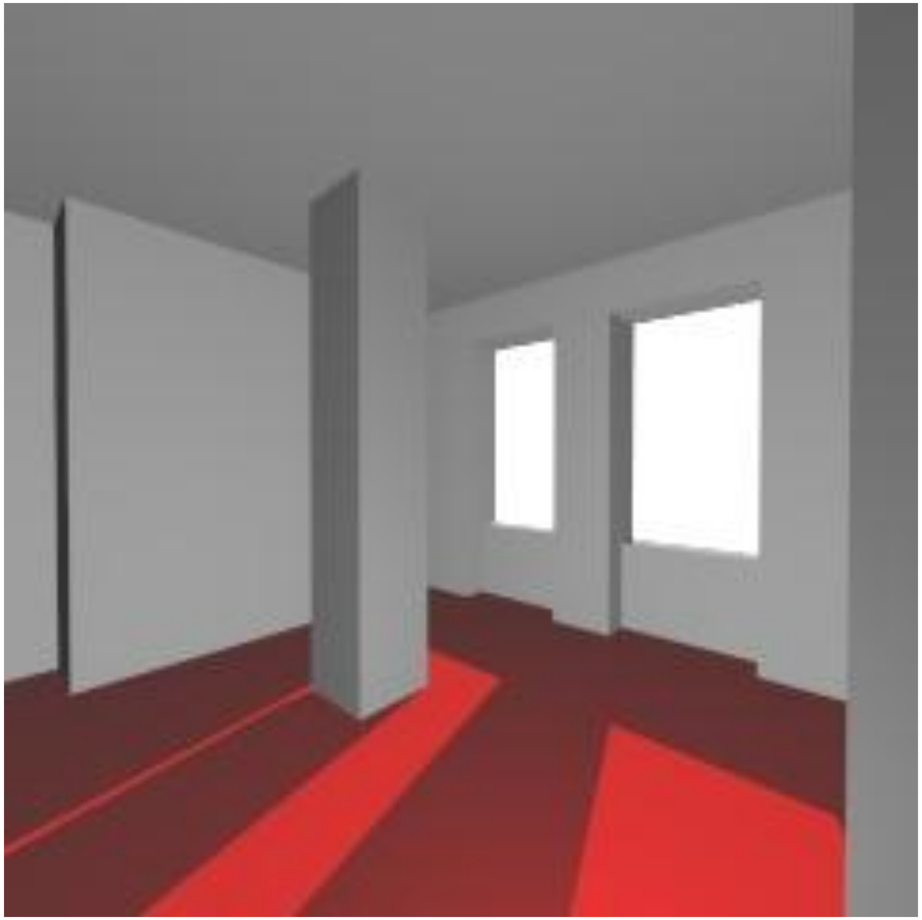
- Equation: $B_i \, dA_i = E_i \, dA_i + R_i \int_j B_j F_{ji} \, dA_j$

(more details on the board...)



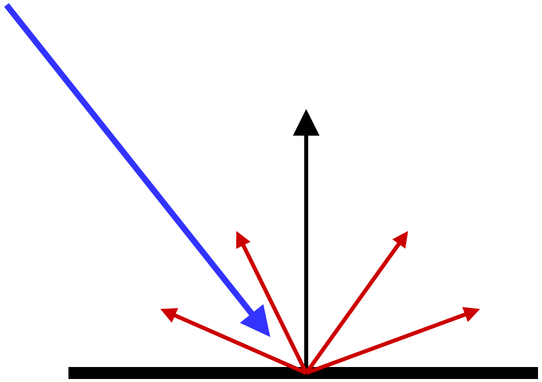
Ray Tracin


Radiosit



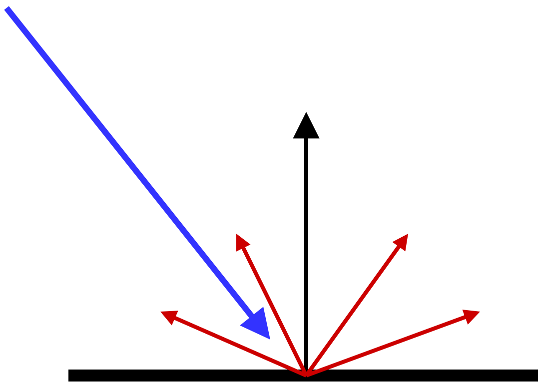
- Diffuse Interreflection

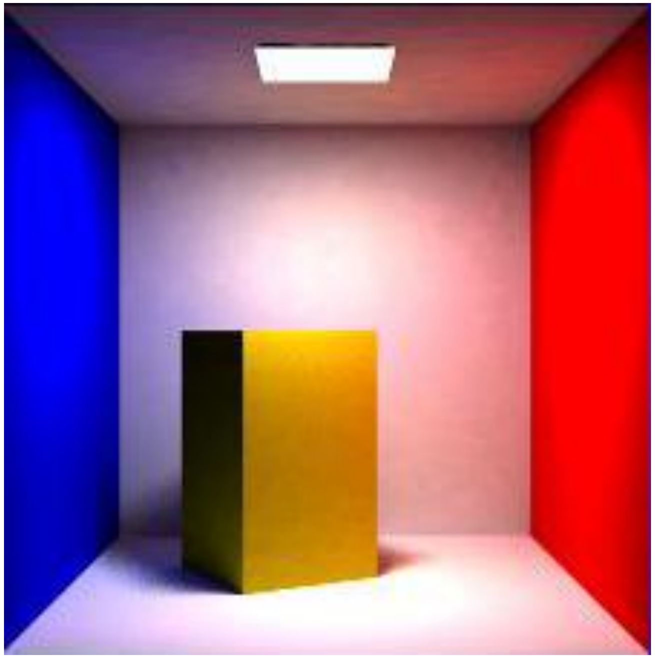
Diffuse Interreflection



- Surface = "diffuse reflector" of light energy,
- means: any light energy which strikes the surface will be reflected in all directions,
- dependent only on the angle between the surface's normal and the incoming light vector (Lambert's law).

Diffuse Interreflection

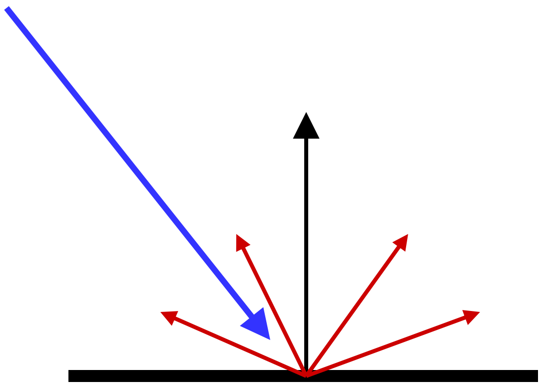




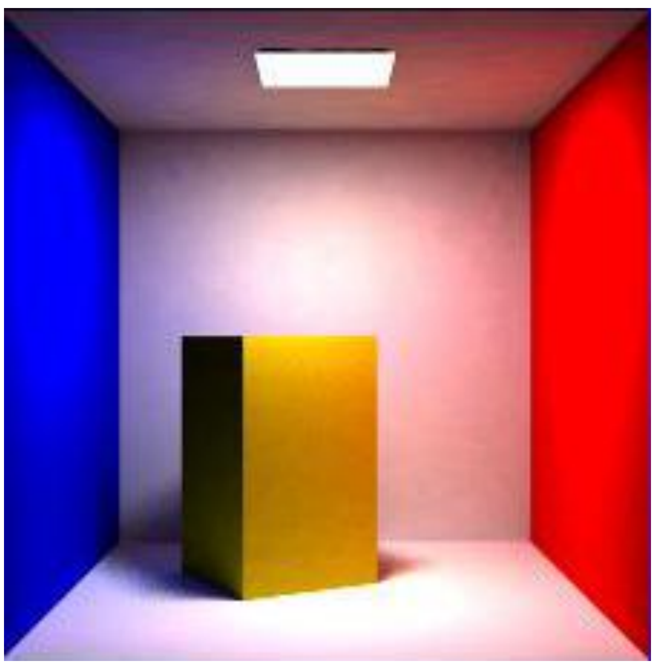
- The reflected light energy often is colored, to some small extent, by the color of the surface from which it was reflected.

- This reflection of light energy in an environment produces a phenomenon known as "color bleeding," where a brightly colored surface's color will "bleed" onto adjacent surfaces.

Diffuse Interreflection



- The reflected light energy often is colored, to some small extent, by the color of the surface from which it was reflected.



"Color bleeding", as both the red and blue walls "bleed" their color onto the white walls, ceiling and floor.

Radiosity (Thermal Heat Transfer)

- The "radiosity" method has its basis in the field of thermal heat transfer.

- Heat transfer theory describes radiation as the transfer of energy from a surface when that surface has been thermally excited.

- This encompasses both surfaces which are basic emitters of energy, as with light sources, and surfaces which receive energy from other surfaces and thus have energy to transfer.

- This "thermal radiation" theory can be used to describe the transfer of many kinds of energy between surfaces, including light energy.

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.

The Radiosity Equation

- The "radiosity equation" describes the **amount of energy** which can be emitted from a surface, as the sum of the energy inherent in the surface (a light source, for example) and the energy which strikes the surface, being emitted from some other surface.

- The energy which leaves a surface (surface "j") and strikes another surface (surface "i") is attenuated by two factors:
  - the **"form factor"** between surfaces "i" and "j", which accounts for the physical relationship between the two surfaces
  - the **reflectivity of surface "i"**, which will absorb a certain percentage of light energy which strikes the surface.

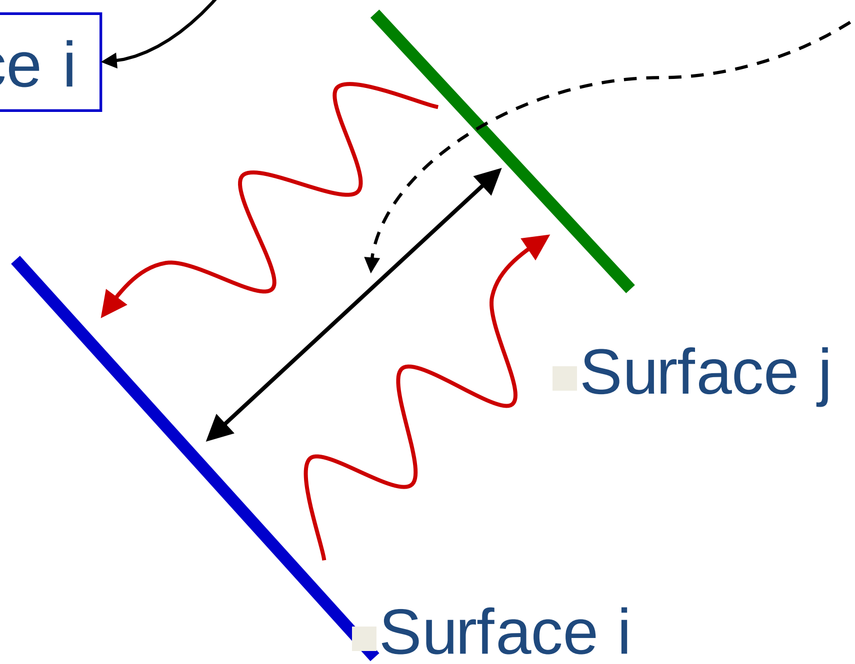
The Radiosity Equation

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

- 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

will <u>absorb</u> a certain percentage of light energy which strikes the surface

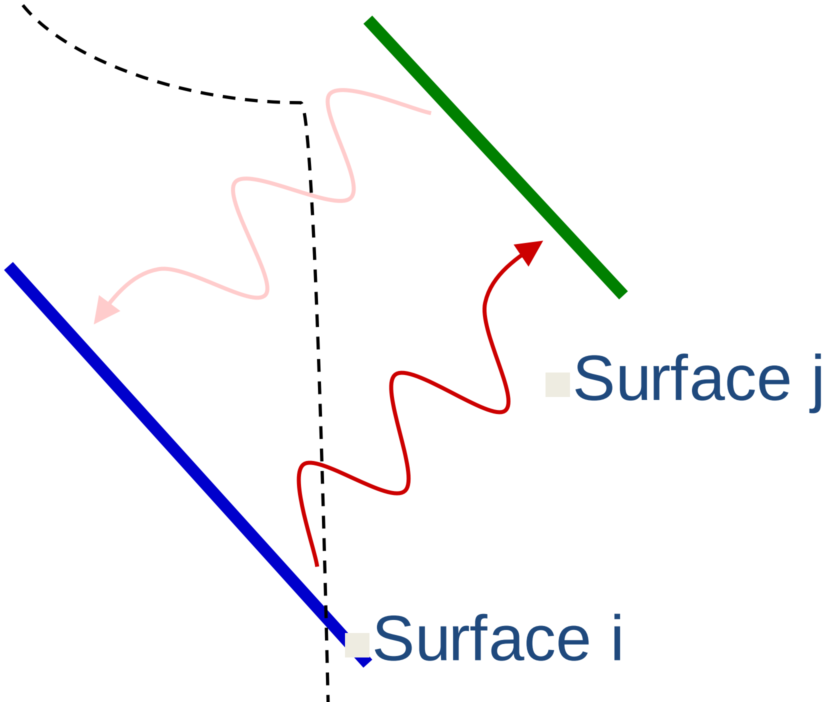
accounts for the <u>physical relationship</u> between the two surfaces



