Radiosity

EECS 487

March 26, 2007

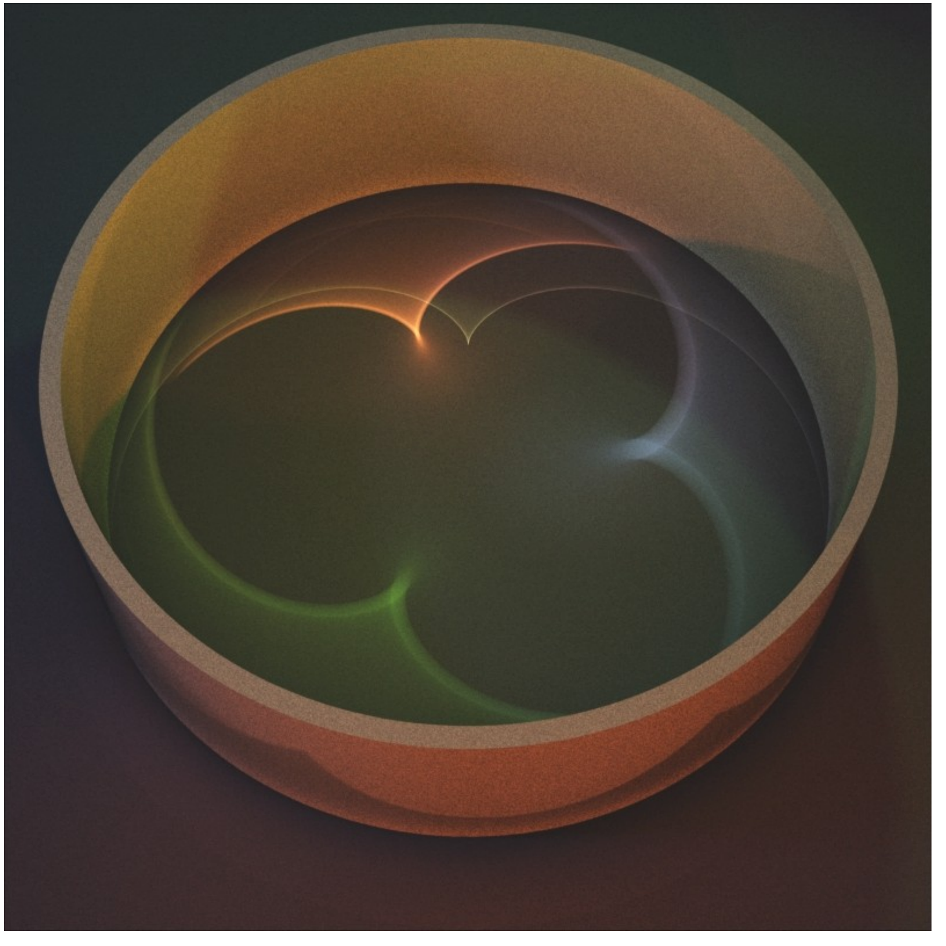
Adapted from slides by John F. Hughes and Andries van Dam

Recap of ray tracing

- what effects are hard for ray tracing?
- why?

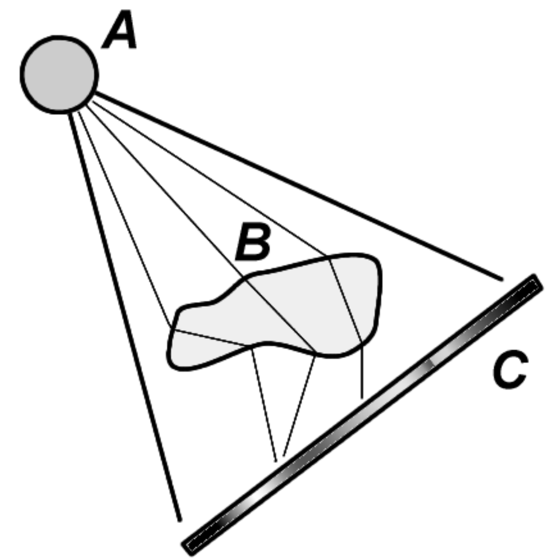
Recap of ray tracing

- why are caustics hard?



Recap of ray tracing

- why are caustics hard?
- need to trace paths from light sources to eye
- ray tracing traces from eye "backwards" out into the scene



Degrees of ray tracing

- ray casting
- ray tracing
- monte carlo ray tracing
- what effects can they achieve?

