Introduction to Radiosity

John F. Hughes

and

Andries van Dam

Brown University

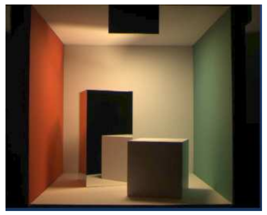
Rendering a Scene

- The *scene* consists of a geometric arrangement of surfaces
- It's illuminated by some *luminaires* (light sources)
- We *observe* it from some point and try to make a synthetic "photograph"
- This is, to put it bluntly, too hard
- *Raytracing* makes an approximation of how light gets from the luminaires into the camera
  - assumes knowledge of reflectance on surfaces
  - assumes that reflectance information can be condensed into a simple illumination equation
  - assumes (generally) that each ray of light bounces from the surface in only one direction, specularly
- Diffuse surfaces look bad
- Subtle reflection characteristics like *anisotropy* (property of reflecting light in a non-uniform way across a surface) are lost
- Requires ambient term in the illumination function as a gross hack to approximate diffuse global reflections
- Ignores/approximates a substantial amount of the light-energy transfer in the scene

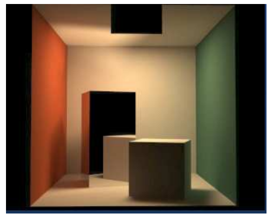
What Can We Do?

- Can design an alternative simulation of the real transfer of light energy in the scene
- With any luck, this will be more accurate
- Accuracy is relative
  - hall of mirrors is specular → raytracing
  - museum with latex-painted walls is diffuse → radiosity
- Neither radiosity nor raytracing solves all the problems
- Current best solutions use a hybrid technique: raytracers that take a final "diffuse illumination" pass, or radiosity solutions that add a "specular" pass
  - both are temporary hacks
- Radiosity approximates global diffuse inter-object reflection by tessellating the scene and considering how each pair of surface elements (patches) send and receive light energy, an $O(n^2)$ operation that will be best accomplished by an iterative solution (progressive refinement)

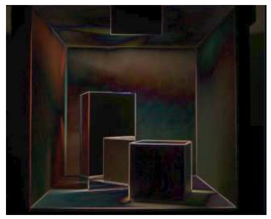
Pretty Pictures



Reality (or at least diffuse reality)…



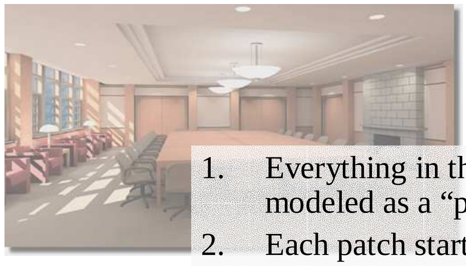
Minus Radiosity…



Equals "Not very much"

http://www.graphics.cornell.edu/online/box/compare.html

The Radiosity Technique: An Overview

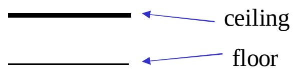


1. Everything in the scene is modeled as a "patch"
2. Each patch starts off with an initial luminance value (all but luminaries are probably zero)
3. We iteratively determine how much luminance travels from each patch to each other patch until the entire system converges to stable values

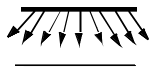
We can then render the scene *from any angle* without recomputing these final patch luminances

Overview of Radiosity

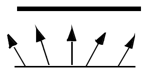
- The radiometric term *radiosity* means the rate at which energy leaves a surface, which is the sum of the rates at which the surface emits energy and reflects (or transmits) energy received from all other surfaces. Radiosity simulations are usually based on a thermal engineering model of emission and reflection of radiation using finite element approximations. They assume conservation of energy in closed environments. First determine all light interactions in a view-independent way, then render one or more views.
- Consider a room with only floor and ceiling:



- Suppose the ceiling is actually a fluorescent drop-panel ceiling which emits light…



- The floor gets some of this light and reflects it back



- The ceiling gets some of this reflected light and sends it back… you get the idea.

Some Important Symbols
(Pay Attention!)

