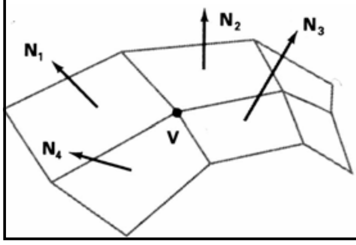
Global Illumination Methods

- Recap: Gouraud, Phong
- Ray-tracing
- Radiosity

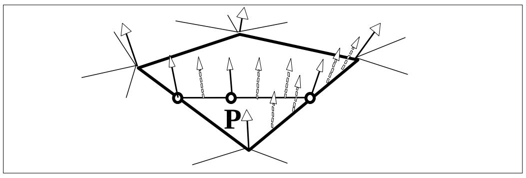
Gouraud Shading Overview

- intensity-interpolation
  - determine average unit normal vector at each polygon vertex
  - apply illumination model to each vertex
  - linearly interpolate vertex intensities



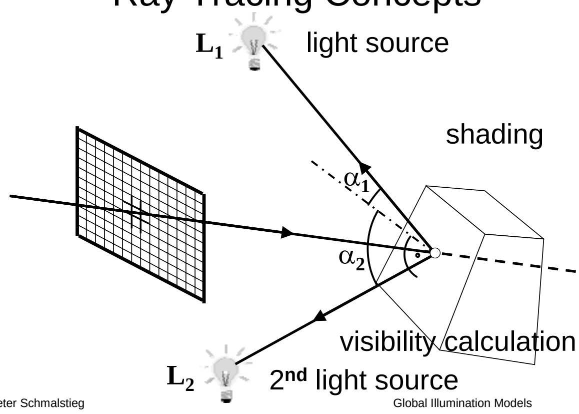
$$N_V = \frac{\sum_{k=1}^{n} N_k}{\left|\sum_{k=1}^{n} N_k\right|}$$

Phong Shading

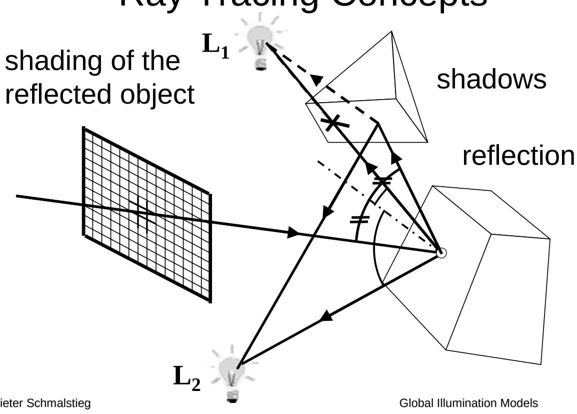


1. normal vectors at corner points
2. interpolate normal vectors along the edges
3. interpolate normal vectors along scanlines & calculate shading (intensities) for every pixel