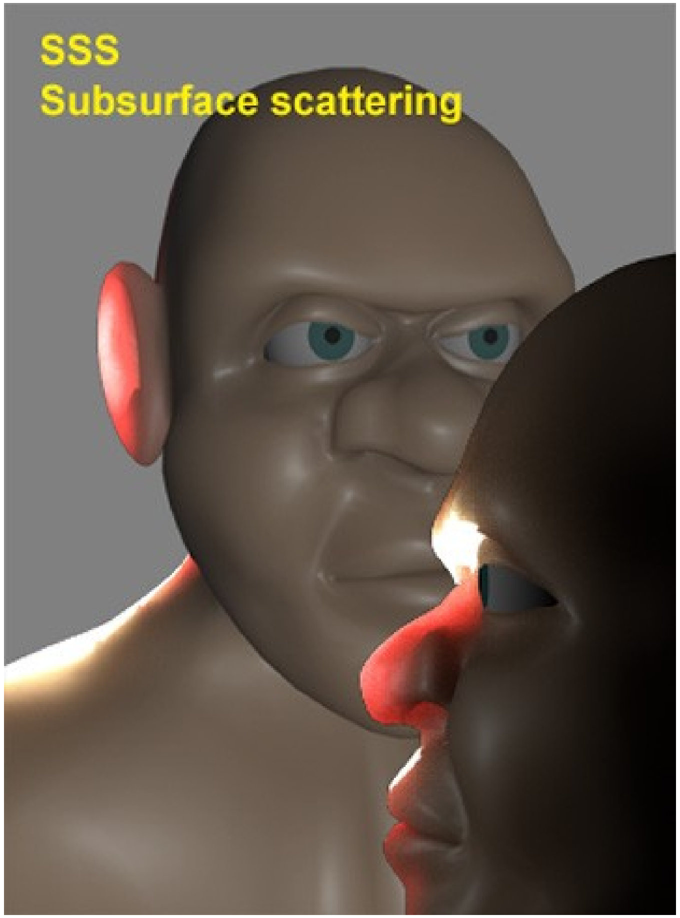
Effects achieved by types of ray tracing

- ray casting
  - local illumination (same as OpenGL)
- ray tracing
  - mirrored surfaces
  - hard shadows
  - refraction (e.g. through glass, water)

Effects achieved by types of ray tracing

- monte carlo ray tracing
  - antialiasing
  - depth of field
  - motion blur
  - soft shadows
  - glossy reflection
  - subsurface scattering
  - color bleeding?
  - caustics?