- energy = light = radiosity (for our purposes)
- E_i: The **initial** amount of energy radiating from the *i*'th patch
- B_i: The **final** amount of energy radiating from the *i*'th patch
- $F_{j\text{-}i}$: The fraction of the energy emitted by the *j*'th patch that is gathered by the *i*'th patch (the relationship between the *i*'th and *j*'th patches is based on their distance and their angles to each other)
- Rho ρ_i: The fraction of the incoming energy to a patch that is then exported in the next iteration

(more radiosity pictures)

Let's Arrange Those Symbols (1/2)

$$B_i = E_i + \rho_i \sum_{i \le j \le n} F_{j-i} B_j \text{ ; rewrite as } E_i = B_i - \rho_i \sum_{1 \le j \le n} F_{j-i} B_j$$

- The amount of light/radiosity/energy a patch finally emits is the initial emission plus the emission due specifically to the other *n-1* patches in the scene emitting to the patch.
- Thus: $B_1 - \rho_1 (F_{1-1} B_1 + F_{2-1} B_2 + F_{3-1} B_3 + \ldots) = E_1$
- Rewrite as a vector product:

$$\begin{bmatrix}(1-\rho_1 F_{1-1}) & -\rho_1 F_{2-1} & -\rho_1 F_{3-1} & \ldots\end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ B_3 \\ \vdots \end{bmatrix} = E_1$$

- And the whole system:

$$\begin{bmatrix}(1-\rho_1 F_{1-1}) & -\rho_1 F_{2-1} & -\rho_1 F_{3-1} & \cdots \\ -\rho_2 F_{1-2} & (1-\rho_2 F_{2-2}) & -\rho_2 F_{3-2} & \cdots \\ -\rho_3 F_{1-3} & -\rho_3 F_{2-3} & (1-\rho_3 F_{3-3}) & \cdots \\ \vdots & \vdots & \vdots & \cdots \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ B_3 \\ \vdots \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ E_3 \\ \vdots \end{bmatrix}$$

EECS 487: Interactive Computer Graphics

Let's Arrange Those Symbols (2/2)

Decompose the first matrix as:

$$\begin{bmatrix} 1 & 0 & 0 & \cdots \\ 0 & 1 & 0 & \cdots \\ 0 & 0 & 1 & \cdots \\ \vdots & \vdots & \vdots & \cdots \end{bmatrix} - \begin{bmatrix} \rho_1 & 0 & 0 & \cdots \\ 0 & \rho_2 & 0 & \cdots \\ 0 & 0 & \rho_3 & \cdots \\ \vdots & \vdots & \vdots & \cdots \end{bmatrix} \begin{bmatrix} F_{1-1} & F_{2-1} & F_{3-1} & \cdots \\ F_{1-2} & F_{2-2} & F_{3-2} & \cdots \\ F_{1-3} & F_{2-3} & F_{3-3} & \cdots \\ \vdots & \vdots & \vdots & \cdots \end{bmatrix}$$

Can be rewritten:

- $(I - D(\rho)F)B = E$
  - where $D(\rho)$ is a diagonal matrix with ρ_i as its *i*th diagonal entry.
- If we know E, ρ, and F, we can determine B
- If we let

$$A = I - D(\rho)F$$

- Then we are solving (for B) the equation

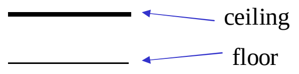
$$AB = E$$

- This is a *linear system*, and methods for solving these are well-known, e.g. Gaussian elimination or Gauss-Seidel iteration (although which method is best depends on the nature of the matrix A)
- Typically want B, knowing E and A

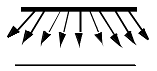
EECS 487: Interactive Computer Graphics

An[other] Intro to Radiosity(1/3)

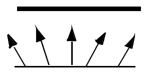
- The radiometric term *radiosity* means the rate at which energy leaves a surface, which is the sum of the rates at which the surface emits energy and reflects (or transmits) energy received from all other surfaces. Radiosity simulations are usually based on a thermal engineering model of emission and reflection of radiation using finite element approximations. They assume conservation of energy in closed environments. First determine all light interactions in a view-independent way, then render one or more views.
- Consider a room with only floor and ceiling:



- Suppose the ceiling is actually a fluorescent drop-panel ceiling which emits light…



- The floor gets some of this light and reflects it back



- The ceiling gets some of this reflected light and sends it back… you get the idea.