The Radiosity Equation

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

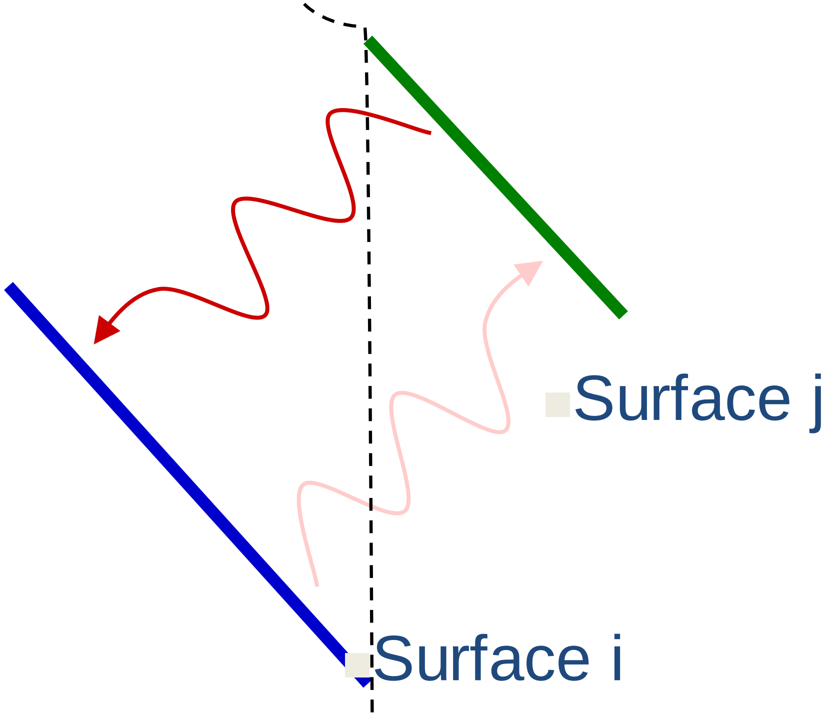
Energy emitted by surface i



The Radiosity Equation

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

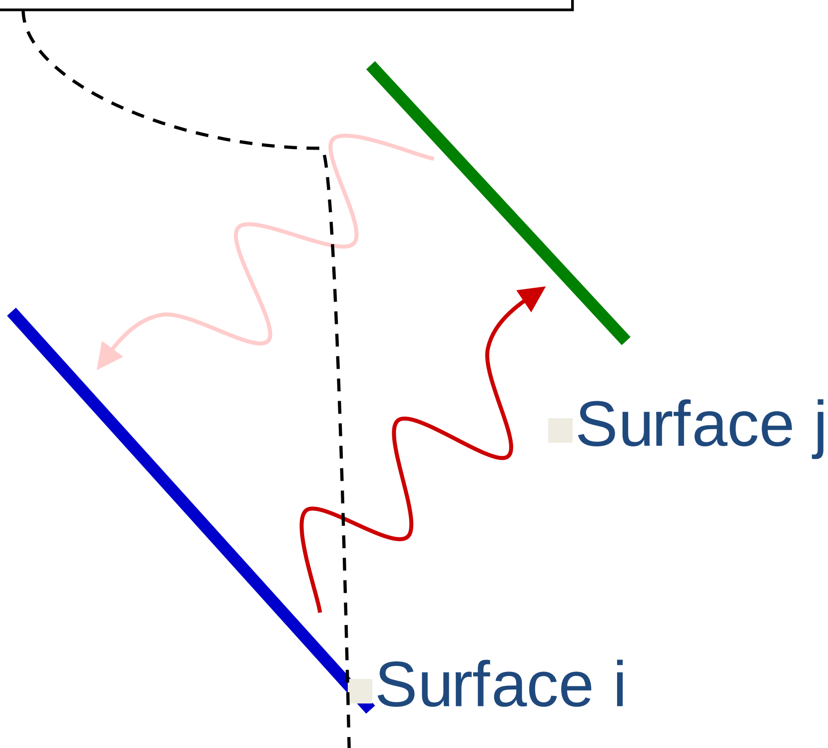
Energy reaching surface i from other surfaces



The Radiosity Equation

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

Energy reflected by surface i



Radiosity

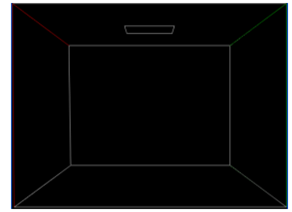
- **Classic radiosity = finite element method**

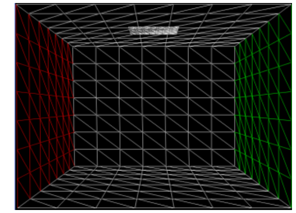
- **Assumptions**
  - **Diffuse reflectance**
  - **Usually polygonal surfaces**

- **Advantages**
  - **Soft shadows and indirect lighting**
  - **View independent solution**
  - **Precompute for a set of light sources**
  - **Useful for walkthroughs**

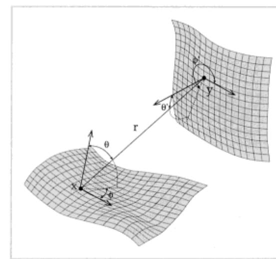
Classic Radiosity Algorithm

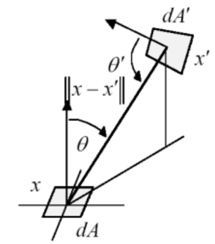
Mesh Surfaces into Elements



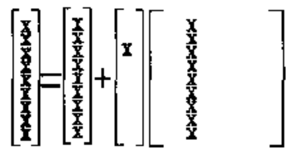


Compute Form Factors Between Elements

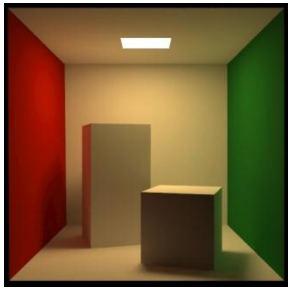




Solve Linear System for Radiosities

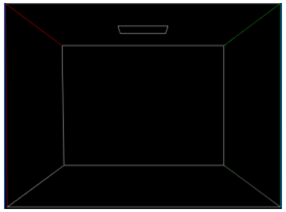


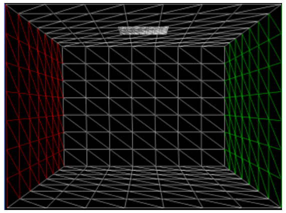
Reconstruct and Display Solution



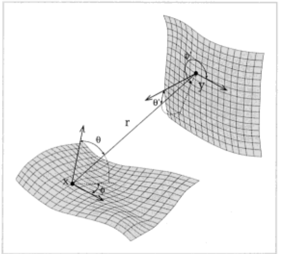
Classic Radiosity Algorithm

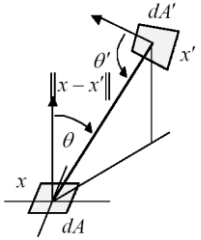
Mesh Surfaces into Elements



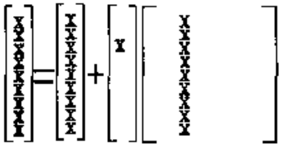


Compute Form Factors Between Elements

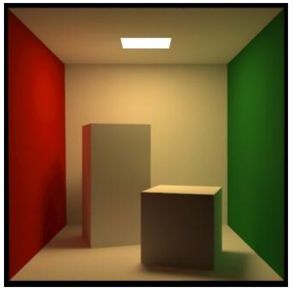




Solve Linear System for Radiosities



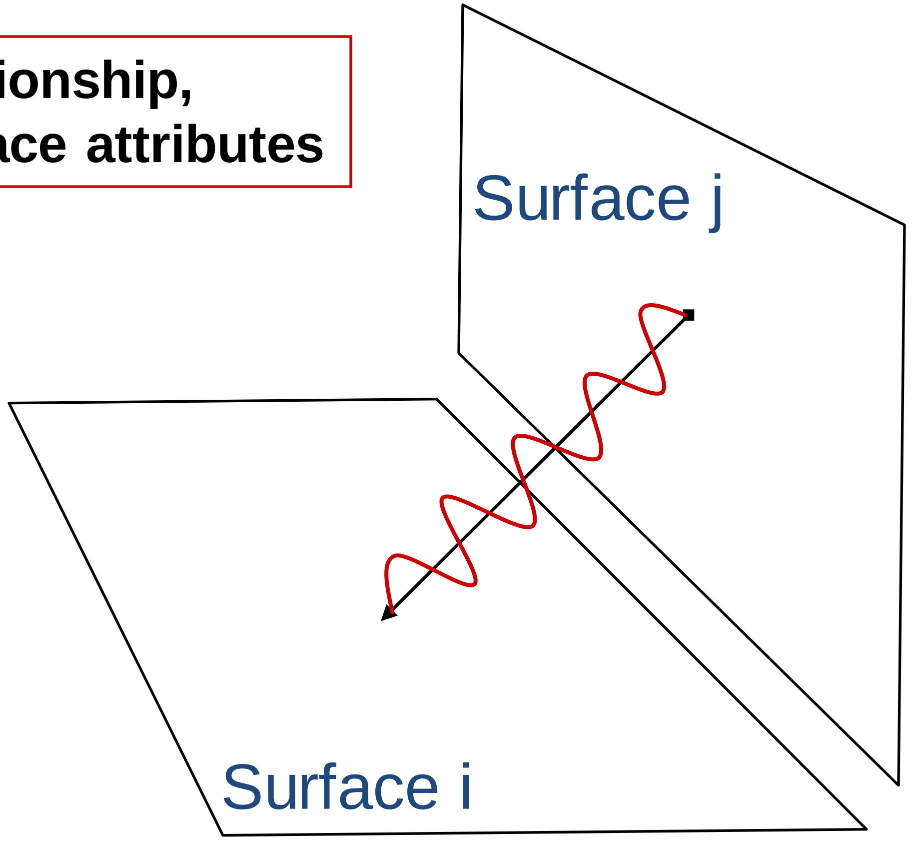
Reconstruct and Display Solution



The Form Factor:

The <u>fraction</u> of energy leaving one surface that reaches another surface

It is a purely geometric relationship, independent of viewpoint or surface attributes



- Between <u>differential areas</u>, the form factor equals:

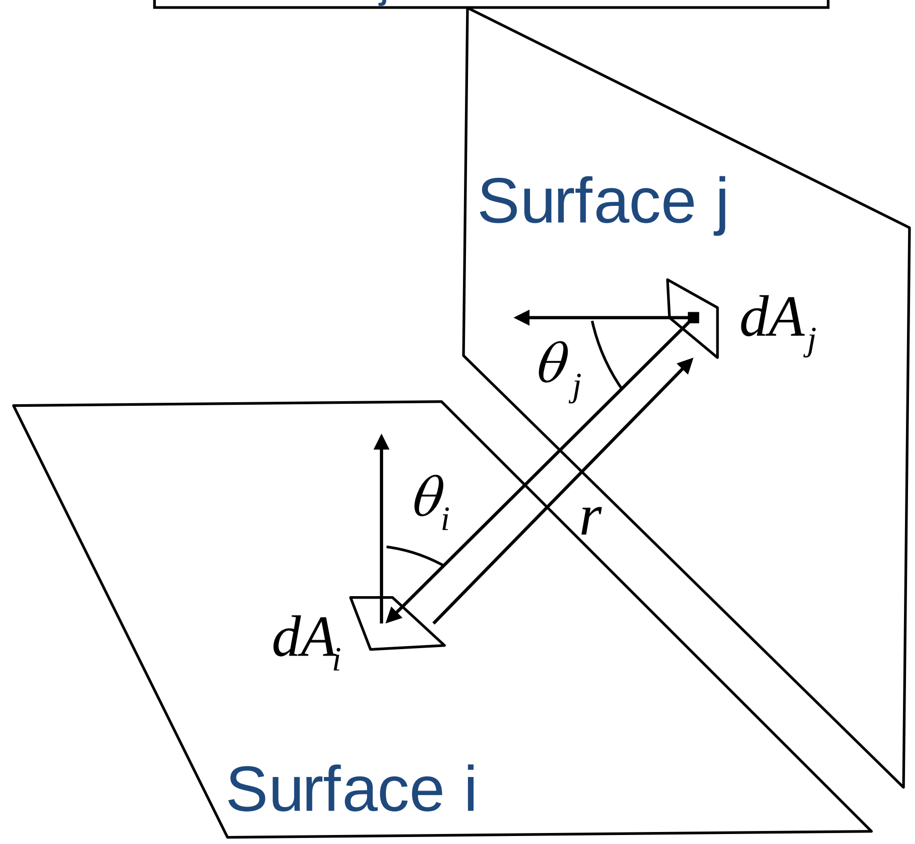
$$F_{dA_i dA_j} = \frac{\cos\theta_i \cos\theta_j}{\pi |r|^2}$$

differential area of surface i, j

angle between Normal$_i$ and r

angle between Normal$_j$ and r

vector from dA_i to dA_j

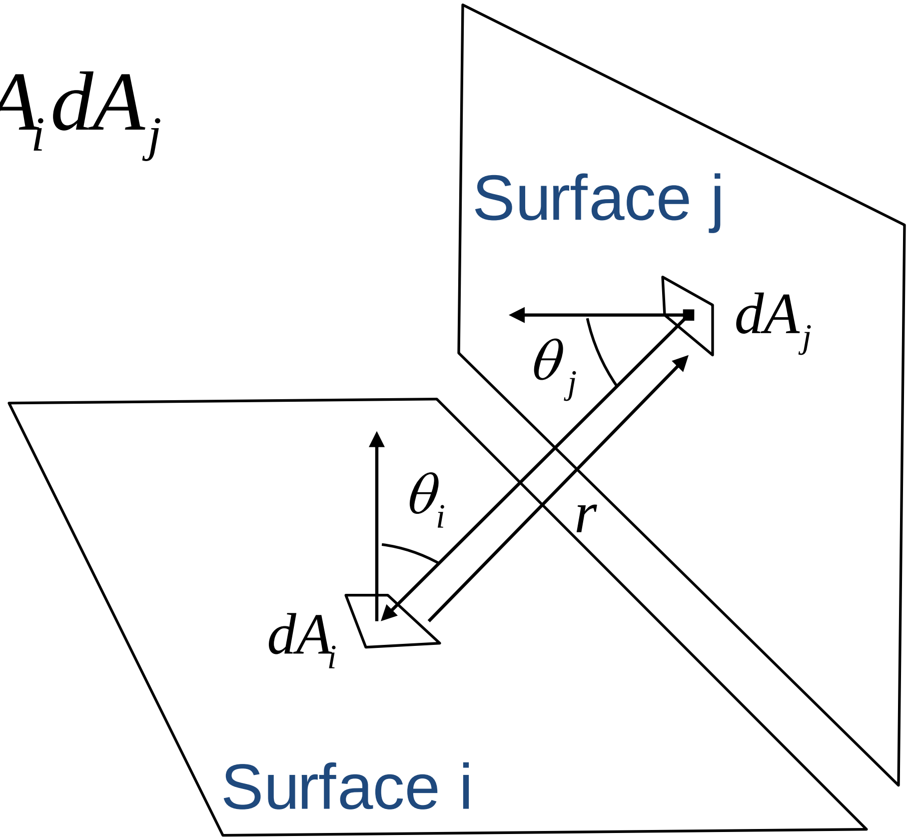


Between differential areas, the form factor equals:

$$F dA_j dA_j = \frac{\cos\theta_i \cos\theta_j}{\pi |r|^2}$$

The overall form factor between i and j is found by integrating

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



Next Step:
Learn ways of computing **form factors**

- Recall the Radiosity Equation:

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

- The F_{ij} are the form factors

- Form factors independent of radiosities (depend only on scene geometry)

Form Factors in (More) Detail

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

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

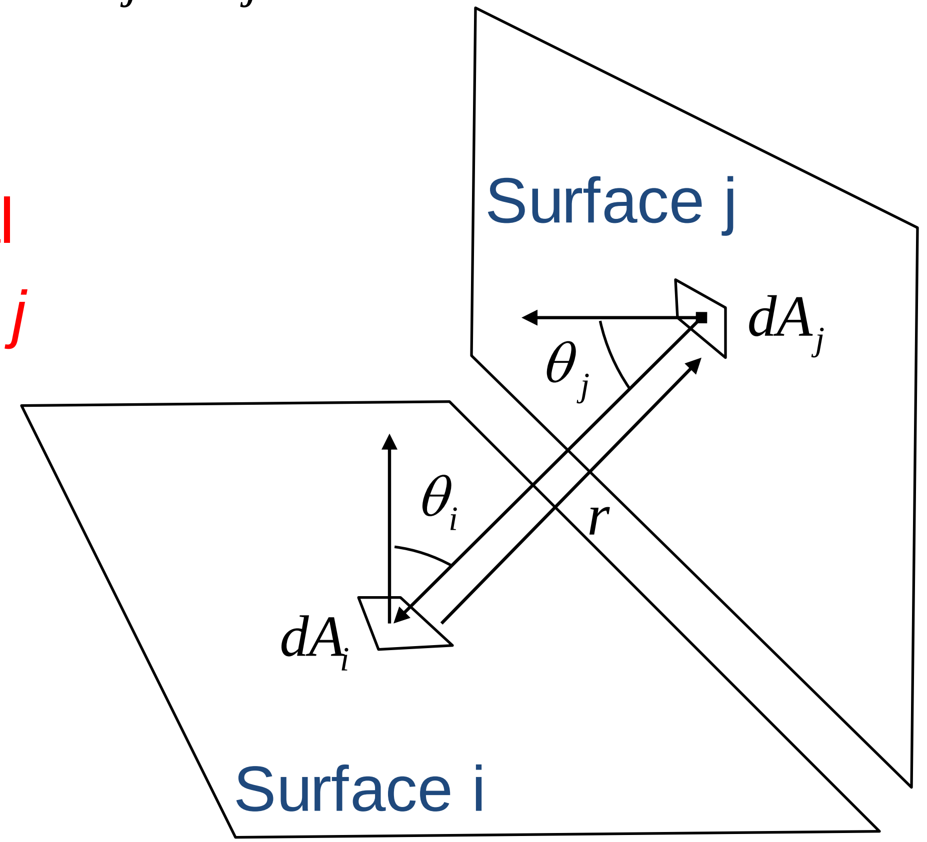
where V_{ij} is the visibility (0 or 1)

We have two integrals to compute:

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

Area integral over surface *i*

Area integral over surface *j*



The Nusselt Analog

- Differentiation of the basic form factor equation is difficult even for simple surfaces!
- Nusselt developed a geometric analog which allows the simple and accurate calculation of the form factor between a surface and a point on a second surface.

The Nusselt Analog

- The "Nusselt analog" involves placing a hemispherical projection body, with unit radius, at a point on a surface.
- The second surface is spherically projected onto the projection body, then cylindrically projected onto the base of the hemisphere.
- The form factor is, then, the area projected on the base of the hemisphere divided by the area of the base of the hemisphere.

Numerical Integration: The Nusselt Analog

This gives the form factor F_{dAiAj}

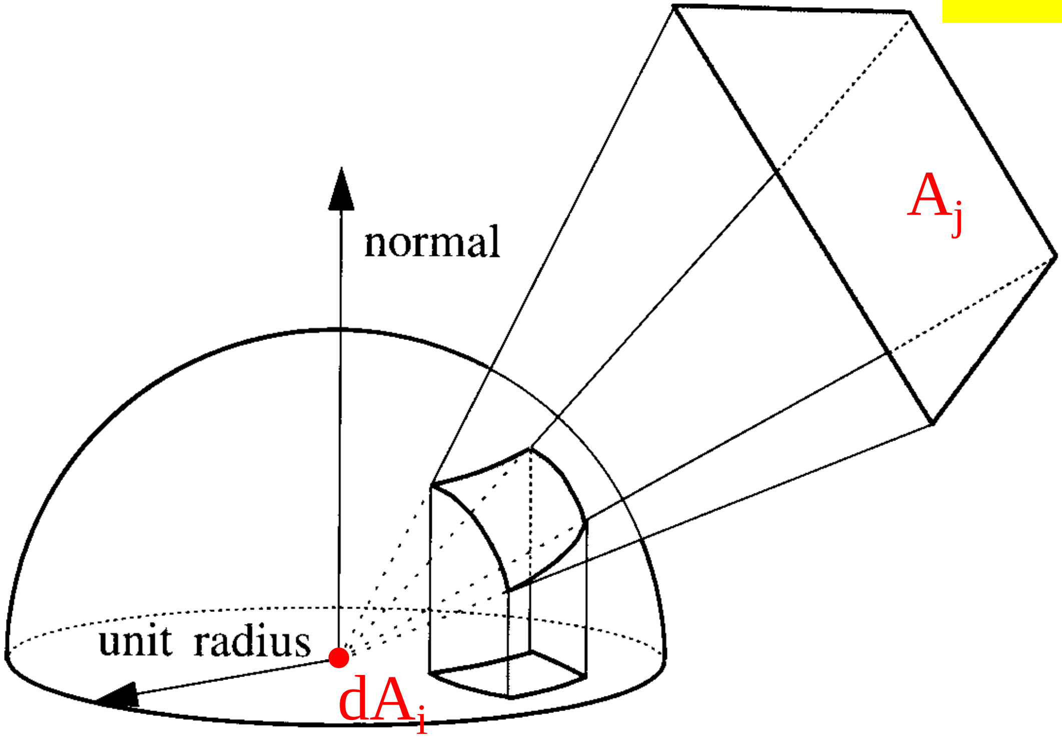
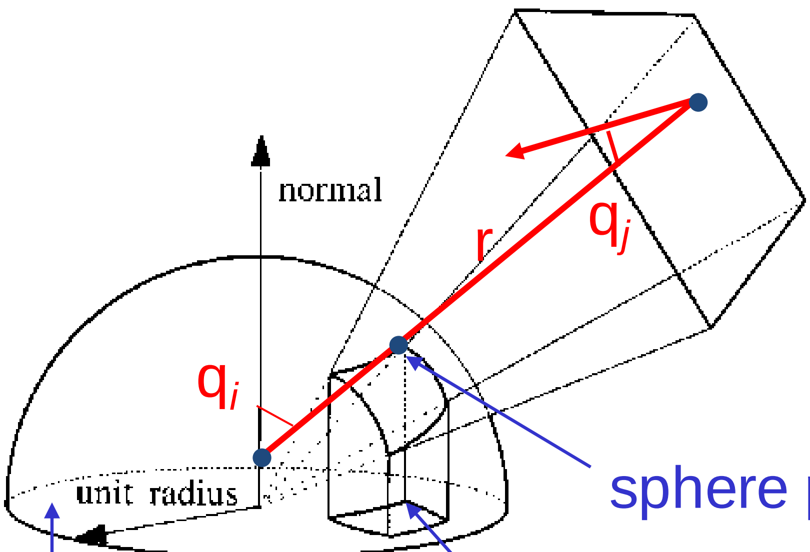


Figure 4.8: *Nusselt analog. The form factor from the differential area dA_i to element A_j is proportional to the area of the double projection onto the base of the hemisphere.*

The Nusselt Analog

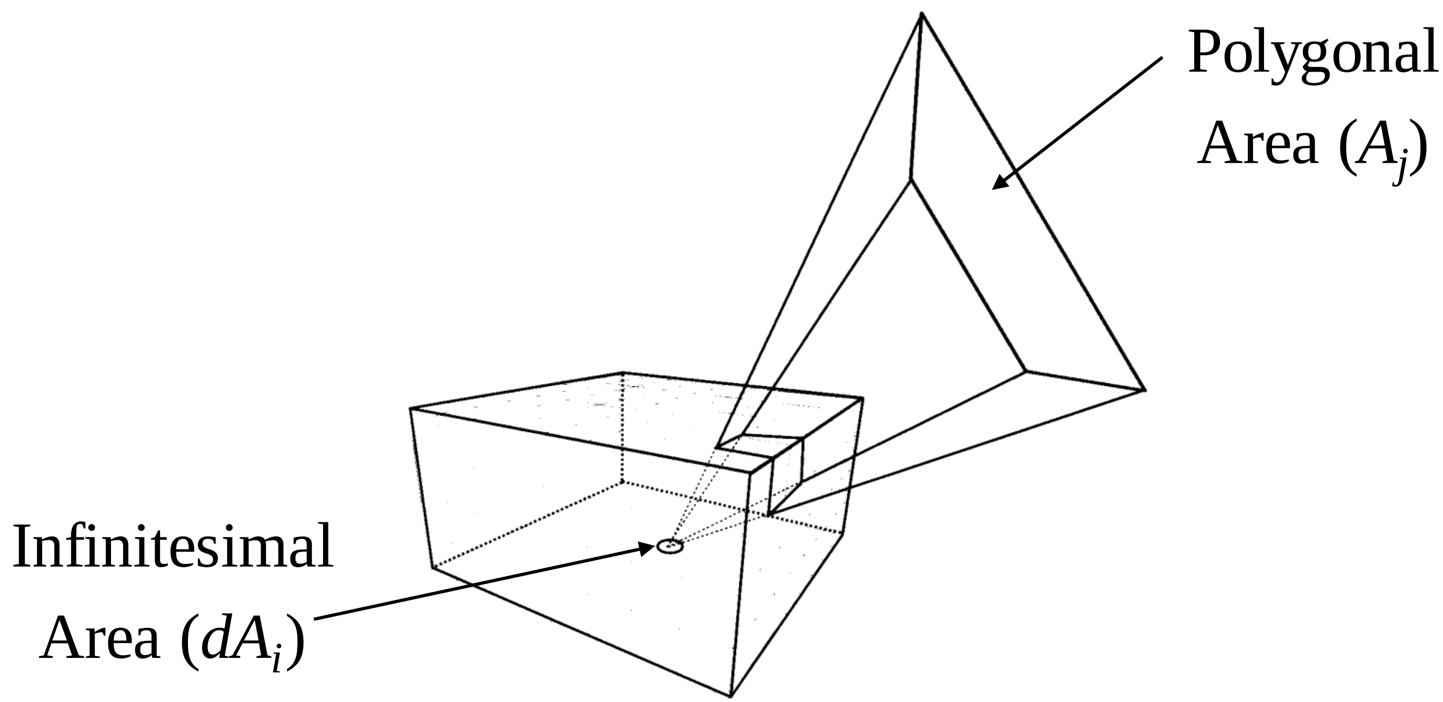
1. Project A_j along its normal:
   $A_j \cos q_j$
2. Project result on sphere:
   $A_j \cos q_j / r^2$
3. Project result on unit circle:
   $A_j \cos q_j \cos q_i / r^2$
4. Divide by unit circle area:
   $A_j \cos q_j \cos q_i / p r^2$
5. Integrate for all points on A_j:

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



Method 1: Hemicube

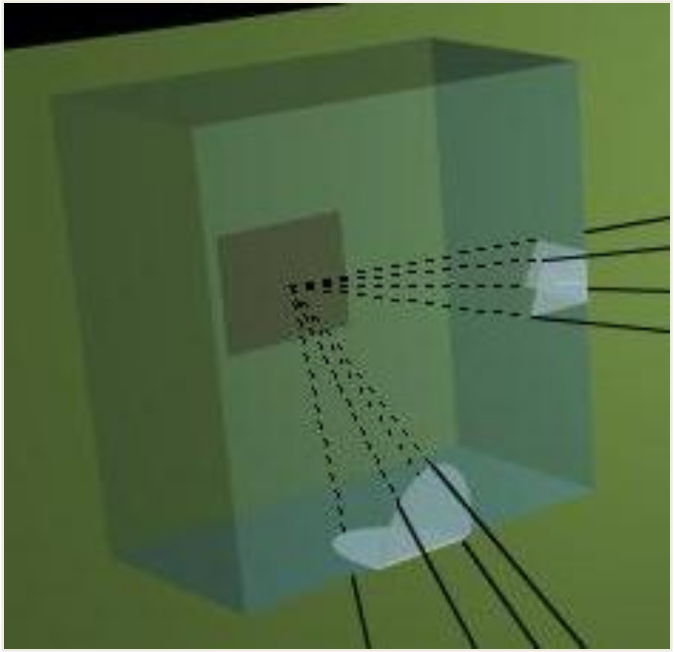
- Approximation of Nusselt's analog between a point dA_i and a polygon A_j



