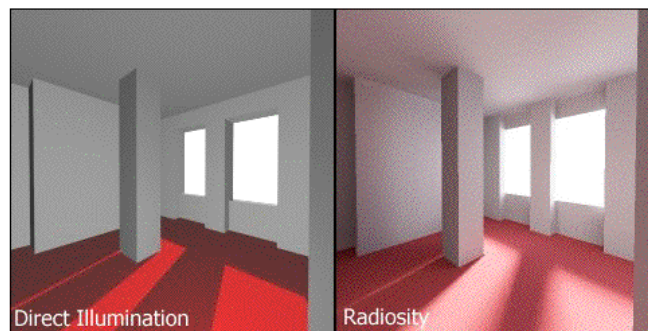




## Radiosity

A method for high-quality rendering of scenes with diffuse reflections and soft shadows.

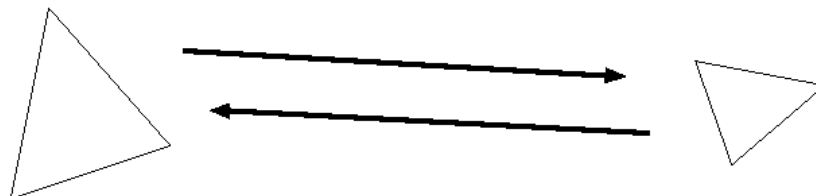


(image from Wikipedia)



## Radiosity

**Problem: Ray-tracing can not accurately model how diffuse light is reflected from object to object!**



**ALL diffusely reflecting objects are turned into light sources!**

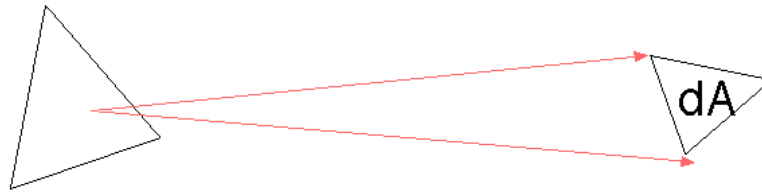
**But – how bright?**



## Radiosity

Radiosity models the amount of energy that is emitted in different directions.

Ideal diffuse reflector => same intensity => energy proportional to the area



The size of a surface element as seen from some other point varies with the angle.



## The model

A surface emits energy that is a sum of reflected and emitted energy:

Energy \* area = emitted + reflected

$$B_k * dA_k = E_k * dA_k + R_k * \int B_j * F_{kj} * dA_k$$

Emitted light  
(true light  
sources only)

Reflectivity

Outgoing  
energy from  
the surface  
element j

Form factor  
between j  
and k



## Simplified to discrete patches

$$B_k = E_k + R_k \sum B_j F_{jk}$$

=> Equation system!

- 1) Solve equation system!
- 2) We must interpolate between patches (e.g. Gouraud shading)
- 3) The form factors  $F_{jk}$  must be calculated