An Introduction to Radiosity(2/3)

- Both ceiling and floor are acting as area light sources emitting and reflecting light uniformly over their areas (all surfaces are considered such in radiosity)
- Let the ceiling emit 12 units of light per second
- Let the floor reflect 50% of what it gets; let it get 1/3 of the light from the ceiling (based on geometry)
- And let the ceiling get 1/3 of the floor's light (based on geometry), and reflect 75% of what it gets
- Writing B_1 for the ceiling's total light, and B_2 for the floor's, and E_1 and E_2 for the light generated internally by each, we have:

  ceiling:

  $$E_1 = 12, B_1 = E_1 + \rho_1 (F_{2-1} B_2) = 12 + \frac{3}{4} \cdot \frac{1}{3} B_2$$

  floor:

  $$E_2 = 0, B_2 = E_2 + \rho_2 (F_{1-2} B_1) = 0 + \frac{1}{2} \cdot \frac{1}{3} B_1$$

- First, we'll solve this simple case, then write equations for the more general case

An Introduction to Radiosity(3/3)

- Method 1, gathering energy: send out light from emitters everywhere, accumulate it, resend from all patches… Each iteration uses the radiosity values from the previous iteration as estimates for the recursive form. Iterate by rows.
  $B^k = E + D(\rho)FB^{k-1}$;

  $$B^1 = E$$
  $$B^2 = E + D(\rho)FB^1$$
  $$B^3 = E + D(\rho)FB^2$$

  Where B^k is your kth guess at the radiosity values B_1, B_2…

  Results for our example:

  $\{12, 0\} = \{B_1, B_2\} = \{E_1, E_2\}$

  $\{12, 2\} = \{E_1 + \rho_1 F_{2-1} B_2, E_2 + \rho_2 F_{1-2} B_1\} = \left\{12 + 0, 0 + \frac{1}{2} \cdot \frac{1}{3} \cdot 12\right\}$

  $\{12.5, 2\} = \left\{12 + \frac{3}{4} \cdot \frac{1}{3} \cdot 2, 0 + \frac{1}{2} \cdot \frac{1}{3} \cdot 12\right\}$

  $\{12.5, 2.08373\} = \left\{12 + \frac{3}{4} \cdot \frac{1}{3} \cdot 2, 0 + \frac{1}{2} \cdot \frac{1}{3} \cdot 12.5\right\}$

  {12.5208, 2.08373}

  {12.5208, 2.08681}

  {12.5217, 2.08681}

  {12.5217, 2.08695}

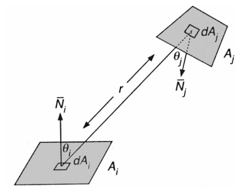
- Like "hitting the cosine button" repeatedly on calculator to solve cos(x) = x; a form of progressive refinement.

General Radiosity Equation(1/3)

- The radiosity equation for **normalized unit areas** of Lambertian diffuse patches is:

$$B_i = E_i + \rho_i \sum_{1\le j\le n} \frac{(B_j A_j)\cdot F_{j-i}}{A_i}$$

- A_i is the area of the i'th patch
- B_i is total radiosity in watts/m^2 (i.e. energy/unit-time / unit-area) radiating from patch i
  - Note that we are now calculating B_i (and E_i) **per unit area**
- E_i is light emitted in watts/m^2
- ρ_i is fraction of incident energy reflected by patch i (related to diffuse reflection coefficient k_d in simple lighting model)
- ($B_j A_j$) is total energy radiated by patch j with area A_j (i.e., radiosity *x* area)



General Radiosity Equation(2/3)