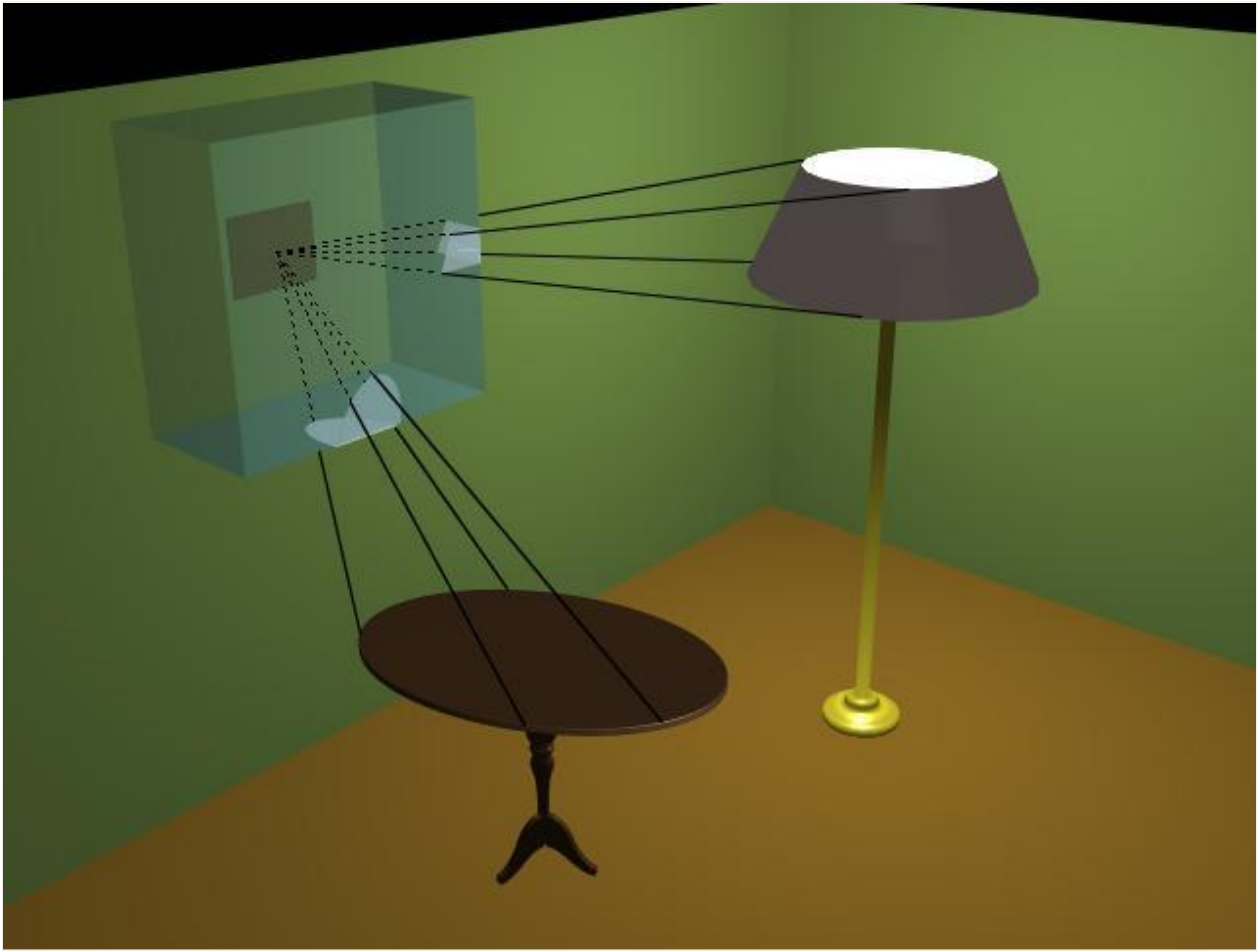
Hemicube

- For convenience, a cube 1 unit high with a top face 2 x 2 is used. Side faces are 2 wide by 1 high.
- Decide on a <u>resolution</u> for the cube.
  Say 512 by 512 for the top.

The Hemicube In Action

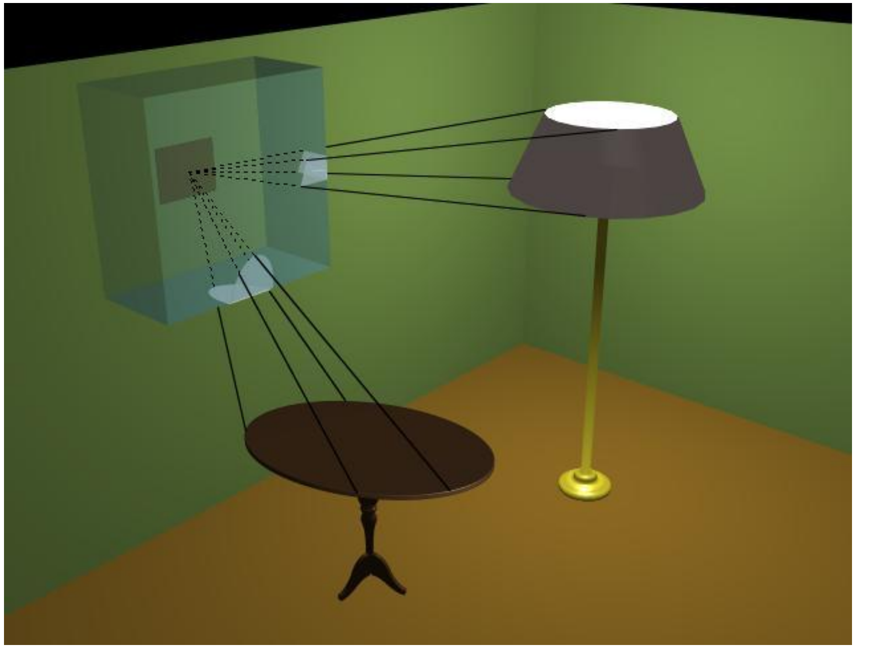


The Hemicube In Action



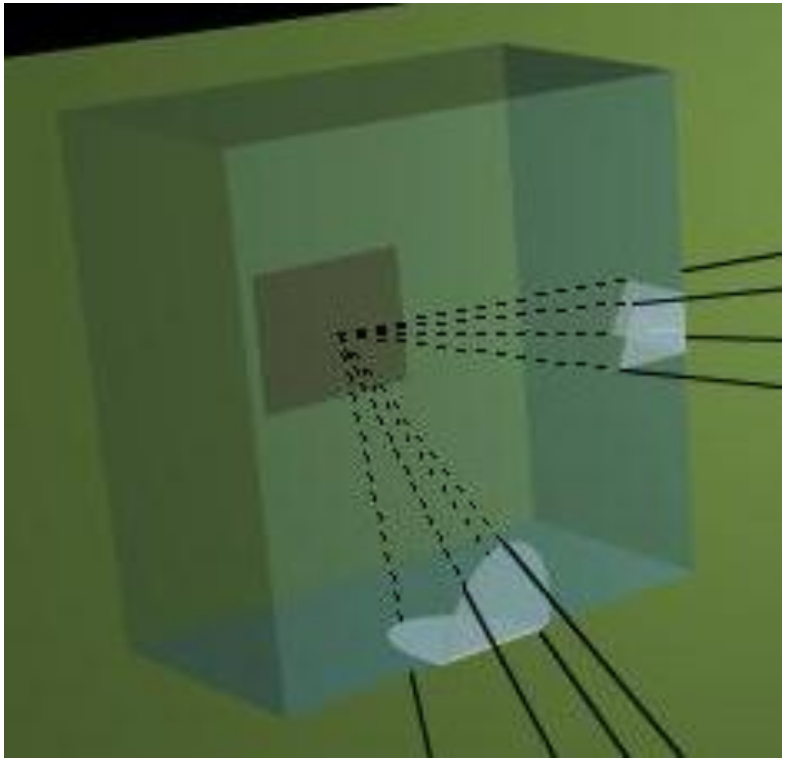
The Hemicube In Action

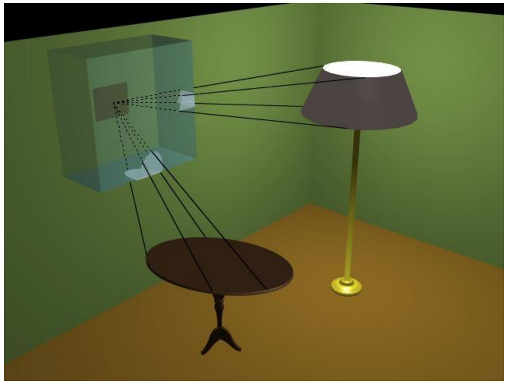
- This illustration demonstrates the calculation of form factors between a particular surface on the wall of a room and several surfaces of objects in the room.



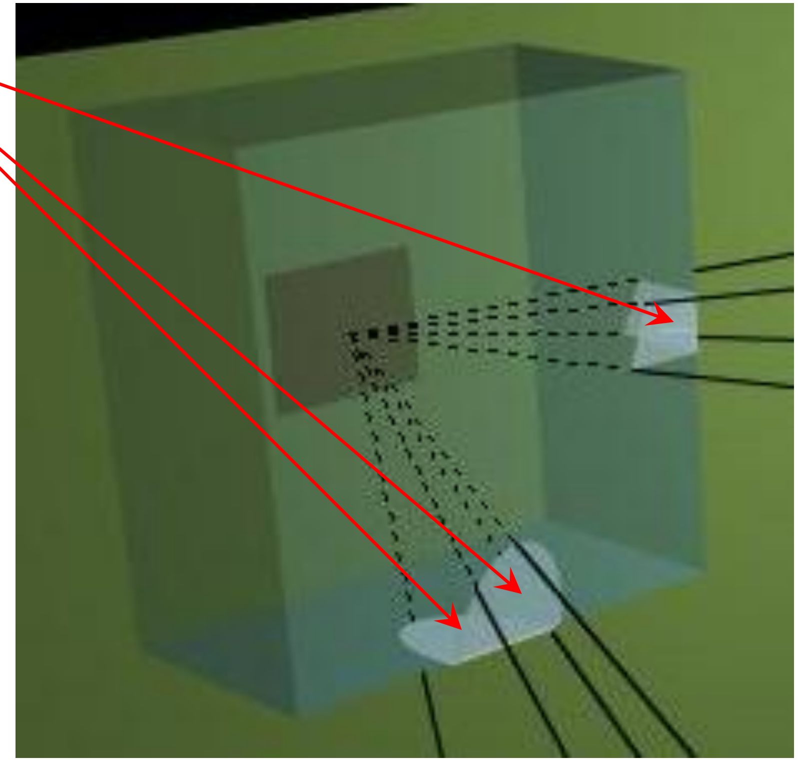
Compute the form factors from a point on a surface to all other surfaces by:

- Projecting all other surfaces onto the hemicube
- Storing, at each discrete area, the identifying index of the surface that is closest to the point.



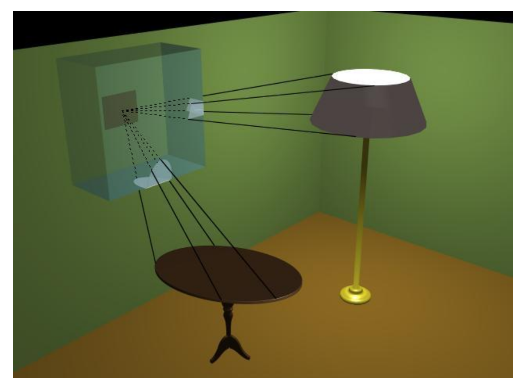


Discrete areas with the indices of the surfaces which are ultimately visible to the point.



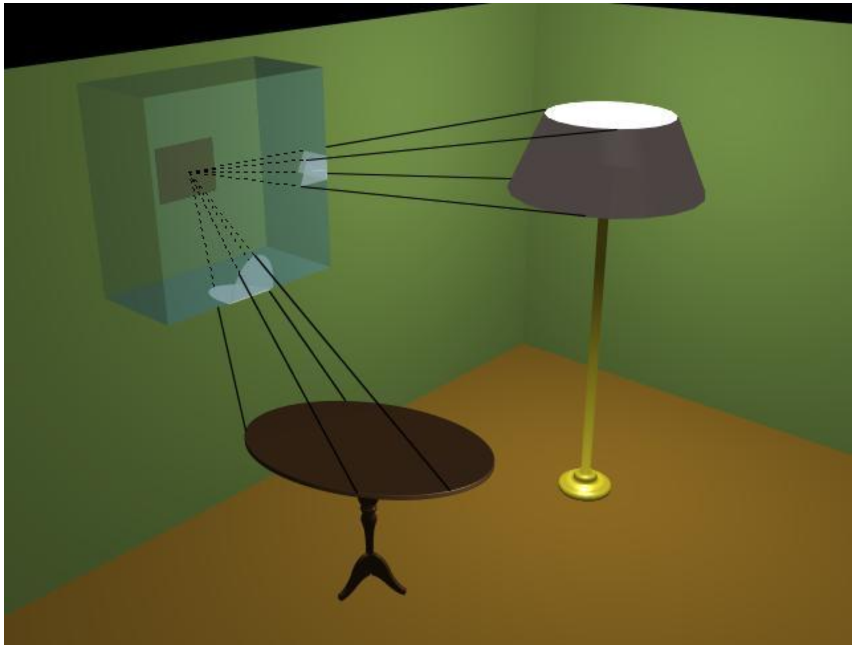
From there the form factors between the point and the surfaces are calculated.

For greater accuracy, a large surface would typically be broken into a set of small surfaces before any form factor calculation is performed.