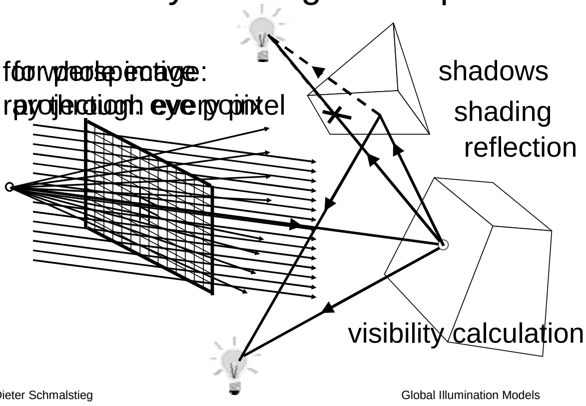
Ray Tracing Concepts



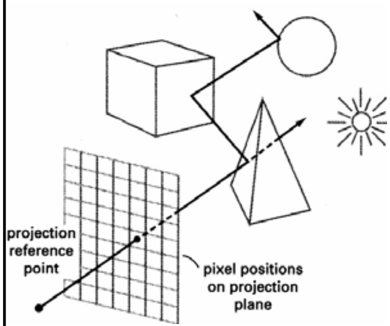
Ray Tracing Concepts



Ray Tracing Concepts



Ray-Tracing Principle



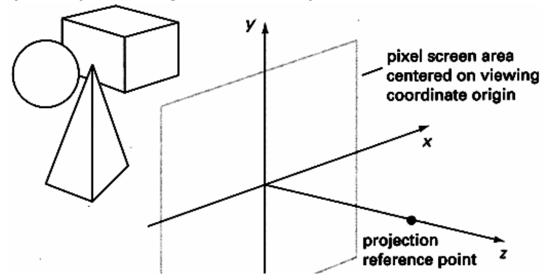
tracing a ray from the projection reference point through a pixel position with multiple reflections and transmissions

Ray-Tracing Properties

- highly realistic images
- very time consuming
- global reflection, transmission
- visible-surface detection
- shadows
- transparency
- multiple light sources

Ray-Tracing

- principles of geometric optics



ray-tracing coordinate reference frame

Shading: Diffuse Shading

$\boxed{\mathbf{I_d} = \mathrm{xxx}}$

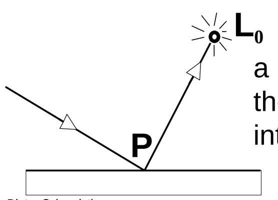
I_d … illumination caused by diffuse shading
xxx … any shading model
(Phong, Blinn, Cook/Torrance,…)

Ray-Tracing: Shadows

ray = intersection point + s · vector to light source

ray = $\mathbf{P} + \mathbf{s} \cdot (\mathbf{L_0} - \mathbf{P})$

$\mathbf{P}$ …intersection point
$\mathbf{L_0}$…light source position



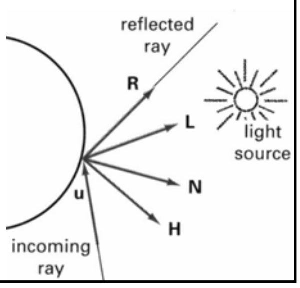
a light source influences the result only if there is no intersection with $\mathbf{0} < \mathbf{s} < \mathbf{1}$

Ray-Tracing: Shadows and Shading

- shadow ray along L
- ambient light $k_a I_a$
- diffuse reflection $k_d(N\cdot L)$
- specular reflection $k_s(H\cdot N)^{n_s}$

$$\boxed{\mathbf{I_d} = k_d I_a + k_d(N\cdot L) + k_s(H\cdot N)^{n_s}}$$

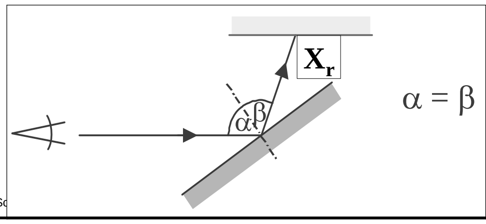
unit vectors at the surface of an object intersected by an incoming ray along direction u



Ray-Tracing: Reflection

$$\boxed{I_r = k_r \cdot X_r}$$

I_r … illumination caused by reflexion
k_r … reflection coefficient of the material
X_r … shading in the reflected direction

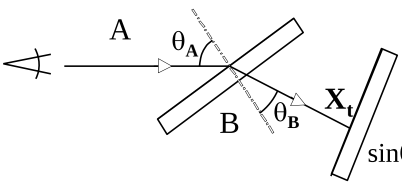


$\alpha = \beta$

Ray-Tracing: Transparency

$$\boxed{I_t = k_t \cdot X_t}$$

I_t … illumination caused by transparency
k_t … transparency coefficient of the material
X_t … shading in the transparency direction