- From the previous slide:

$$B_i = E_i + \rho_i \sum_{1\le j\le n} \frac{(B_j A_j)\cdot F_{j-i}}{A_i}$$

- F_{j-i} is fraction of energy leaving ("exported by") patch j arriving at patch i. It is the dimensionless *form factor* that takes into account shape and relative orientation of each patch and occlusion by other patches. It is a function of (r, θ_i, and θ_j).
  - Geometrically, F_{j-i} is the relative area of receiver patch i subtends in sender patch j's "view", a hemisphere centered over patch j
  - Note: generally patches may be concave and have self-reflection, where $F_{i-i} \neq 0$
- $\sum_{i=1}^{n} F_{j-i} = 1$ for all i (conservation of energy)
- $(B_j A_j)\cdot F_{j-i}$ is total amount of energy leaving patch j arriving at patch i
- $(B_j A_j)\cdot F_{j-i} / A_i$ is total amount of energy leaving patch j arriving at *unit area* of patch i

General Radiosity Equation(3/3)

- *Reciprocity relationship* between $F_{i\text{-}j}$ and $F_{j\text{-}i}$, proven later:

$$\frac{F_{i-j}}{A_j} = \frac{F_{j-i}}{A_i} \quad \text{or} \quad A_i \cdot F_{i-j} = F_{j-i} \cdot A_j$$

  – which means form factors scaled for unit area of receiver patch are equal

$$\frac{A_i}{A_j} = \frac{F_{j-i}}{F_{i-j}}$$

$$F_{j-i} = \frac{A_i}{A_j} F_{i-j}$$

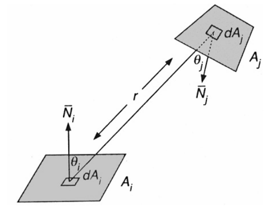
$$B_i = E_i + \rho_i \sum_{1 \le j \le n} \frac{(B_j A_j) \cdot F_{j-i}}{A_i}$$

$$B_i = E_i + \rho_i \sum_{1 \le j \le n} (B_j A_j) \cdot \left(\frac{A_i}{A_j} F_{i-j} \right) \Big/ A_i$$

- Therefore, $B_i = E_i + \rho_i \sum_{1 \le j \le n} B_j F_{i-j}$
  – which is easier to deal with, if less intuitive
  – in terms of the matrix of form factors, *F*, this says that the radiosity of receiver patch *i* is the energy emitted by that patch plus the attenuated sum of each sender *j*'s radiosity times the form factor from *i* to *j;* in other words, for each receiver row *i* iterate across all the sender columns *j* to gather the energy.

- Note: we should calculate this for all wavelengths --- approximate with B_{iR}, B_{iG}, B_{iB}

Computing Form Factors(1/7)

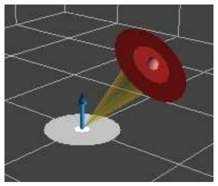


- Form factor from differential sending area dA_i to differential receiving area dA_j is:

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

  – for ray of length r between patches, at angles θ_i, θ_j to the normals of the areas. H_{ij} is 1 if dA_j is visible from dA_i and 0 otherwise.

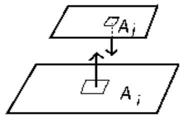
- We will motivate this equation…
- Also see form factor applet:



Applet: http://www.cs.brown.edu/exploratories/freeSoftware/catalogs/lighting_and_shading.html

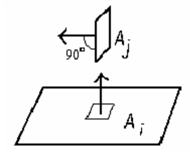
Computing Form Factors(2/7)

- When the two patches directly face each other, maximum energy is transmitted from A_i to A_j



- their normal vectors are parallel, $\cos\theta_j = 1$, $\cos\theta_i = 1$ since $\theta_i = \theta_j = 0°$

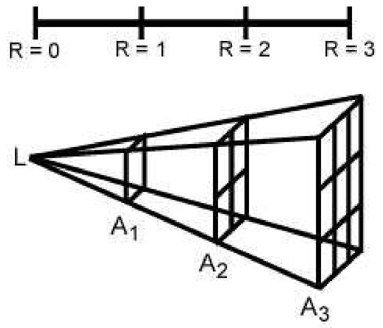
- Rotate A_j so that it is perpendicular to A_i. Now $\cos\theta_i$ is still 1, but $\cos\theta_j = 0$ since $\theta_j = 90°$