Subsurface Scattering

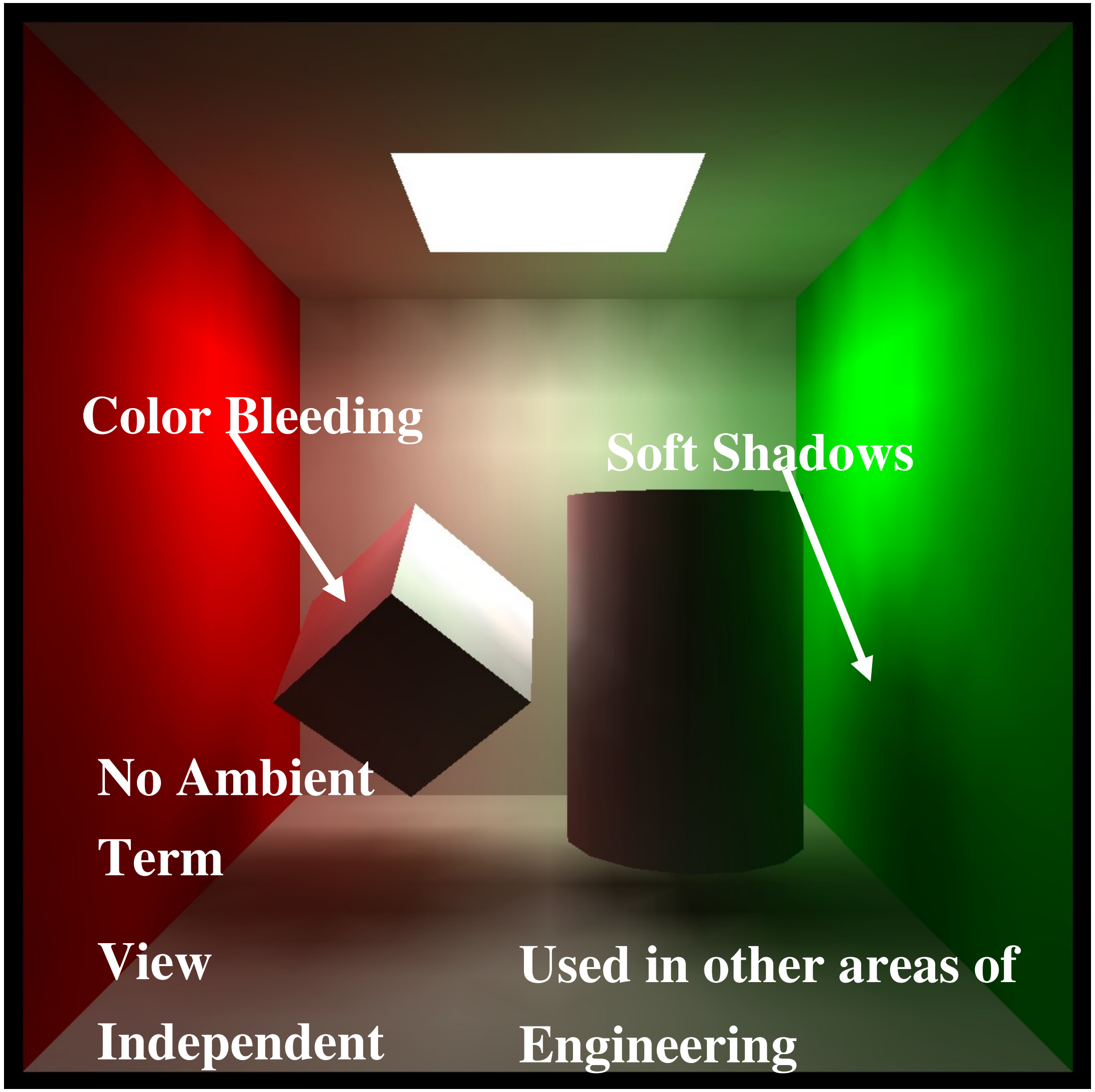


http://features.cgsociety.org/story_custom.php?story_id=2420&page=4

Easy vs. hard

- easy effects need relatively few rays
- hard effects need many rays
- light transport in a hall of mirrors is easy for ray tracing
- much harder for a scene made of diffuse surfaces
- radiosity simulates diffuse light transport
- does not handle specularities

Radiosity for Inter-object Diffuse Reflection



Comparison to photographs

Reality (actual photograph)…

Comparison to photographs

Minus Radiosity Rendering…

Comparison to photographs

Equals the difference (or error) image

Mostly due to mis-calibration

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

Radiosity: overview

- scene represented as many small “patches”
- each patch has a luminance value
  - initially 0 (black)
  - except for light sources
- iteratively simulate light flow between patches to finalize stable values of this luminance
- result is view-independent
  - like a texture map for entire scene!

terms

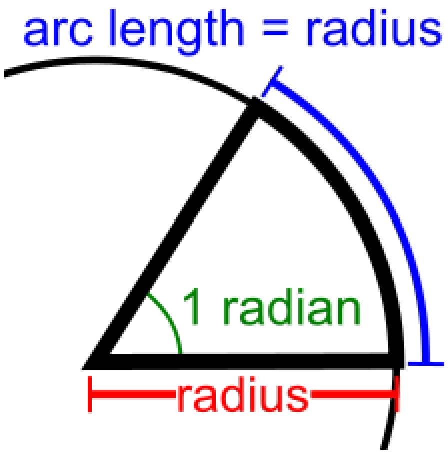
- *Power*: energy flow per unit time
- *Energy Flux*: Energy per unit area per unit time
  - Power per unit area
- *Irradiance*: Total power per unit area incident at a spot
- *Radiant Exitance*: Total power per unit area reflected from a spot