Hemicube Method

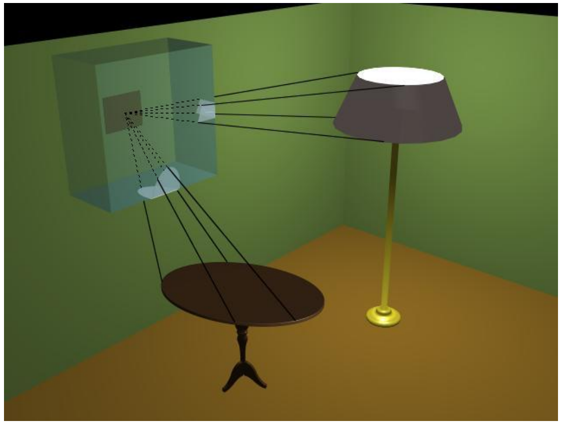
1. Scan convert all scene objects onto hemicube's 5 faces
2. Use Z buffer to determine visibility term
3. Sum up the delta form factors of the hemicube cells covered by scanned objects
4. Gives form factors from hemicube's base to all elements,
   i.e. F_{dAiAj} for given i and all j



Hemicube Algorithms

Advantages

+ First practical method
+ Use existing rendering systems; Hardware
+ Computes row of form factors in *O(n)*



Disadvantages

- Computes differential-finite form factor
- Aliasing errors due to sampling

  Randomly rotate/shear hemicube

- Proximity errors
- Visibility errors
- Expensive to compute a single form factor

Method 2: Area Sampling

Subdivide A_j into small pieces dA_j

For all dA_j

cast ray $dAj\text{-}dAj$ to determine V_{ij}

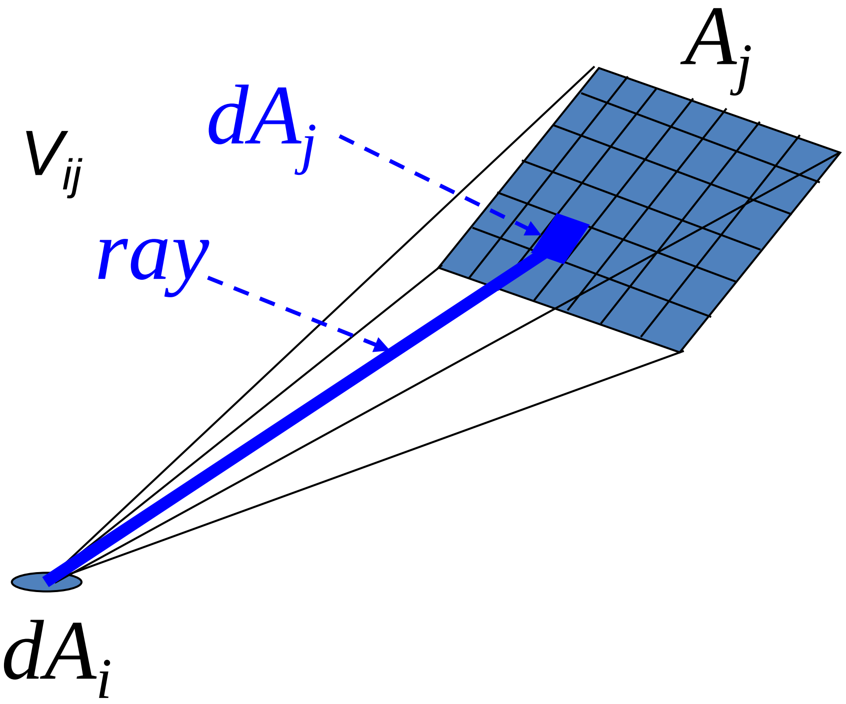
if visible

compute F_{dAidAj}

$$F_{dA_i dA_j} = \frac{\cos\theta_i \cos\theta_j}{\pi r^2} V_{ij} dA_j$$

sum up

$F_{dAiAj} \mathrel{+}= F_{dAidAj}$



We have now F_{dAiAj}

Summary

- Several ways to find form factors

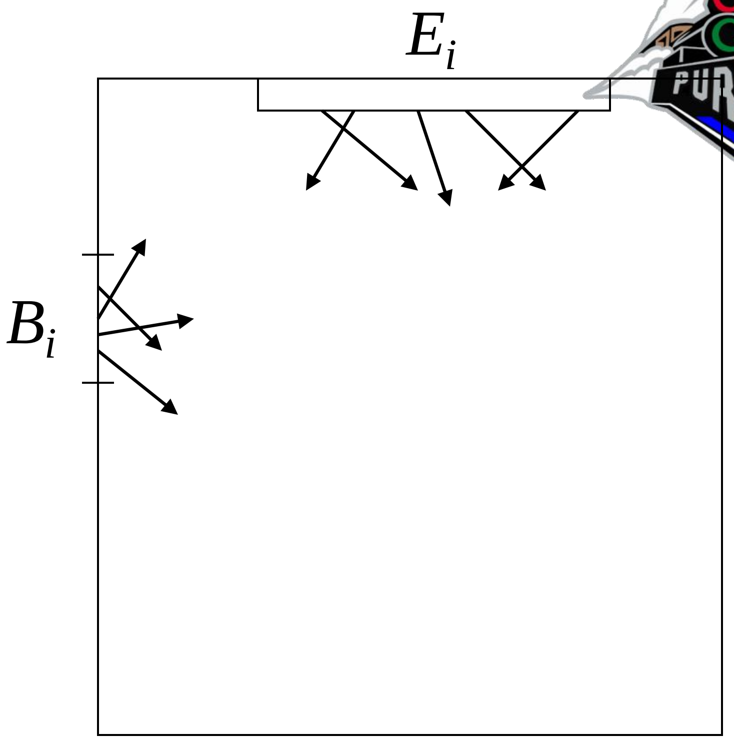
- Hemicube was original method
  + Hardware acceleration
  + Gives $F_{dA_iA_j}$ for all j in one pass
  - Aliasing

- Area sampling methods now preferred
  → Slower than hemicube
  → As accurate as desired since adaptive

Next

- We have the form factors
- How do we find the radiosity solution for the scene?
  - The "Full Matrix" Radiosity Algorithm
  - Gathering & Shooting
  - Progressive Radiosity
- Meshing

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.

Radiosity Matrix

- This method produces a complete solution, at the substantial cost of
  - first calculating form factors between each pair of surfaces
  - and then the solution of the matrix equation.

- This leads to substantial costs not only in computation time but in storage.

Next

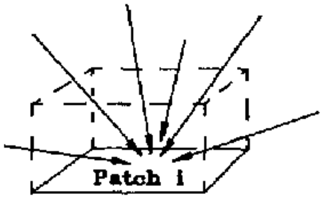
- We have the form factors
- How do we find the radiosity solution for the scene?
  - The "Full Matrix" Radiosity Algorithm
  - Gathering & Shooting
  - Progressive Radiosity
- Meshing

Solve [F][B] = [E]

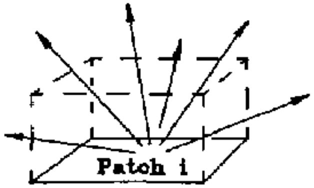
- **Direct methods: *$O(n^3)$***
  - Gaussian elimination
    - Goral, Torrance, Greenberg, Battaile, 1984
- **Iterative methods: *$O(n^2)$***

Energy conservation
→¨diagonally dominant¨ → iteration converges

  - Gauss-Seidel, Jacobi: Gathering
    - Nishita, Nakamae, 1985
    - Cohen, Greenberg, 1985

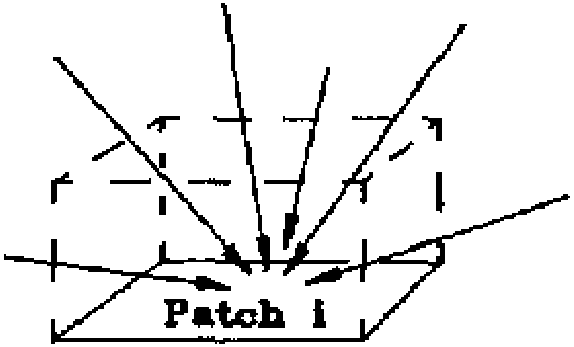


  - Southwell: Shooting
    - Cohen, Chen, Wallace, Greenberg, 1988

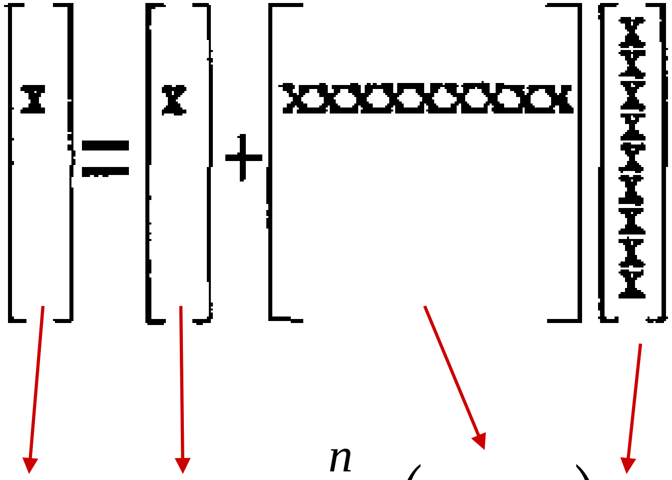


Gathering

- In a sense, the light leaving patch i is determined by *gathering* in the light from the rest of the environment



GATHERING

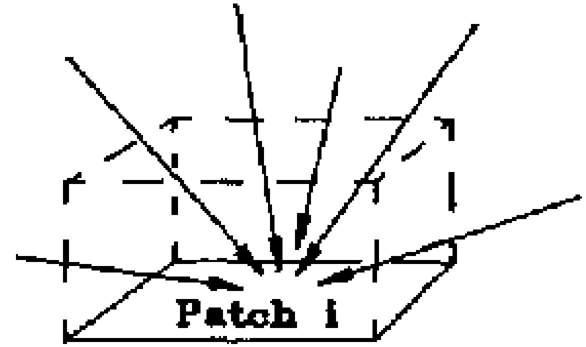


$$B_i = E_i + \rho_i \sum_{j=1}^{n} B_j F_{ij}$$

$$B_i = E_i + \sum_{j=1}^{n} \left(\rho_i F_{ij}\right) B_j$$

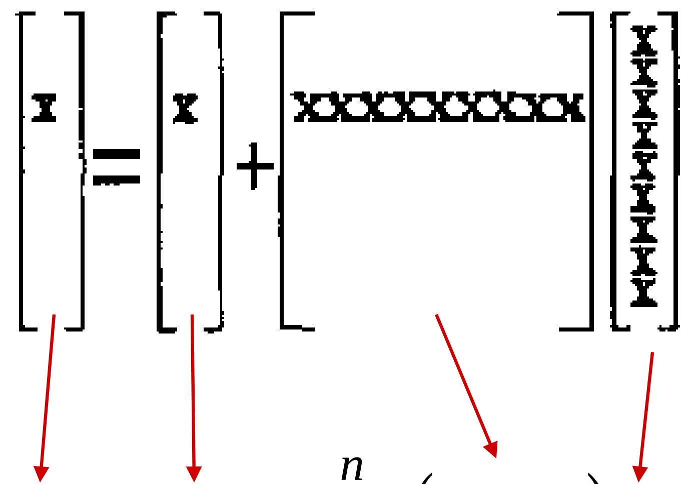
$$B_i \ due \ to \ B_j = \rho_i B_j F_{ij}$$

Gathering



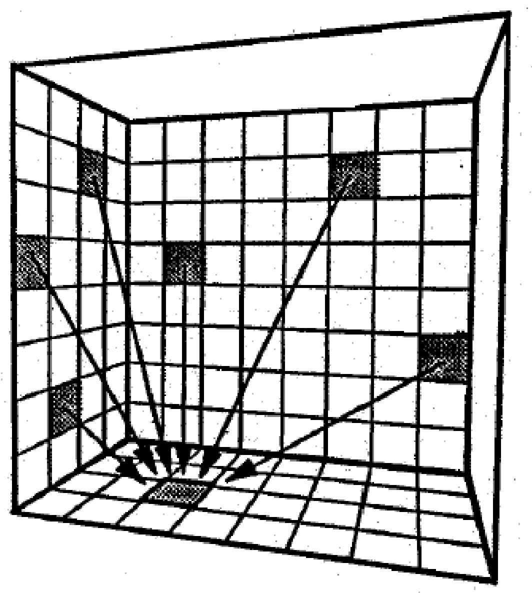
GATHERING

- Gathering light through a hemi-cube allows one patch radiosity to be updated.