- In between the two extrema, we calculate the energy fraction by multiplying by $\cos\theta_j$. Tilting A_i means multiplying by $\cos\theta_i$
- Same as Lambertian diffuse reflection

Computing Form Factors(3/7)

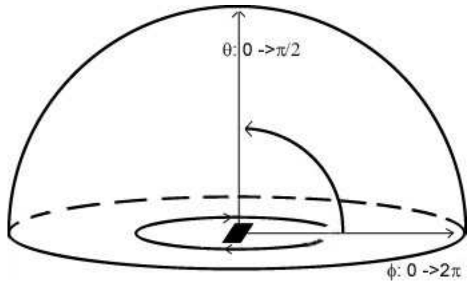
- From where does the r^2 term arise? The inverse-square law of light propagation:



- Consider a patch A_1, at a distance $R = 1$ from light source L. If P photons hit area A_1, their density is P/A_1. These same P photons pass through A_2. Since A_2 is twice as far from L, by similar triangles, it has four times the area of A_1. Therefore each similar patch on A_2 receives 1/4 of the photons

Computing Form Factors(4/7)

- The π in the formula is a normalizing factor



- If we integrate the form factor across the surface of a unit hemisphere, we need to achieve unity (all the light goes somewhere). By what constant k do we scale the integration to normalize this value? $r = 1$, $\theta_j = 0$

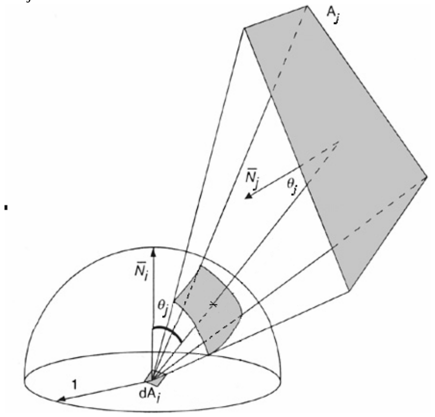
$$\iint_A k\cos\theta_i dA = 1$$

$$k = \frac{1}{\iint_A \cos\theta_i dA} = \int_0^{2\pi}\int_0^{\pi/2} \cos\theta_i[\sin\theta_i d\theta_i d\phi]$$

$$k = \frac{1}{\pi}$$

Computing Form Factors(5/7)

- Now consider a differential patch dA_i radiating to finite patch A_j



- $F_{di\text{-}j}$ can be computed by projecting those parts of A_j visible from dA_i onto the unit hemisphere centered about dA_i. The form factor is effectively the ratio of curved patch area to the total surface area of the hemisphere.
- Total surface area encompasses all energy emitted by dA_i

Computing Form Factors(6/7)

- This is an approximation! It only holds if dA_j is far from dA_i, so the angles θ_i and θ_j do not vary significantly across their respective patches
- To determine $F_{di\text{-}j}$, the form factor from differential area dA_i to finite area A_j, we integrate over area of patch j:

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

- H_{ij} again dictates visibility: $H_{ij} = 0$ implies occlusion
  - not trivial to resolve analytically for finite areas

Computing Form Factors(7/7)

- Let's complete the integration for taking dA_i to A_i to determine $F_{i\text{-}j}$
- Take area average over patch i to give form factor from A_i to A_j:

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

- If center point on patch is typical of all points, can approximate $F_{i\text{-}j}$ by $F_{di\text{-}j}$ for a dA_i, at patch i's center. Remember, both are percentages
- Again this breaks if patches are in close proximity, causing large variations among θ_i and θ_j
- An aside: we are now in a position to prove the reciprocity relationship. Cross multiplying in the equation for the form factor above gives us:

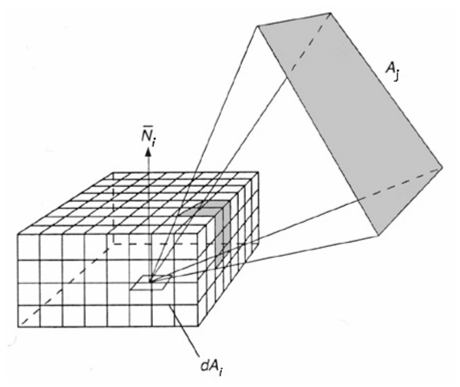
$$F_{i-j} A_i = \int_{A_i} \int_{A_j} \frac{\cos\theta_i \cos\theta_j}{\pi \cdot r^2} H_{ij}\, dA_j\, dA_i$$

$$F_{j-i} A_j = \int_{A_j} \int_{A_i} \frac{\cos\theta_i \cos\theta_j}{\pi \cdot r^2} H_{ji}\, dA_i\, dA_j$$

  - the double integrals are equal since it doesn't matter which is the inner and which is the outer integral
- Using transitivity gives us the reciprocity relationship:

$$F_{i-j} A_i = F_{j-i} A_j$$

Approximating Form Factors(1/2)



- Rather than projecting A_j onto a hemisphere, Cohen and Greenberg proposed projecting onto the upper half of a cube centered about dA_i, with its top face parallel to the surface. Each face of the *hemicube* is divided into equal sized square cells. All patches A_j are clipped against view volume frusta defined by the center of the cube and each of its upper five faces. Think of each face of the cube as a film plane which records what a patch, dA_i , "sees" in each of the five directions. In other words, think of the film as pixels and scan convert the clipped, projected polygonal patch, including z buffering onto each face.

Approximating Form Factors(2/2)

- Identity of closest intersecting patch is recorded at each cell (the survivor of the z buffer algorithm). Each hemicube cell *p* is associated with a precomputed delta form factor value,

$$\Delta F_p = \frac{\cos\theta_i \cos\theta_p}{\pi \cdot r^2} \Delta A$$

where θ_p is the angle between *p*'s surface normal and vector of length r between dA_i and *p*, and where ΔA is area of cell

- Can approximate $F_{di\text{-}j}$ for any patch *j* by summing ΔF_p associated with each cell *p* in A_j's hemicube projection
- Provides visibility determination through z buffer (albeit approximate)

Faster Progressive Refinement: Shooting

- Method 2, dual to gathering: Instead of for each receiver *i* gathering energy sequentially from all senders *j*, shoot it in order from the brightest to the least bright patch (i.e., starting with the most significant light sources first)
  - accumulate at the receivers
  - iteratively shoot from patch that has the largest amount of "unshot" radiosity (e.g., for a single light source, the patch which has the largest form factor with that source will be the next patch to shoot)
- As shown next, computing the form factor can be done by using a single hemicube for each shooter that can be computed and discarded for each receiver
  - solves the $O(n^2)$ processing and storage problem for each iteration of gathering
  - shooting converges faster than gathering
- Note that in gathering, must process consecutive rows and all rows must be processed for each iteration
  - all form factors must be calculated before the first iteration of Gauss-Seidel occurs
- In shooting, iterate by column in order of patch with the most "unshot" radiosity
  - can form an estimate even with only first column shot
  - can add decreasing ambient term (goes to 0) as a hack

Details on Shooting(1/2)

- Each row of matrix used in "gathering" *(I – D(ρ)F)* represents estimate of patch *i*'s radiosity B_i based on estimates of other patch radiosities
  - each term in summation gathers light from patch *j* for all *j*:

$$B_i^t\left(\text{due to } B_j^{t-1}\right) = \rho_i\, B_j^{t-1} F_{i-j}, \text{ for all } j$$

  - therefore,

$$B_i^t = \rho_i \sum_{j=1}^{n} B_j^{t-1} F_{i-j}$$

- For shooting, shoot from patch *i* to each patch *j* in turn; again

$$B_j^t\left(\text{due to } B_i^{t-1}\right) = \rho_j\, B_i^{t-1} F_{j-i}, \text{ for all } j$$