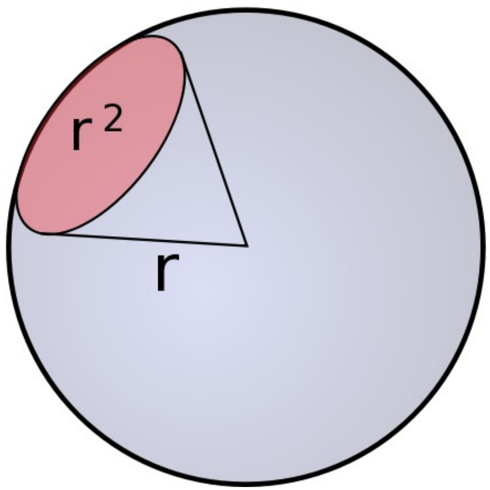
terms

- *Radiance*: Irradiance contribution from a particular incident angle (direction) through a differential solid angle
- *Surface Radiance*: Exiting radiance (exitance)
- *Field Radiance*: Incident radiance

terms

- *Radian*: Angle subtended at center when arc length = radius
- *Steradian*: Solid angle subtended at center when surface portion area is r^2
  - Contour can have any shape





terms

- *Radiosity*: rate at which energy leaves surface
  - Same as Radiant Exitance
- equal to sum of emitted and reflected light
- reflected light depends on:
  - all incoming light
  - surface BRDF
- similar problems in thermal engineering

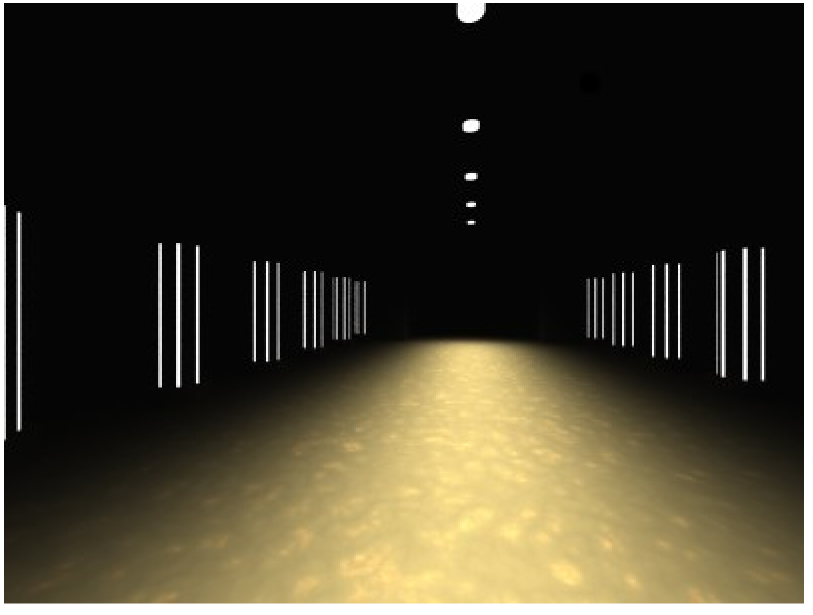
terms

- *BRDF*: Bidirectional Reflectance Distrib Fn
  - For each incident direction, $\mathbf{k}_i$
    - for each reflected direction, $\mathbf{k}_o$
      - what fraction of the incident power is reflected in a differential solid angle around the reflected direction