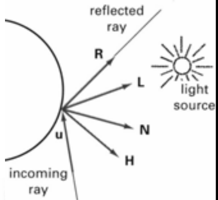


$\sin\theta_A : \sin\theta_B = \eta_B : \eta_A$

Ray-Tr.: Reflection & Transparency

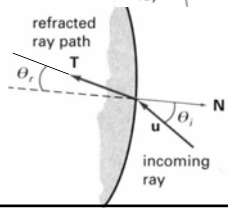
- reflection ray

$R = u - (2u\cdot N)N$



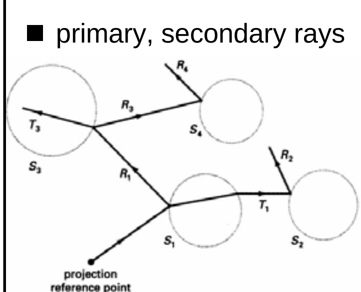
- transparency ray
  - Snell's law

$$\sin\theta_r = \frac{\eta_i}{\eta_r}\sin\theta_i$$

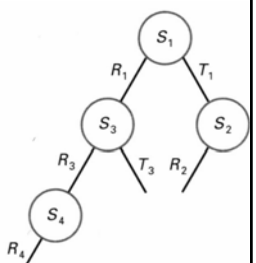


Ray-Tracing: Rays & Ray Tree

- primary, secondary rays



reflection and refraction ray paths for one pixel



corresponding binary ray-tracing tree

Ray-Tracing: A Complete Shading Method

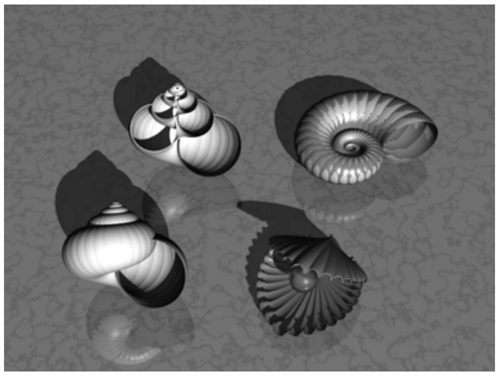
$$\boxed{I = I_d + I_r + I_t}$$

additional requirement: $k_d + k_r + k_t \le 1$

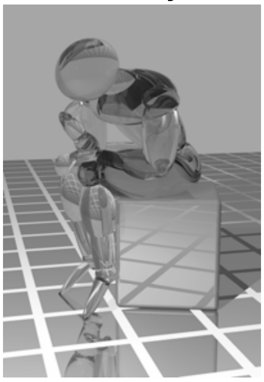
Ray-Tracing: Basic Algorithm

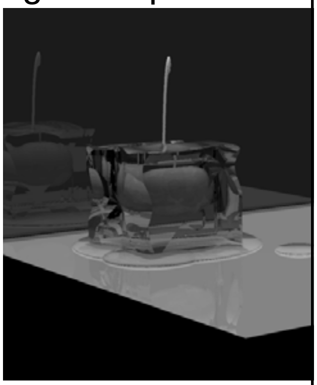
```
FOR all pixels p DO
1.trace primary ray eye -> p
  find closest intersection S
2.FOR all light sources L DO
    trace shadow feeler S -> L
    IF no inters. between S, L
    THEN shading+=influence of L
3.IF surface of S is reflective
  THEN trace secondary ray;
    shading+=influence of refl.
4.IF surface of S is transparent
  THEN trace secondary ray;
    shading+=influence of transp.
```


Ray-Tracing Examples



Ray-Tracing Examples





True Global Illumination Example

„Professional“ Raytracing

www.oyonale.com

Requirements for Object Data

(to use them for ray-tracing)

- intersection calculation ray - object possible
- surface normal calculation possible
  - B-Rep: simple
  - CSG: recursive evaluation