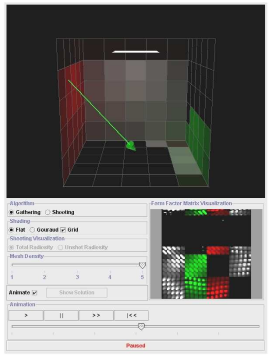
  - for each receiver *j*, keep adding radiosity from successive sources *i* in order of decreasing radiosity
- So given an estimate of B_i we can estimate its impact on all receiving patches *j*, at the cost of computing $F_{j\text{-}i}$ for each receiver patch *j*, i.e., via *n* hemicubes. But that is still too much work.

Details on Shooting(2/2)

- Using reciprocity again,

$$B_j^t\left(\text{due to } B_i^{t-1}\right) = \rho_j B_i^{t-1} F_{i-j}\,\frac{A_i}{A_j}$$

  – which requires only the hemicube over patch *i*! Thus only a single hemicube and its *n* form factors need be computed each pass!
- Note that for a given shooter *i*, we loop through all receiving patches *j*. Given our notation for the form factor matrix, holding *i* constant and looping through all *j* corresponds to traversing a column.
- See shooting vs. gathering applet:


Algorithm: [✓ Gathering] Shooting
Shading: [✓ Flat] Gouraud [✓ Grid]
Shooting Visualization: [✓ Total Radiosity] Unshot Radiosity
Mesh Density: 5
Animate [✓]
Paused

Applet: http://www.cs.brown.edu/exploratories/freeSoftware/catalogs/lighting_and_shading.html

Radiosity Pseudocode

- Algorithm for fast progressive refinement through shooting:

```
$U^0$ = e;
$B^0$ = e;
t = 1;

do {
   i = index_of(MAX($u^{t-1}$));
   precalculate $hemicube_i$;

   for (j = 1; j < n; ++) {
        $b_j^t = u_i^{t-1} F_{i-j} A_i/A_j + b_j^{t-1}$;
        $u_j^t = u_i^{t-1} F_{i-j} A_i/A_j + u_j^{t-1}$;
   }
   $u_i^t = u_i^{t-1} F_{j-j}$;
   ++t;

} while $B^t - B^{t-1}$ > tolerance;
```

Benefits of Radiosity

- Color bleeding: a red wall next to a white one casts a reddish glow on the white wall near the corner.
- Soft shadows – an “area” light source casts a soft shadow from a polygon.
- No ambient term hack, so when you want to look at your object in low light, you don’t have to adjust parameters of the objects – just the intensities of the lights!
- View independent: it assigns a brightness to every surface and you can just draw those suckers! (using a standard VSD algorithm and, say, Gouraud shading to obviate the faceted look – derive vertex radiosities by averaging patch radiosities.
- Used in other areas of engineering where energy radiation is computed.

Limitations of Radiosity

- Assumption that radiation is uniform in all directions
- Assumption that radiosity is piecewise constant
  - usual renderings make this assumption, but then interpolate cheaply to fake a nice-looking answer
  - this introduces quantifiable errors
- Computation of the form factors $F_{i\text{-}j}$ can be tough
  - especially with intervening surfaces, etc.
- Assumption that reflectivity is independent of directions to source and destination
- Assumption that intermediate medium is non-participatory (although there are additional equations and algorithms for calculating surface-to-volume form factors which can then be used in volume rendering a scene)
- Assumption that no surface is transparent or translucent
- Independence from wavelength – no fluorescence or phosphorescence
- Independence from phase – no diffraction
- *Enormity of matrices!* For large scenes, 10K x 10K matrices are not uncommon (shooting reduces need to have it all memory resident)

More Comments

- Even with these limitations, it produces lovely pictures
- For *n* surface patches, we have to build an *n x n* matrix and solve *Ax = b*, which takes $O(n^2)$, this gets rather expensive for large scenes
- Could we do it in *O(n)* instead?
- The answer, for lots of nice scenes, is "Yes"