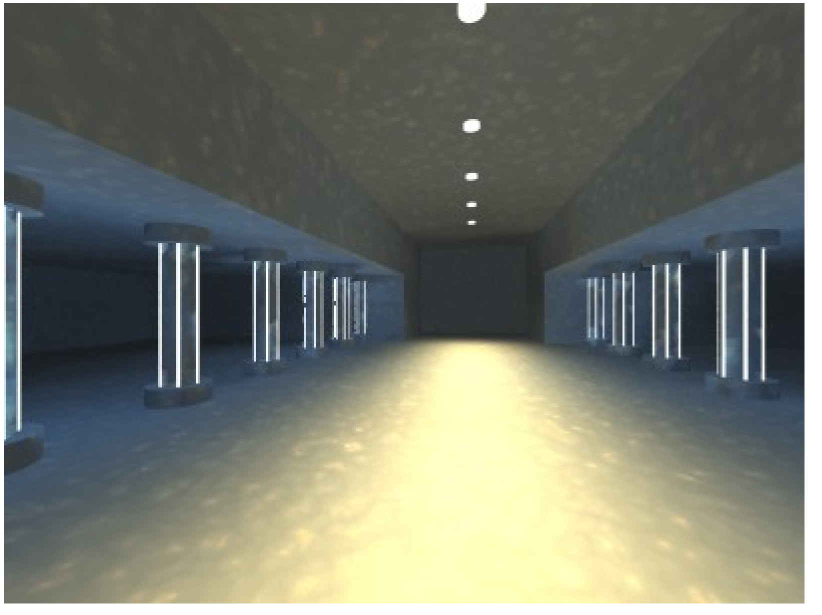
radiosity example

Radiosity - Significance

Without radiosity



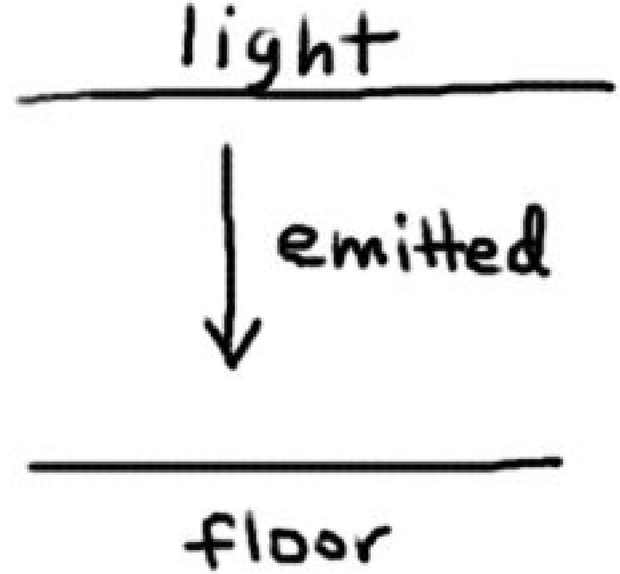
With radiosity



http://bisqwit.iki.fi/kala/povray/radiosity/

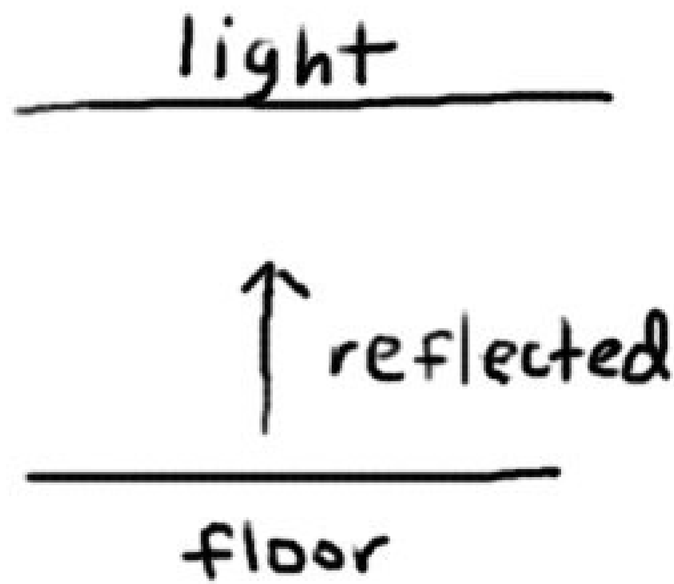
simple example

- two patches
- fluorescent light in ceiling
- floor



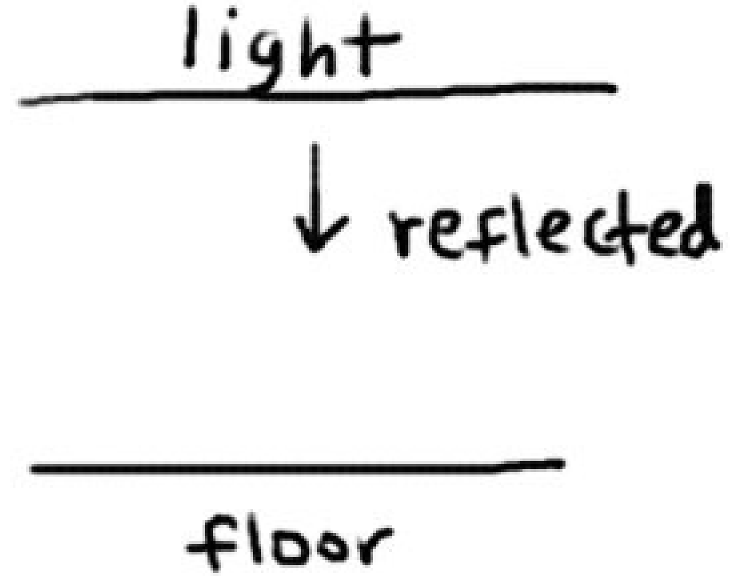
simple example

- two patches
- fluorescent light in ceiling
- floor



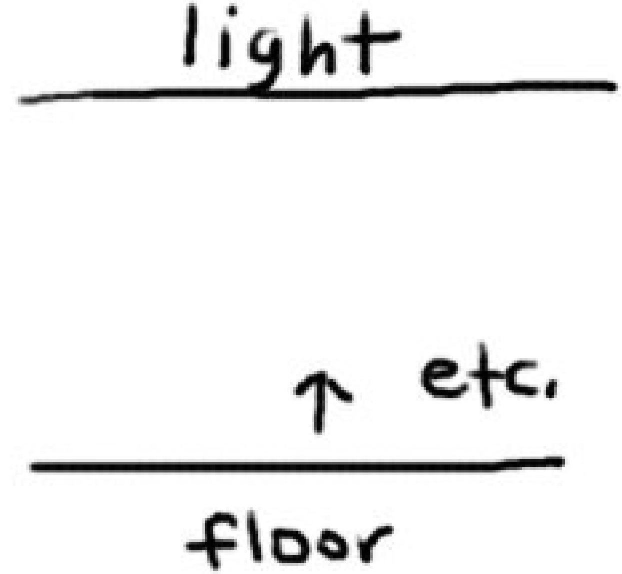
simple example

- two patches
- fluorescent light in ceiling
- floor



simple example

- two patches
- fluorescent light in ceiling
- floor
- less light with each reflection



convergence?

- what do we need for convergence?

convergence?

- what do we need for convergence?
- reflected light should always be less than incoming light
- i.e., absorption is always > 0

Kajiya's Rendering Equation

- the "rendering equation" formulated by Jim Kajiya in 1986
- tells how much light energy leaves patch *j* in a particular direction (to *k*)
- depends on:
  - light emitted from *j* (toward *k*)
  - light arriving at *j* from whole scene, reflected toward *k*

Kajiya's Rendering Equation

$$L(j \to k) = L_e(j \to k) + \int_i L(i \to j)\, g(i \to j)\, f(i \to j \to k)$$

- f is the BRDF of surface at j
  - tells how light arriving at j from all directions is reflected toward k
- g is a "geometry" function
  - tells how much light leaving i towards j arrives without being blocked by an occluder

rendering a scene

- a scene has:
  - geometry
  - light sources
  - camera parameters
- must compute light arriving at camera from all angles, taking into account multiple bounces off geometry

Recap - rendering a scene

- a scene has:
  - geometry
  - light sources
  - camera parameters