$$B_i = E_i + \sum_{j=1}^{n} (\rho_i F_{ij}) B_j$$

Gathering



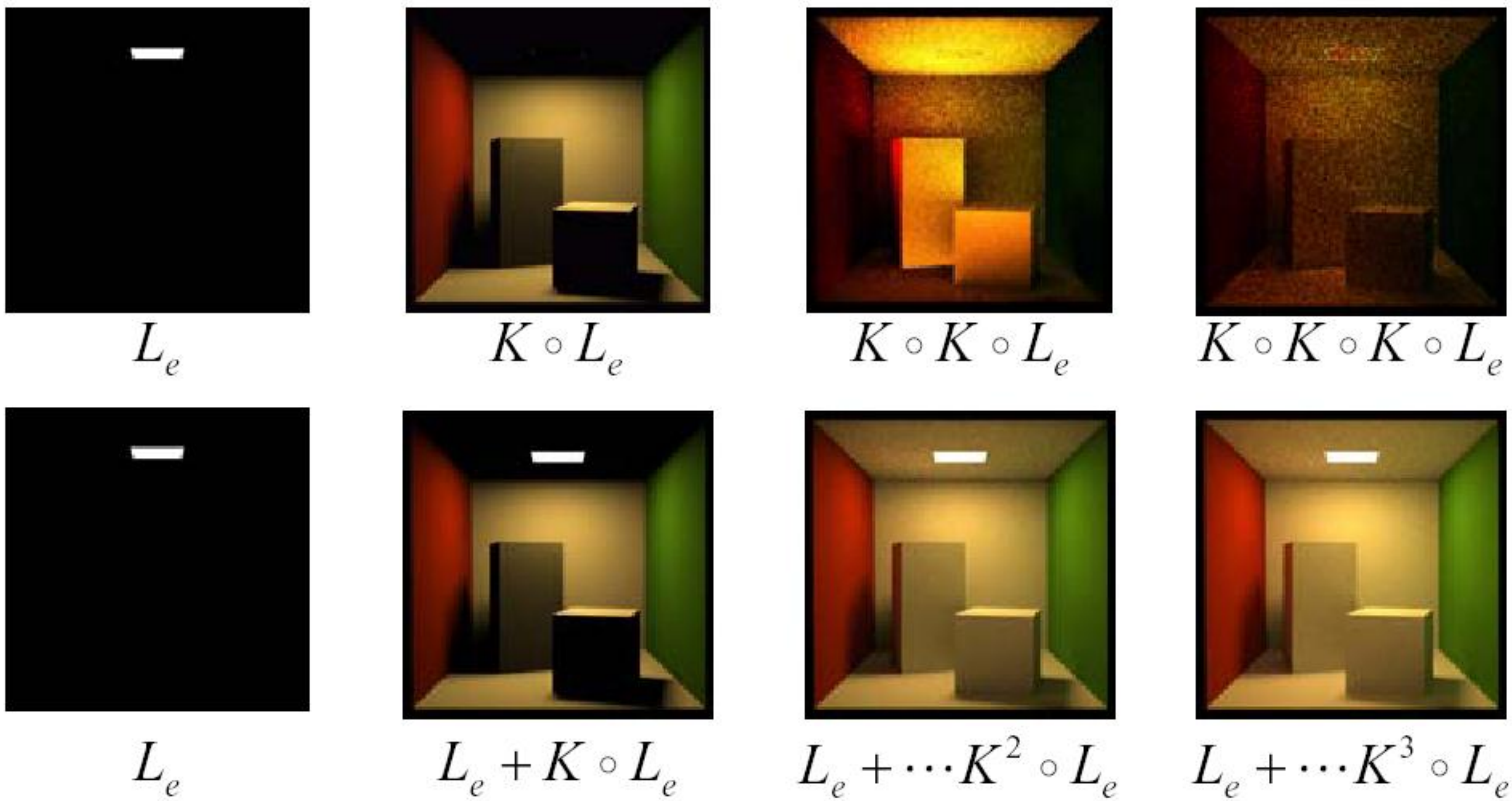
```
for(i=0; i<n; i++)
  B[i] = Be[i];

while( !converged ) {
  for(i=0; i<n; i++) {
    E[i] = 0;
    for(j=0; j<n; j++)
      E[i] += F[i][j]*B[j];
    B[i] = Be[i]+rho[i]*E[i];
  }
}
```

Row of F times B

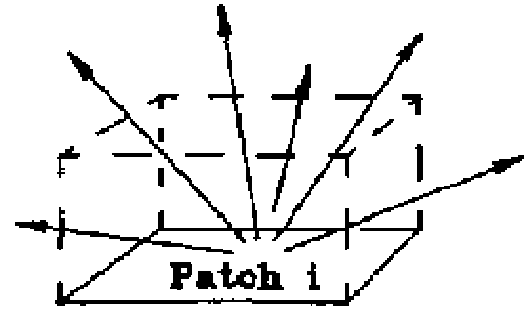
Calculate one row of F and discard

Successive Approximation

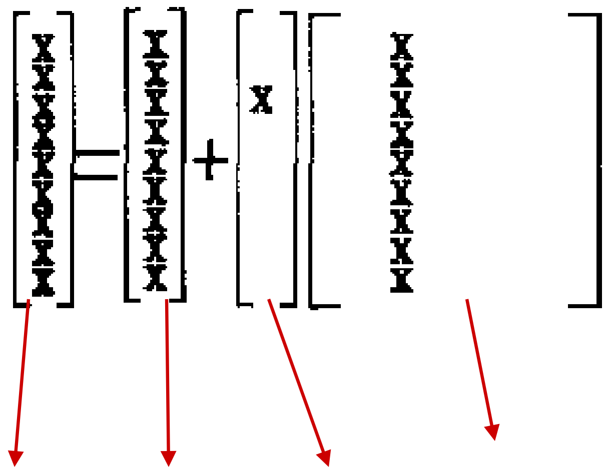


Shooting

- Shooting light through a single hemi-cube allows the whole environment's radiosity values to be updated simultaneously.



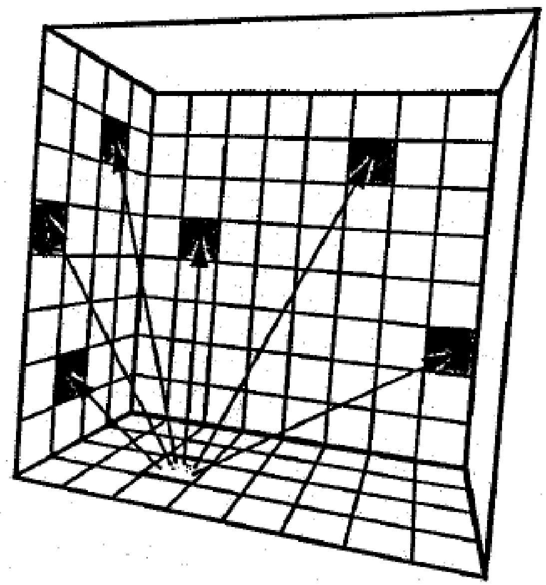
SHOOTING



For all j $\Rightarrow$ $B_j = B_j + B_i\left(\rho_j E_{ji}\right)$

where $F_{ji} = \dfrac{F_{ij} A_i}{A_j}$

Shooting

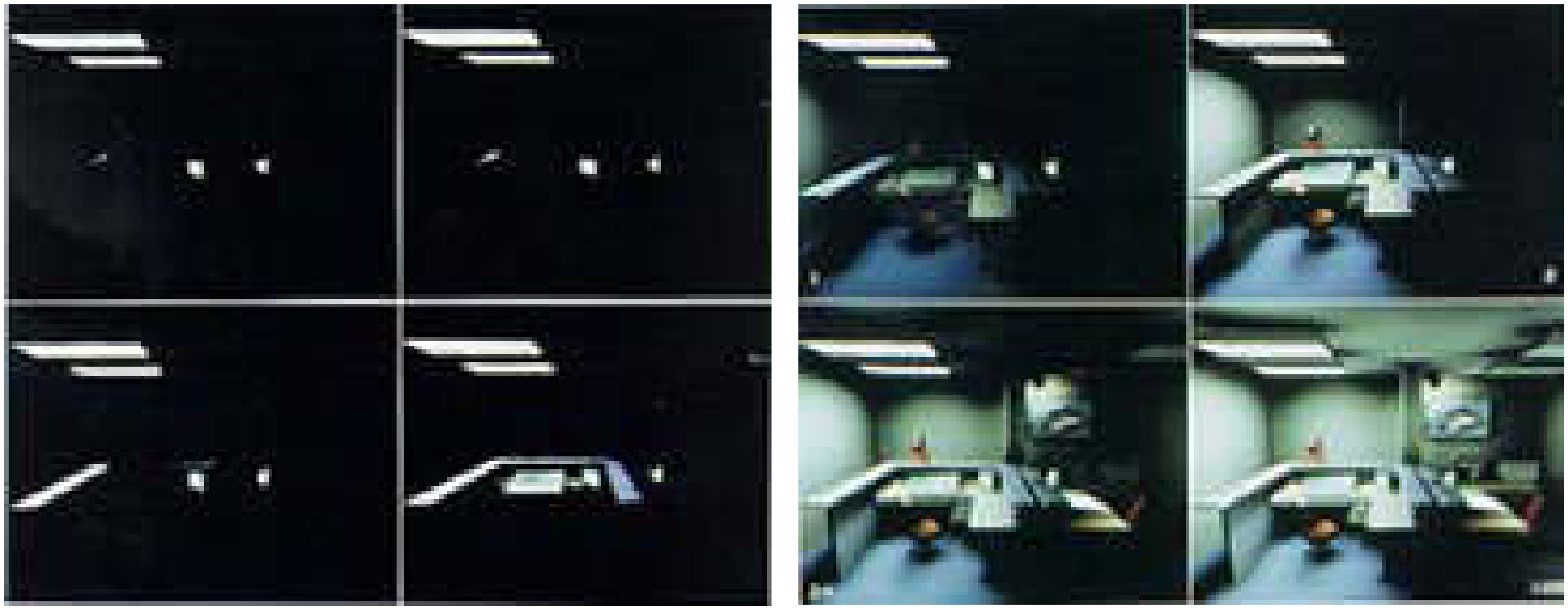


```
for(i=0; i<n; i++) {
  B[i] = dB[i] = Be[i];
  while( !converged ) {
    set i st dB[i] is the largest;
    for(j=0;j<n;j++)
      if(i!=j) {
        db =rho[j]*F[j][i]*dB[i];
        dB[j] += db;
        B[j] += db;
      }
    dB[i]=0;
  }
}
```

Brightness order

Column of F times B

Progressive Radiosity



(a) (b)

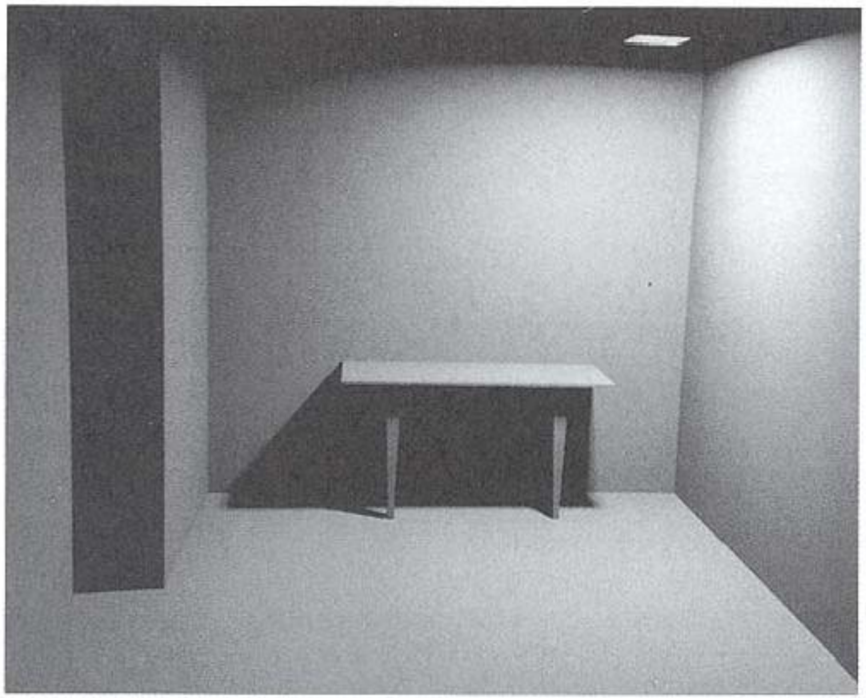
(a) Traditional Gauss-Seidel iteration of 1, 2, 24 and 100.
(b) Progressive Refinement (PR) iteration of 1, 2, 24 and 100.

From Cohen, Chen, Wallace, Greenberg 1988

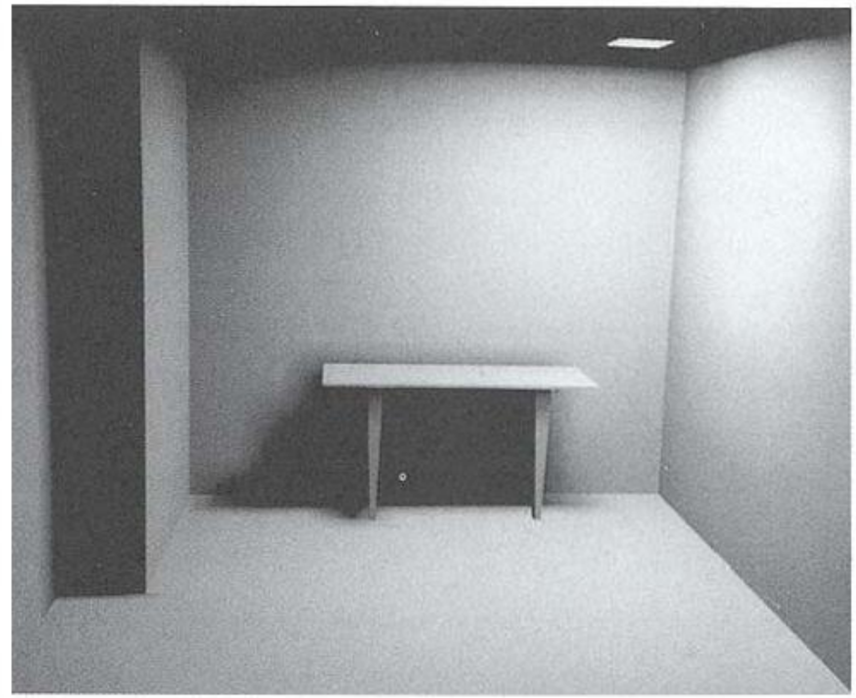
Next

- We have the form factors
- How do we find the radiosity solution for the scene?
  - The "Full Matrix" Radiosity Algorithm
  - Gathering & Shooting
  - Progressive Radiosity
- Meshing