Ray-Surface Intersection

- ray equation

$$P = P_0 + s \cdot u$$

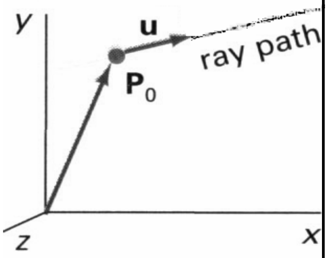
- for primary rays

$$u = \frac{P_{pix} - P_{prp}}{|P_{pix} - P_{prp}|}$$

- for secondary rays

$$u = R$$

$$u = T$$



describing a ray with an initial-position vector P_0 and unit direction vector u

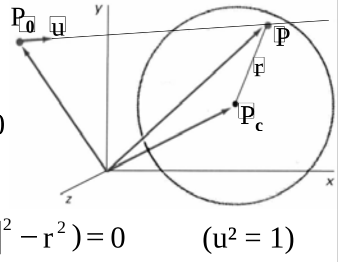
Ray-Sphere Intersection

- parametric ray equation inserted into sphere equation

$$|P - P_c|^2 - r^2 = 0$$

$$|(P_0 + su) - P_c|^2 - r^2 = 0$$

$$\Delta P = P_c - P_0$$



$$s^2 - 2(u \cdot \Delta P)s + (|\Delta P|^2 - r^2) = 0 \qquad (u^2 = 1)$$

$$s = u \cdot \Delta P \pm \sqrt{(u \cdot \Delta P)^2 - |\Delta P|^2 + r^2}$$

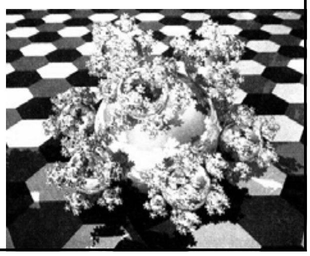
Ray-Sphere Intersection

- discriminant negative $\Rightarrow$ no intersections

$$\mathbf{s = u \cdot \Delta P \pm \sqrt{(u \cdot \Delta P)^2 - |\Delta P|^2 + r^2}}$$

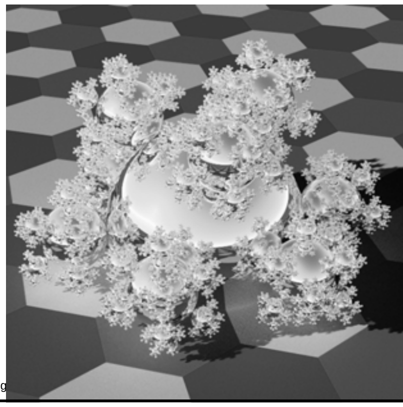
$$\Rightarrow \quad \mathbf{s = u \cdot \Delta P \pm \sqrt{r^2 - |\Delta P - (u \cdot \Delta P)u|^2}} \qquad \text{because } u^2=1$$

(to avoid roundoff errors when $r^2 << |\Delta P|^2$)



"sphereflake": 7381 spheres 3 light sources

Sphereflake



Ray-Polyhedron Intersection

- use bounding sphere to eliminate easy cases
- locate front faces $u \cdot N < 0$
- solving plane equation

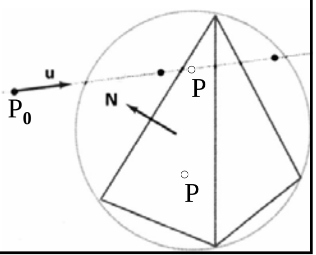
$$Ax + By + Cz + D = 0$$

$$N = (A, B, C)$$

$$N \cdot P = -D$$

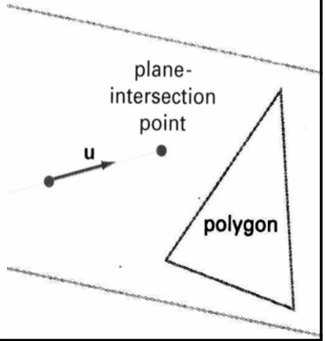
$$N \cdot (P_0 + su) = -D$$

$$s = -\frac{D + N \cdot P_0}{N \cdot u}$$



Ray-Polyhedron Intersection

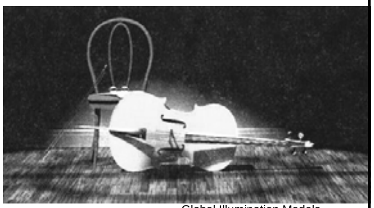
- intersection point inside polygon boundaries?
- inside-outside test
- smallest s to inside point is first intersection point of polyhedron



Ray-Surface Intersection

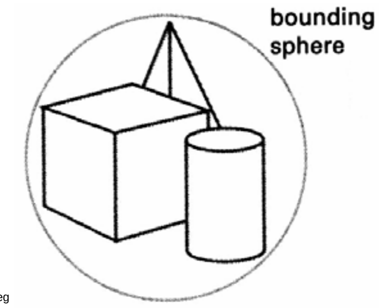
- quadric, spline surfaces:
  - parametric ray equation inserted into surface definition
  - methods like numerical root-finding, incremental calculations

ray-traced scene with global reflection of texture



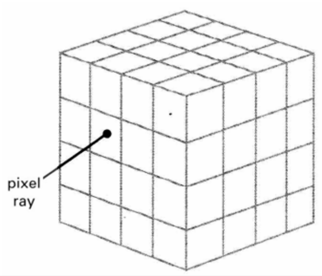
Reducing Object-Intersection Calc.