Tangent: This is a lot like Google

- The Google search engine uses an system much like radiosity to rank its pages
  - Site rankings are determined not only by the number of links from various sources, but by the number of links coming into those sources (and so on)
  - After multiple iterations through the link network, site rankings stabilize
  - Site importance is like luminance, and every site is initially considered an "emitter"

Making Radiosity Fast

- One approach is *importance driven* radiosity: if I turn on a bright light in the graphics lab with the door open, it'll lighten my office a little…
- …but not much
- By taking each light source and asking "what's illuminated by this, really?" we can follow a "shooting" strategy in which unshot radiosity is weighted by its importance, i.e., how likely it is to affect the scene from *my point of view*
- No longer a view-independent solution…but much faster

Combining Radiosity with Raytracing

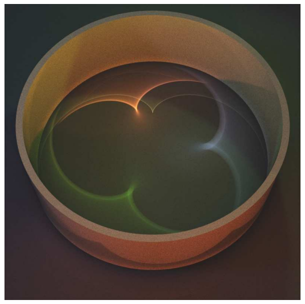
- Radiosity is perfect for diffuse reflection; rotten for specular
- Raytracing is perfect for specular; rotten for diffuse

So we compromise:

- Perform radiosity, and then raytrace your scene afterwards, treating every point in the scene as a "light source" emitting its radiosity; blend in the results
- Raytrace first, and then take all specularly reflected lights as "new lights" and do a final radiosity pass over the world and blend results
- Of course, we left out a few details, but you should be able to implement the combined algorithm from this slide

What is MLT? (1/4)

- Metropolis Light Transport is a type of Monte Carlo algorithm
- Monte Carlo describes a group of rendering algorithms which randomly sample every path which a light ray can take in the world
  - based on ray casting
  - classical ray tracing is a very simple, specialized type of Monte Carlo algorithm
  - point sampling of the complete rendering equation
  - in general, capable of much more complicated effects than radiosity and classical ray tracing combined



What is MLT? (2/4)

- Kajiya's Rendering Equation as reformulated over the set of all points in the scene (M):

$$L(x' \rightarrow x'') = L_e(x' \rightarrow x'') + \int_M L(x \rightarrow x') f_s(x \rightarrow x' \rightarrow x'') G(x \leftrightarrow x') dA(x)$$

- More complete model of *light transport*
- Basically describes how much *light energy* leaves a surface at a given point (x') in a particular direction (to x") in terms of how much light is incident on the surface from all other points in the world.
  - $L(y \rightarrow z)$ is the amount of light traveling along the ray from point y to point z. L_e is the amount of light emitted by the surface
  - $f_s(x \rightarrow y \rightarrow z, \lambda)$ is the *Bidirectional Reflectance Distribution Function* (BRDF) of the surface. Describes how much of the light incident on the surface at *y* from the direction of *x* leaves the surface in the direction of *z*
  - $G(y \leftrightarrow z)$ is a Geometry term which involves occlusion and the angle between the surfaces
- How do we evaluate this function?
  - very difficult to solve complicated integral equations analytically

What is MLT? (3/4)

- There are many ways to integrate a function
- Simplest method is to take samples of the function uniformly spaced across its domain
  - basic high school calculus integration
  - this is what distributed ray tracing does
  - might have to take many samples if function is very complicated in some area
- A smarter method: importance sampling
  - if we already know something about a function, we can utilize this knowledge
  - sample more in areas which contribute more to final value of the integral

- *Metropolis sampling* is a variety of importance sampling (Metropolis, Rosenbluth, Rosenbluth, Teller, and Teller, 1953)
  - chooses samples in such a way that they are eventually distributed proportionally to the value of the function being integrated

What is MLT? (4/4)

- *Metropolis Light Transport* algorithm (Veach, 1995) reformulated the integral rendering equation as a pure integration problem over the space of all light paths
- Conceptually:

$$\text{Final Image} = \int_P \text{contribution of a single path}$$

  – where P is the space of all possible light paths
- This allows us to apply Metropolis sampling to the complete rendering problem
  – often much faster than previous Monte Carlo methods
  – handles scenes commonly considered to be difficult