Accuracy



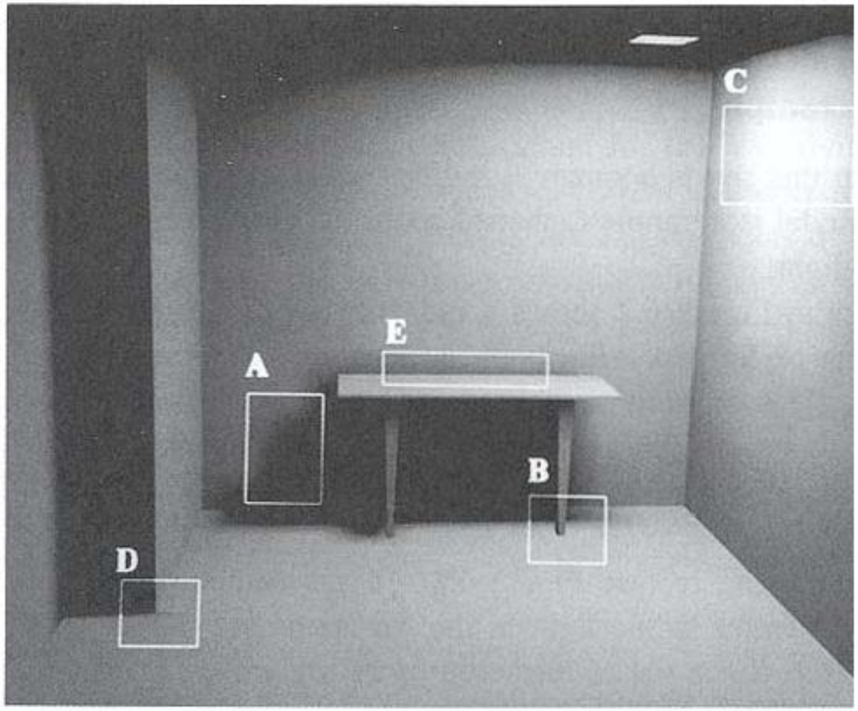
Reference Solution

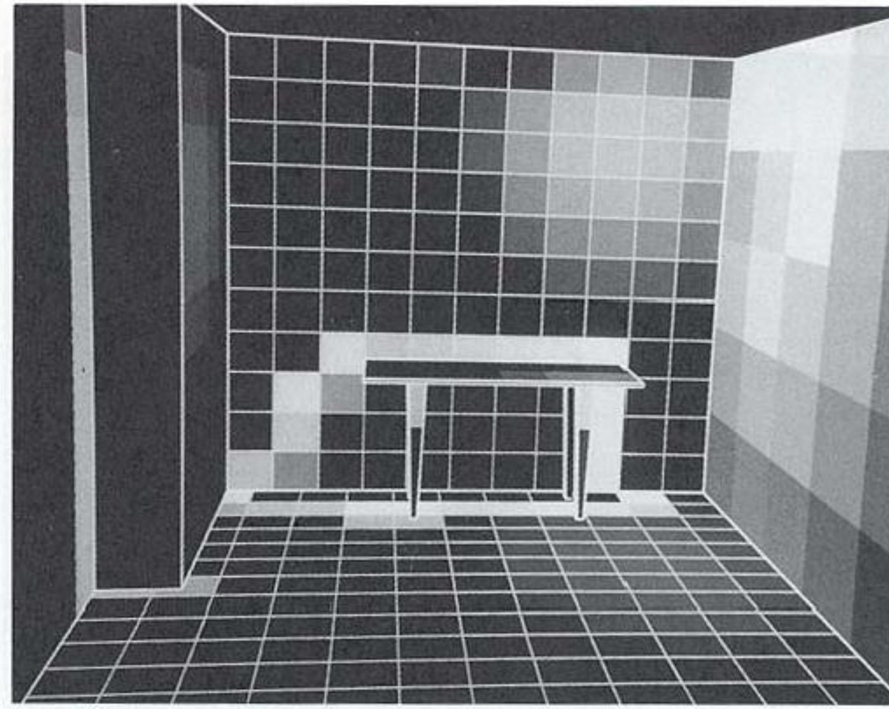


Uniform Mesh

Table in room sequence from Cohen and Wallace

Artifacts

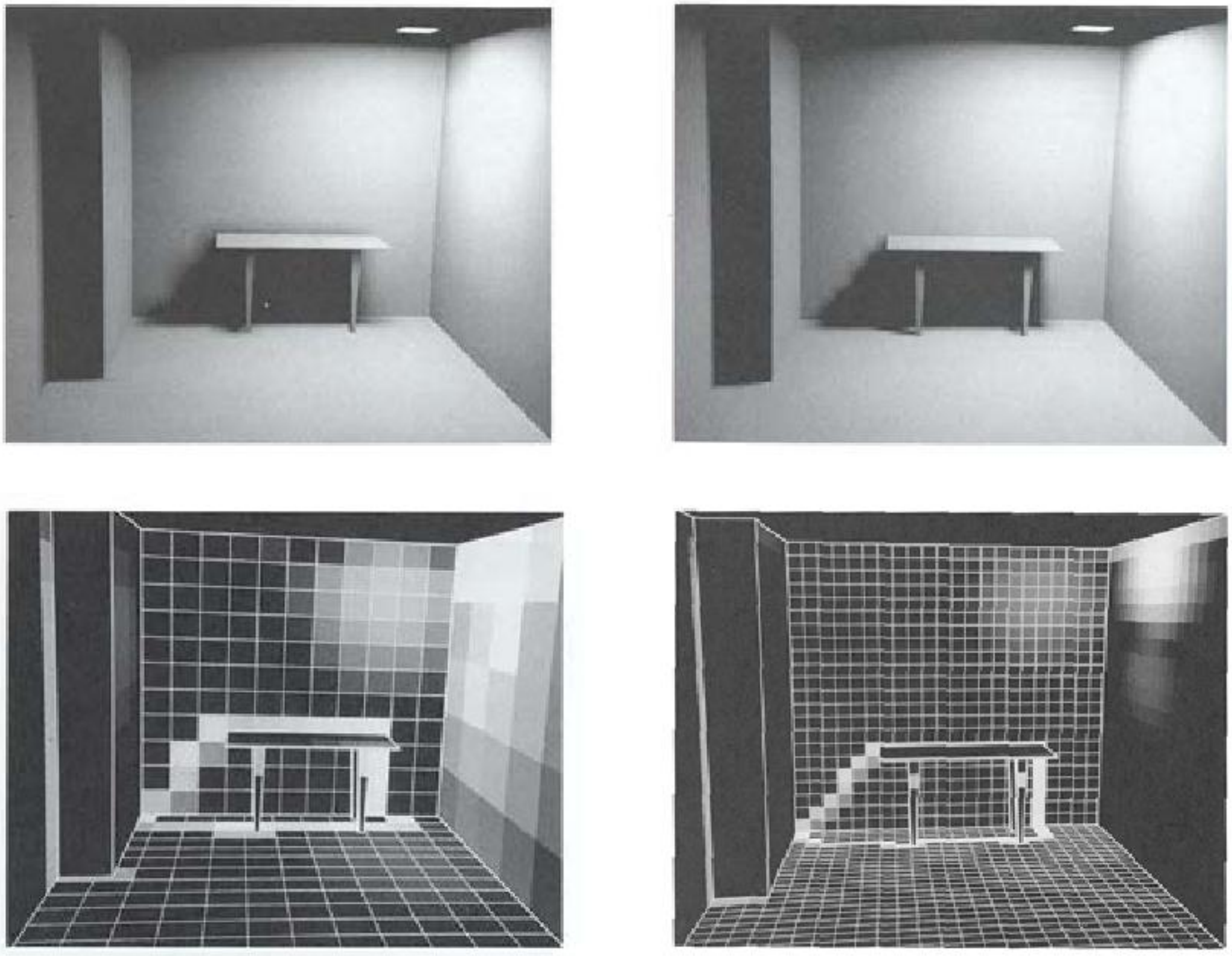




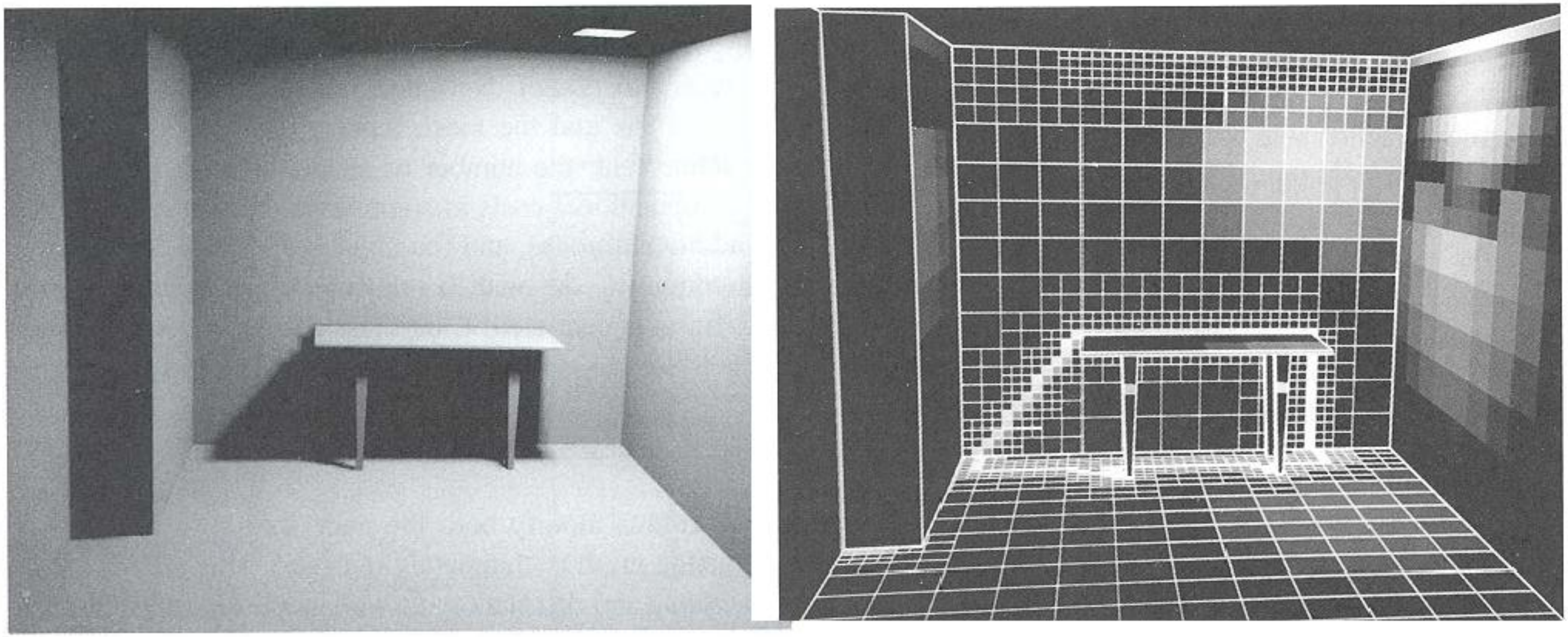
Error Image

A. Blocky shadows
B. Missing features
C. Mach bands
D. Inappropriate shading discontinuities
E. Unresolved discontinuities

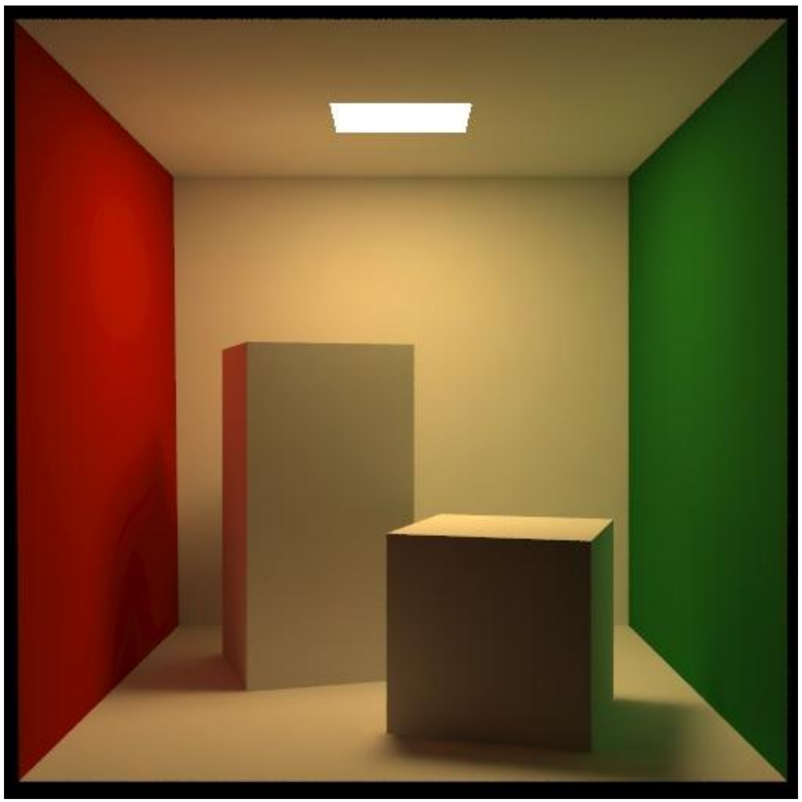
Increasing Resolution



Adaptive Meshing



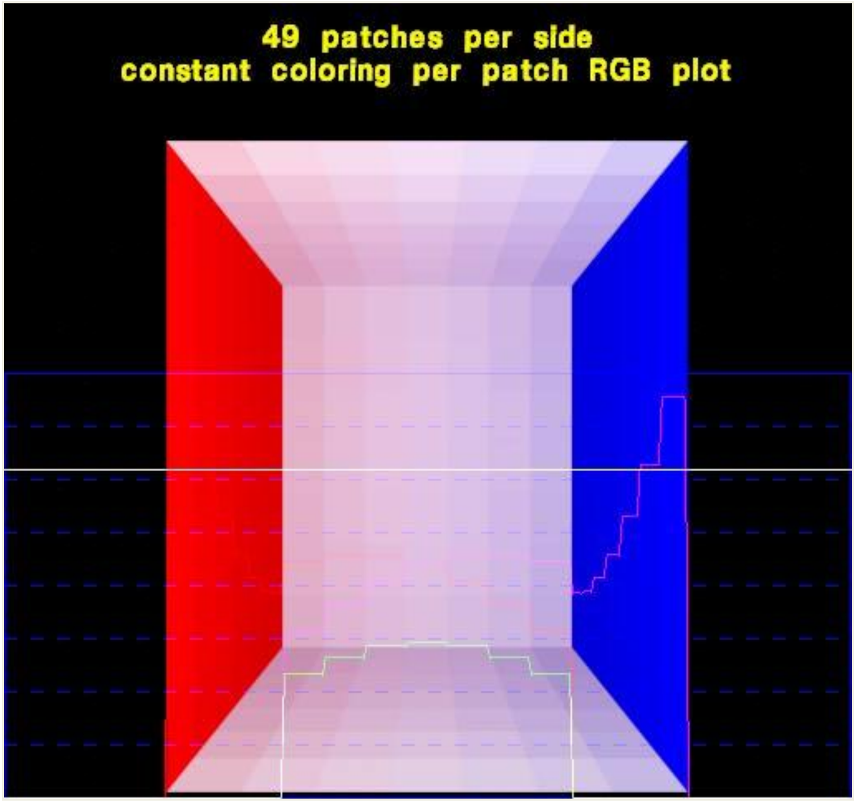
Some Radiosity Results



The Cornell Box

- This is the original Cornell box, as simulated by Cindy M. Goral, Kenneth E. Torrance, and Donald P. Greenberg for the 1984 paper *Modeling the interaction of Light Between Diffuse Surfaces*, Computer Graphics (SIGGRAPH '84 Proceedings), Vol. 18, No. 3, July 1984, pp. 213-222.