- must compute light arriving at camera from all angles, taking into account multiple bounces off geometry
- too hard

approximations

- ray tracing:
  - small number of rays (relatively)
  - mainly handle specular reflection
  - ignore inter-object diffuse reflections
- radiosity
  - handle only diffuse reflection
  - result is view-independent
    (ignores all view-dependent effects)

radiosity: which effects are supported?

- depth of field
- motion blur
- soft shadows
- glossy reflection
- subsurface scattering
- color bleeding
- caustics

radiosity: which effects are supported?

- depth of field
- motion blur
- soft shadows
- glossy reflection
- subsurface scattering
- color bleeding
- caustics

important symbols

- i: patch index
- B_i: total light leaving patch i
- E_i: amount of light emitted from patch i
- G_{ji}: fraction of light emitted by patch j that is received by patch i
  - depends on distance, orientation, occlusion
- R_i: fraction of incoming light that is reflected

how are these terms related?

- total light leaving patch *i* equals
  - light emitted from *i*, plus
  - fraction of incoming light at patch *i* that is reflected

how are these terms related?

$$B_i = E_i + R_i \sum_j G_{ij} B_j$$

In matrix notation:

$$\vec{\mathbf{B}} = \vec{\mathbf{E}} + \mathbf{M}\vec{\mathbf{B}}$$

Q: What do the terms mean?

Q: Which are “known” and which are unknown?

Q: Where is the BRDF?

meaning of the terms

- $\vec{\mathbf{B}}$: vector of radiosity values (1 per patch). Unknown.
- $\vec{\mathbf{E}}$: vector of emissive values (1 per patch) Known.
- **M**: matrix that encodes R_i and G_{ij} values; relates how light leaving all the patches is transported to other patches and reflected. “Known.”

computing radiosity

The unknown $\vec{\mathbf{B}}$ occurs twice in the equation!