- bounding volumes
- bounding volume hierarchies



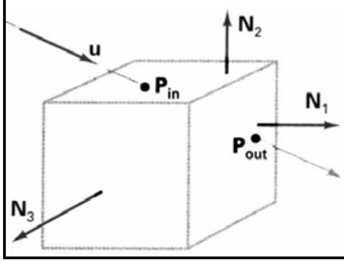
Reducing Object-Intersection Calc.

- space-subdivision methods
  - regular grid
  - octree



Reducing Object-Intersection Calc.

- space-subdivision methods
  - incremental grid traversal
    - 3D Bresenham
    - processing of potential exit faces

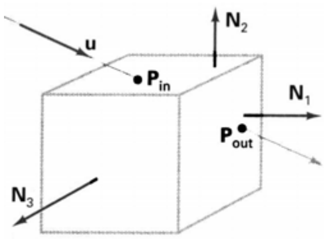


ray traversal through a subregion of a cube enclosing a scene

Incremental Grid Traversal

- ray direction u / ray entry position $P_{\mathbf{in}}$
- potential exit faces $u \cdot N_{\mathbf{k}} > 0$
- normal vectors

$$N_{\mathbf{k}} = \begin{cases} (\pm 1,0,0) \\ (0,\pm 1,0) \\ (0,0,\pm 1) \end{cases}$$



- check signs of components of u

Incremental Grid Traversal

- calculation of exit positions, select smallest s_k

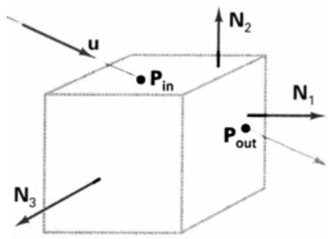
$$P_{\mathbf{out},\mathbf{k}} = P_{\mathbf{in}} + s_{\mathbf{k}} u$$

$$N_{\mathbf{k}} \cdot P_{\mathbf{out},\mathbf{k}} = -D_{\mathbf{k}}$$

$$s_{\mathbf{k}} = \frac{-D_{\mathbf{k}} - N_{\mathbf{k}} \cdot P_{\mathbf{in}}}{N_{\mathbf{k}} \cdot u}$$

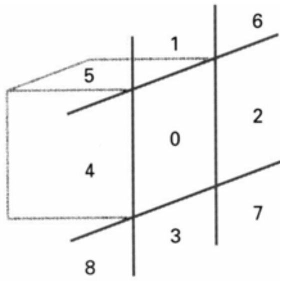
- example:

$$N_{\mathbf{k}} = (1,0,0) \quad s_{\mathbf{k}} = \frac{-x_{\mathbf{k}} - x_{\mathbf{0}}}{u_{\mathbf{x}}}$$



Incremental Grid Traversal

- variation: trial exit plane
  - perpendicular to largest component of u
  - exit point in 0
    => done
  - {1, 2, 3, 4}
    => side clear
  - {5, 6, 7, 8}
    => extra calc.

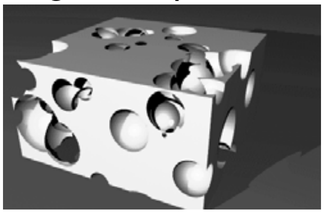


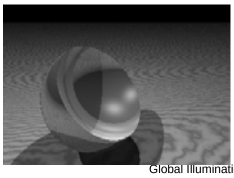
sectors of the
trial exit plane

Ray-Tracing Examples

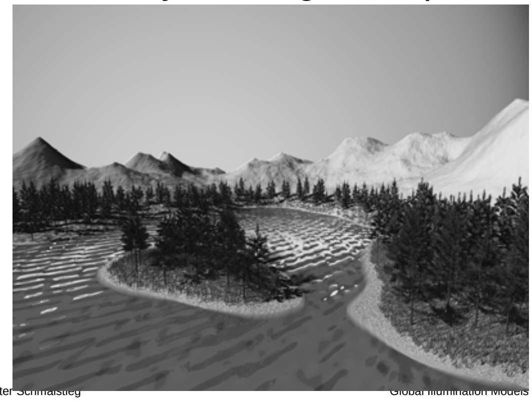
1298 polygons, 4 spheres, 76 cylinders, 35 quadrics
5 light sources