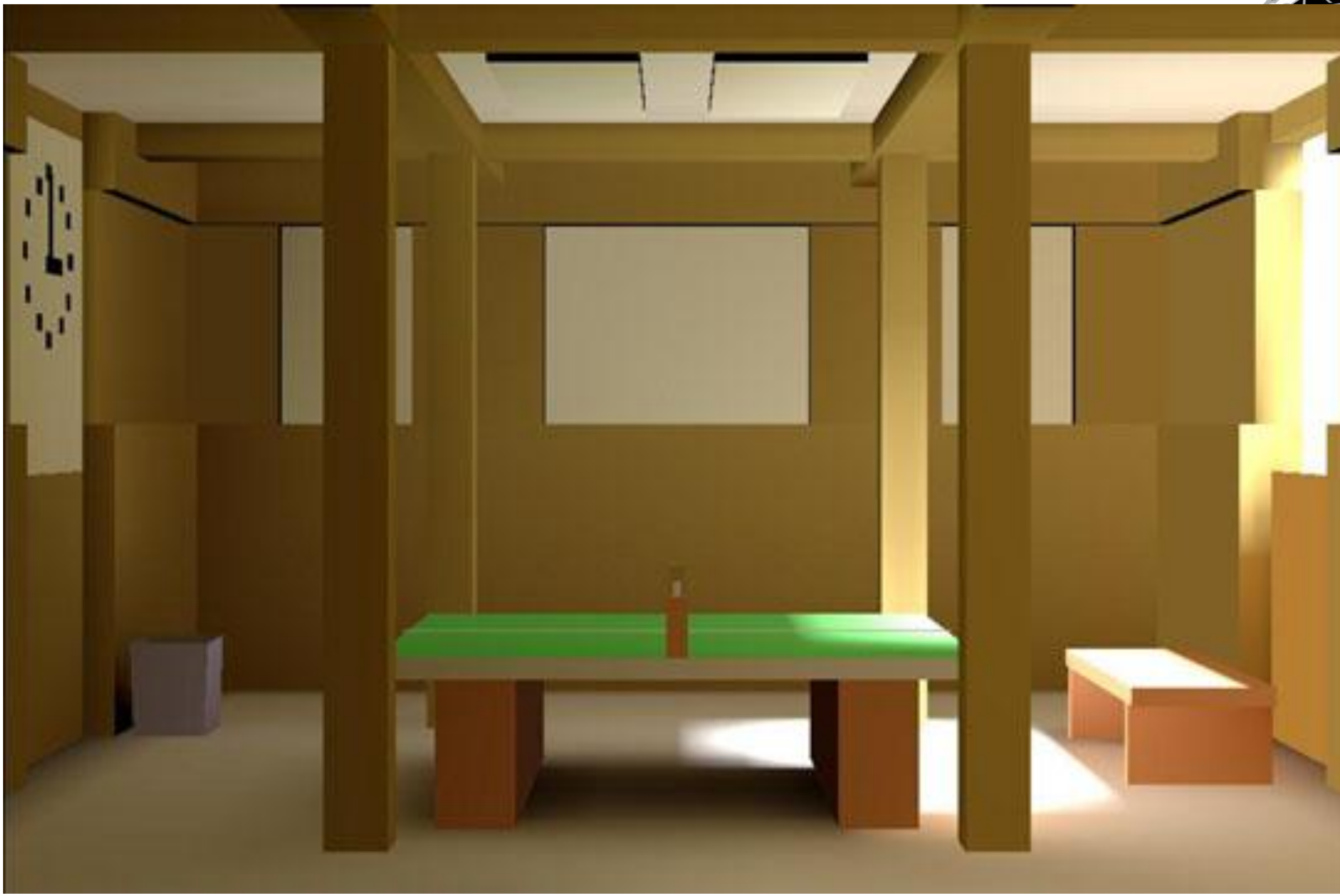
- Because form factors were computed analytically, no occluding objects were included inside the box.

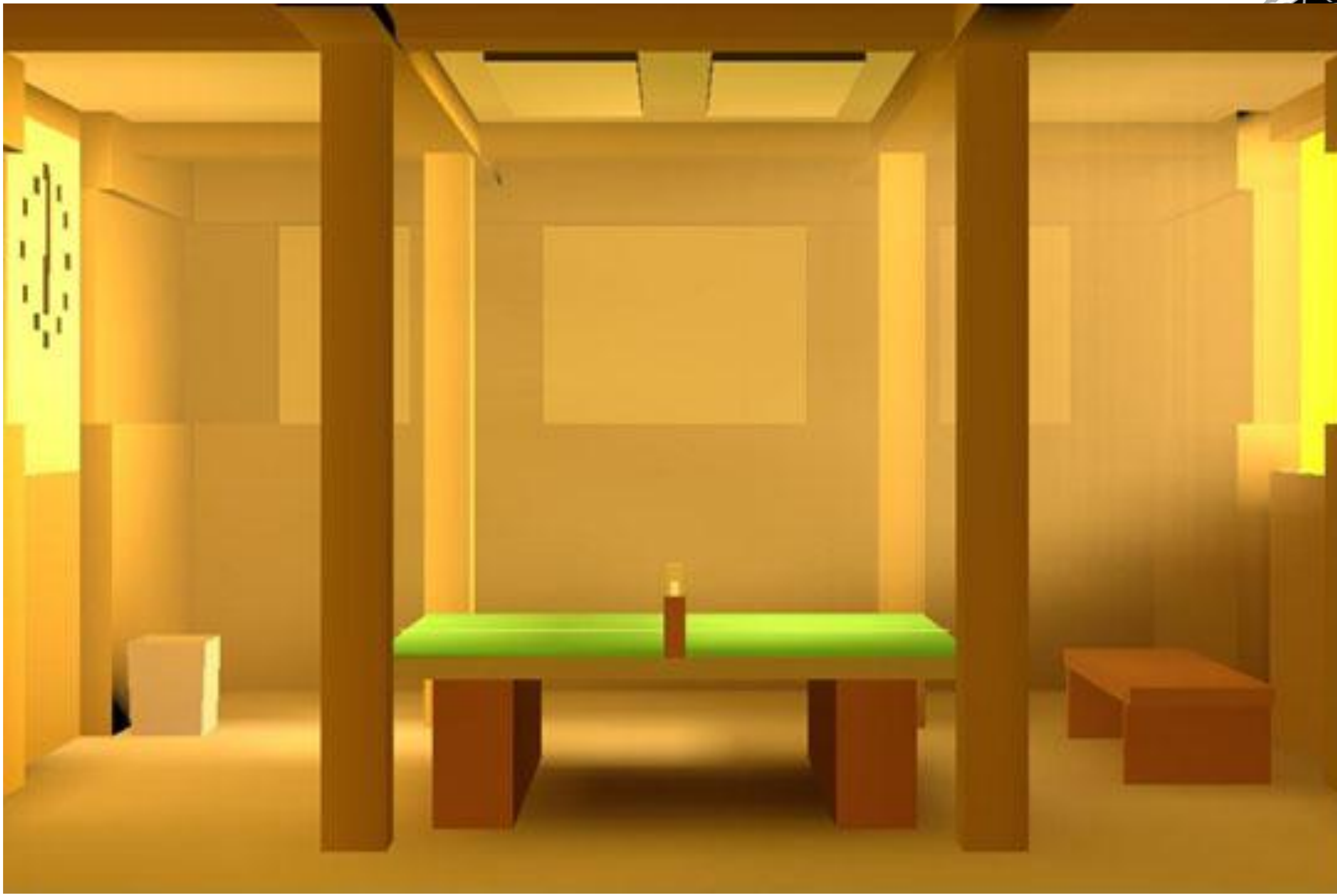


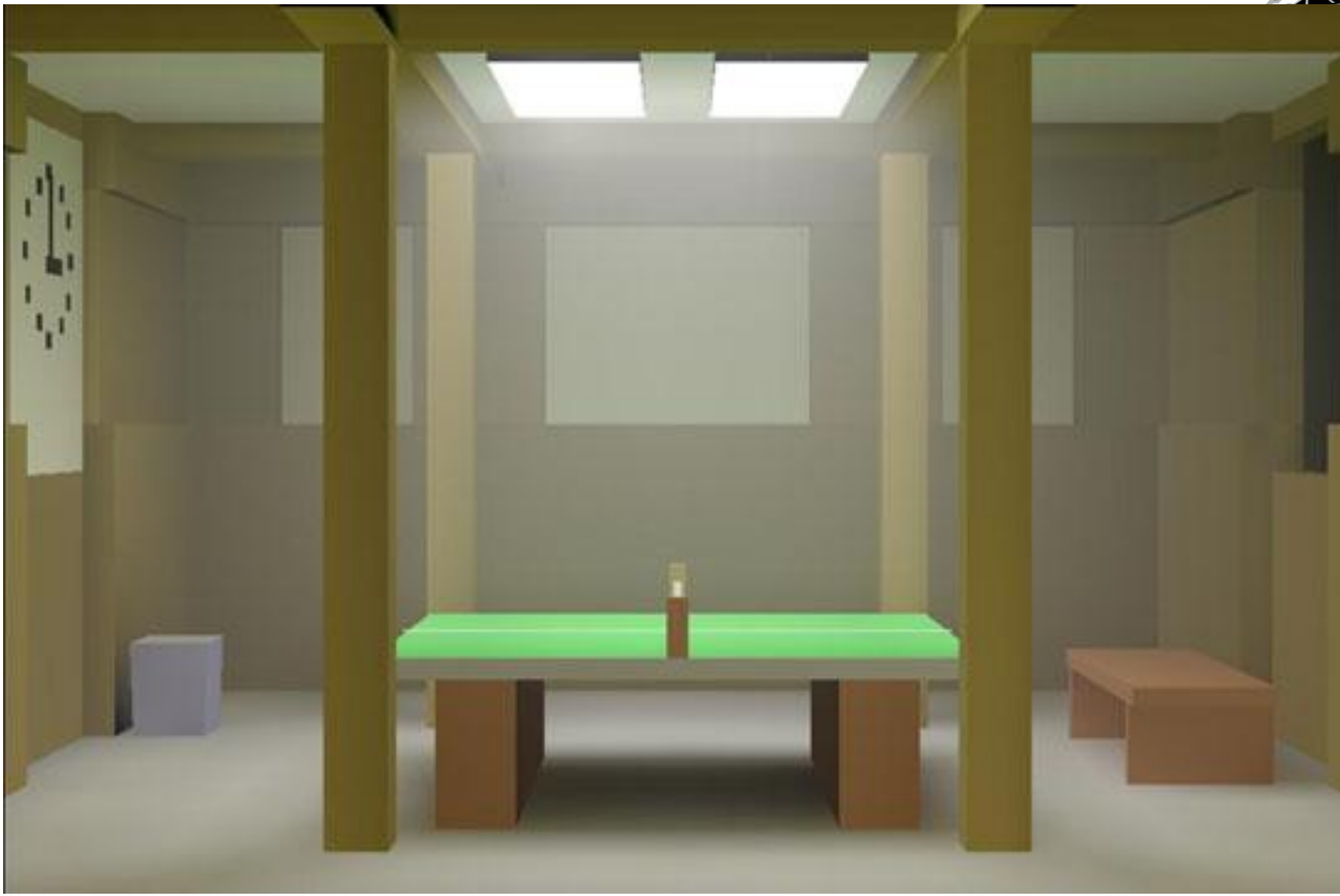
The Cornell Box

- This simulation of the Cornell box was done by Michael F. Cohen and Donald P. Greenberg for the 1985 paper *The Hemi-Cube, A Radiosity Solution for Complex Environments*, Vol. 19, No. 3, July 1985, pp. 31-40.

- The hemi-cube allowed form factors to be calculated using scan conversion algorithms (which were available in hardware), and made it possible to calculate shadows from occluding objects.







Discontinuity Meshing

Opera Lighting

- This scene from *La Boheme* demonstrates the use of focused lighting and angular projection of predistorted images for the background.

- It was rendered by Julie O'B. Dorsey, Francois X. Sillion, and Donald P. Greenberg for the 1991 paper *Design and Simulation of Opera Lighting and Projection Effects*.

Radiosity Factory

- These two images were rendered by Michael F. Cohen, Shenchang Eric Chen, John R. Wallace and Donald P. Greenberg for the 1988 paper *A Progressive Refinement Approach to Fast Radiosity Image Generation*.
- The factory model contains 30,000 patches, and was the most complex radiosity solution computed at that time.
- The radiosity solution took approximately 5 hours for 2,000 shots, and the image generation required 190 hours; each on a VAX8700.

Museum

- Most of the illumination that comes into this simulated museum arrives via the baffles on the ceiling.
- As the progressive radiosity solution executed, users could witness each of the baffles being illuminated from above, and then reflecting some of this light to the bottom of an adjacent baffle.
- A portion of this reflected light was eventually bounced down into the room.
- The image appeared on the proceedings cover of SIGGRAPH 1988.