Impossible to solve!!

$$\vec{\mathbf{B}} = \vec{\mathbf{E}} + \mathbf{M}\vec{\mathbf{B}}$$

class over

- sorry, radiosity is impossible to compute.
- sigh.

computing radiosity

Oh wait....

$$\vec{\mathbf{B}} = \vec{\mathbf{E}} + \mathbf{M}\vec{\mathbf{B}}$$

$$\Rightarrow \; \vec{\mathbf{B}} - \mathbf{M}\vec{\mathbf{B}} = \vec{\mathbf{E}}$$

$$\Rightarrow \; (\mathbf{I} - \mathbf{M})\vec{\mathbf{B}} = \vec{\mathbf{E}}$$

$$\Rightarrow \; \vec{\mathbf{B}} = (\mathbf{I} - \mathbf{M})^{-1}\vec{\mathbf{E}}$$

computing radiosity

- That's the formal solution
- Q: practical problems in implementing?

computing radiosity

- That's the formal solution
- Q: practical problems in implementing?
  - matrix **M** is n x n, where n is number of patches (large!)
  - inverting matrix is impractical
  - instead use numerical methods to solve
  - e.g. Gaussian elimination

computing radiosity

- can solve via iteration using "intuitive" method
- recall geometric series:

$$\mathbf{I} + \mathbf{M} + \mathbf{M}^2 + \ldots + \mathbf{M}^{k-1} = \mathbf{S}$$

$\mathbf{M} \times (\bullet)$ $$\mathbf{M} + \mathbf{M}^2 + \mathbf{M}^3 + \cdots + \mathbf{M}^k = \mathbf{MS}$$

Subtract $$\mathbf{I} - \mathbf{M}^k = (\mathbf{I} - \mathbf{M})\mathbf{S}$$

computing radiosity

- Assume $\mathbf{M}^k \rightarrow \mathbf{0}$ (why?)

$$\mathbf{I} - \mathbf{M}^k = (\mathbf{I} - \mathbf{M})\mathbf{S}$$

$$\mathbf{I} \approx (\mathbf{I} - \mathbf{M})\mathbf{S}$$

$$\Rightarrow \ (\mathbf{I} - \mathbf{M})^{-1} \approx \mathbf{S}$$

computing radiosity

- Assume $\mathbf{M}^k \rightarrow \mathbf{0}$ (why?)

$$\mathbf{I} - \mathbf{M}^k = (\mathbf{I} - \mathbf{M})\mathbf{S}$$

$$\mathbf{I} \approx (\mathbf{I} - \mathbf{M})\mathbf{S}$$

$$\Rightarrow \; (\mathbf{I} - \mathbf{M})^{-1} \approx \mathbf{S}$$

- Wait, how did we get onto this?

computing radiosity

- How did we get onto this?

$$\vec{\mathbf{B}} = (\mathbf{I} - \mathbf{M})^{-1}\vec{\mathbf{E}}$$

$$(\mathbf{I} - \mathbf{M})^{-1} \approx \mathbf{S}$$

$$\vec{\mathbf{B}} \approx \mathbf{S}\vec{\mathbf{E}} = (\mathbf{I} + \mathbf{M} + \mathbf{M}^2 + \ldots + \mathbf{M}^k)\vec{\mathbf{E}}$$

computing radiosity

- Suggests iterative approach:
- let $\vec{\mathbf{B}}_t$ represent vector of radiosities at iteration t

$$\vec{\mathbf{B}}_0 = \vec{\mathbf{E}}$$

$$\vec{\mathbf{B}}_1 = \mathbf{M}\vec{\mathbf{B}}_0 + \vec{\mathbf{E}} = \left(\mathbf{M}\vec{\mathbf{E}} + \vec{\mathbf{E}}\right) = \left(\mathbf{I} + \mathbf{M}\right)\vec{\mathbf{E}}$$

$$\vec{\mathbf{B}}_t = \mathbf{M}\vec{\mathbf{B}}_{t-1} + \vec{\mathbf{E}} = \left(\mathbf{I} + \mathbf{M} + \cdots + \mathbf{M}^{t-1}\right)\vec{\mathbf{E}}$$

computing radiosity

- pros:
  - iterative solution does not require matrix inversion
  - converges to correct answer
  - preliminary results are approximately correct (can stop early and have useful result)
- cons:
  - matrix is hard to compute
  - takes a lot of memory
  - can rearrange the iteration for faster convergence...

more next time

- also: project 5 out wednesday
- project 4 due then too