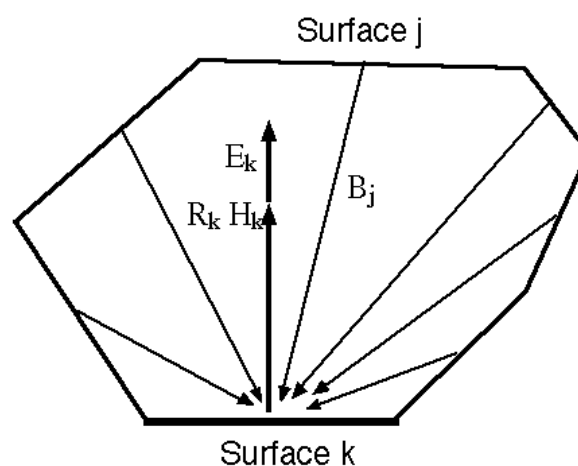
## Radiosity equation

$$H_k = \sum_j B_j F_{jk}$$

$$B_k = E_k + R_k H_k$$

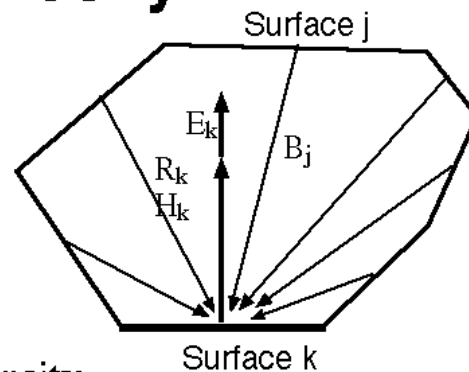
$$B_k = E_k + R_k \sum_j B_j F_{jk}$$

$F_{jk}$  – Form Factor





## Solving radiosity

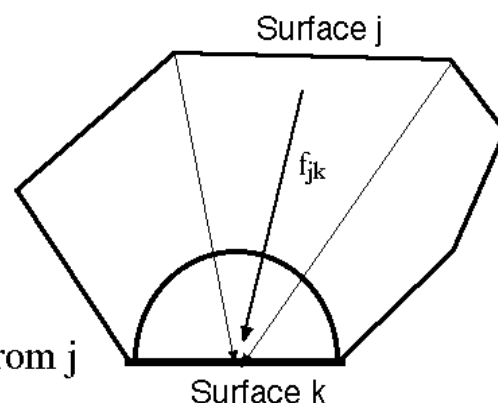


$$B_k = E_k + R_k \sum_j B_j F_{jk}$$

- One equation for each surface
- Solve this system of equations
- Result: radiosity of each surface
- Radiosity = light energy  $\approx$  light intensity
- Solve for R, G, B – get color for each surface
- Render using gouraud shading: nice result guaranteed!
- Problem: calculate Form Factors
- Problem: huge equation system



## Calculating form factors



How much energy comes from surface j to k?

$$f_{jk} = \text{energy on k from j} / \text{total energy from j}$$

Depends on how of j's "view" that is occupied by k, determined by distance, angle, occlusions.

Note! It is what j "see", not what k "see"!



## Calculating $F_{jk}$

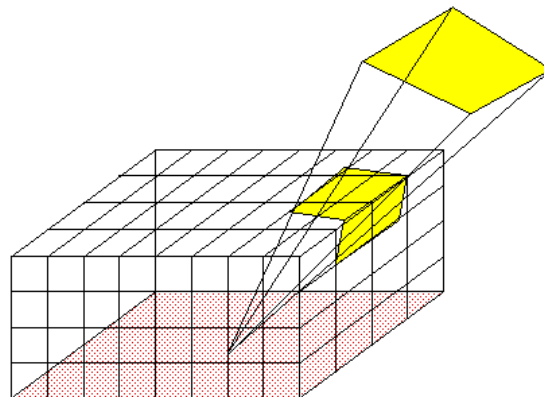
Major problem of radiosity!  $F_{jk}$  calculations take most of the processing time!

$$F_{jk} = (\text{Energy directly from } j \text{ to } k) / (\text{Total energy from } j)$$

This is calculated from the positions, angles and sizes of the two surfaces. All surfaces must be subdivided in parts. More parts give higher realism!



## Hemicube for form factor calculation



Approximates  $f_{jk}$  by calculating projections



## Progressive refinement radiosity

### Step-wise refinement

A method for solving the equation system in real time

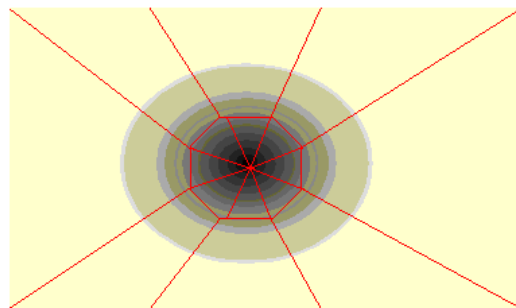
Make one step of emission at a time.

Exact solution takes an infinite number of iterations,  
but a good approximation is found after a few steps.

Preview can be shown instantly!



## Surface subdivision



Surfaces should be sufficiently small

Smaller surfaces where lighting varies

Better form factor approximations



## Weaknesses with radiosity

- Hemicube sampling error
- Insufficient subdivision
- No specular reflections



### Example

(Unknown model)





# Example

I *think* this is the  
"Sibenik Cathedral"  
model.



# Example

(Power Plant  
model, available  
on-line for non-  
commercial use)



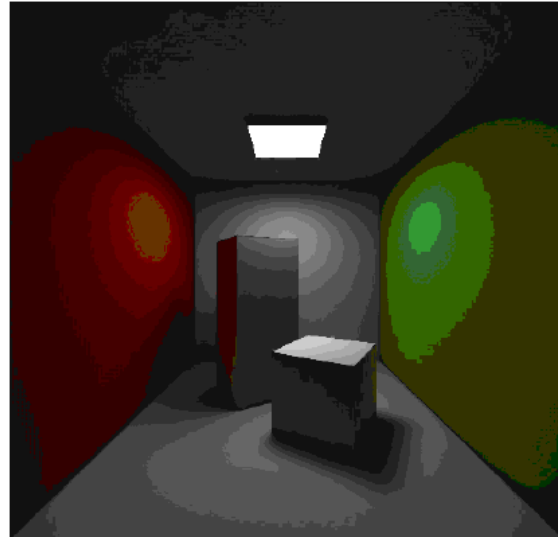




## Example

TSBK03 project on  
progressive  
refinement

(Variant of Cornell  
Box, standard scene  
for illumination)



## Other global illumination models

Very important problem, much research!

- Photon mapping
- Approximation by proximity measurements
  - Hybrid models
- Many variants and parallel implementations



# Backwards ray-tracing

**Illumination Maps  
Photon Maps  
Bidirectional ray-tracing**

**Applies "backwards" light (from light sources) to measure how light is scattered over a scene**

**Gives a good measure of indirect light**



# Illumination maps

**Surfaces split into patches(a grid)**

**Rays traced from light sources**

**Light accumulated for each *vertex***

**Followed by ray-tracing, where diffuse light is interpolated from the illumination map.**



Information Coding / Computer Graphics, ISY, LiTH

## Photon Mapping

Follow rays from light source with ray-tracing methods

Accumulate light in "photon maps"

Saves information about every photon - allows specular surfaces!

Low-pass filter

Then render scene using these maps as surfaces.  
Handles both diffuse and specular reflections!

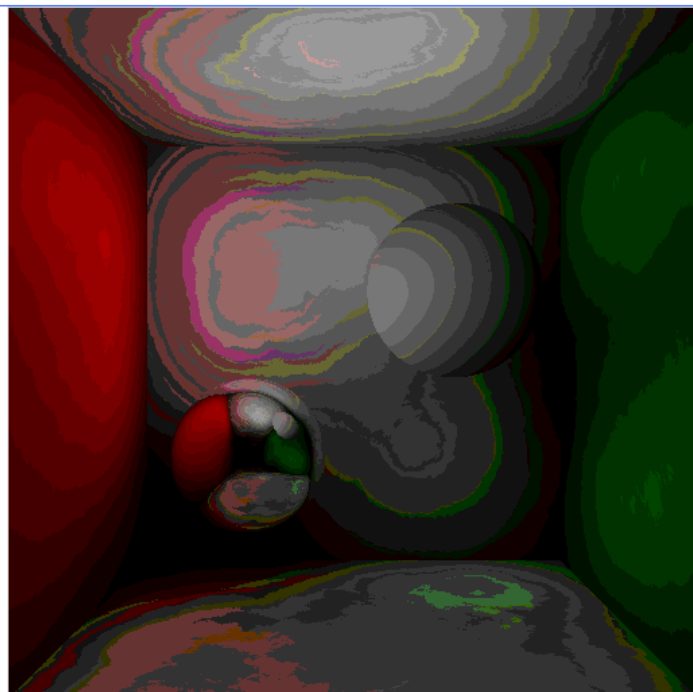


Information Coding / Computer Graphics, ISY, LiTH

## Photon Mapping Example

(The Schindler demo)

Typical features:  
Reflections, caustics  
and diffuse shadows





## Summary

### Ray-tracing:

**Good for shiny surfaces, transparency etc.  
“Hard” images.**

### Radiosity:

**Good for realistic images of diffuse surfaces.  
Can not handle specular reflections!**

**Advanced methods (like Photon Mapping)**