Ray-Tracing Examples





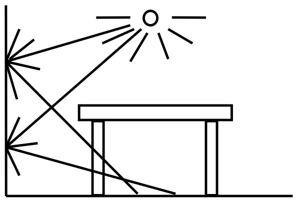
Ray-Tracing Example



1 Billion Raytraced Triangles

Radiosity Method

- describes the physical process of light distribution in a diffuse reflecting environment

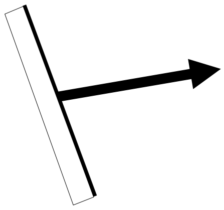


areas that are not illuminated directly are also not completely dark

every object acts as a secondary light source

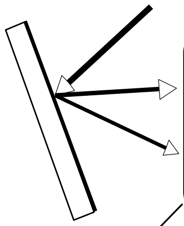
Radiosity

- Radiosity **B** is the „radiant flux per unit area“ that is leaving a surface



Dieter Schmalstieg — Global Illumination Models

Radiosity Equation



incoming light from the environment $\int_{hemi} I(x)\,dx = \int_{hemi} dB$

self emission (only for light sources) E

reflected light from environment $\rho_k \cdot \int_{hemi} dB$

radiosity of the point $B = E + \rho_k \cdot \int_{hemi} dB$

Dieter Schmalstieg — Global Illumination Models

Radiosity Equation

- to calculate the light influence between surfaces

Radiosity = total light leaving a surface point

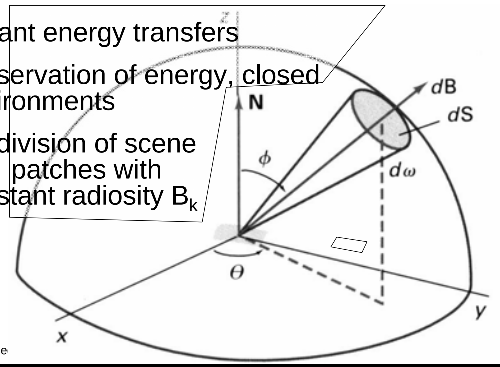
$$B = E + \rho_k \cdot \int_{hemi} dB$$

B…radiosity	hemi…half space over point
E…self emission	ρ_k…reflection coefficient

Dieter Schmalstieg — Global Illumination Models

Radiosity Properties

- diffuse interreflections in a scene
- radiant energy transfers
- conservation of energy, closed environments
- subdivision of scene into patches with constant radiosity B_k



Dieter Schmalstie

Radiosity: Subdivision into Patches

the scene is discretized into **n "patches"** (plane polygons) $\mathbf{P_k}$, for each of these patches a constant radiosity $\mathbf{B_k}$ is assumed:

$$B = E + \rho_k \cdot \int_{hemi} dB \quad \Rightarrow \quad B_k = E_k + \rho_k \cdot \sum_{j=1}^{n} B_j \cdot F_{jk}$$

ρ_k	diffuse reflection coefficient of patch k
F_{jk}	“formfactor”: describes how much % of the influence on patch k comes from patch j; geometric size

Dieter Schmalstieg — Global Illumination Models

Radiosity Model

$$B_k = E_k + \rho_k \sum_{j \neq k} B_j F_{jk}$$

- B_k radiosity of patch k
- E_k self-emission of patch k
- $\Sigma B_j F_{jk}$ contribution of other patches
- F_{jk} form factor, contribution of B_j to B_k
- ρ_k reflectivity factor of patch k

Dieter Schmalstieg — Global Illumination Models

Radiosity Equation

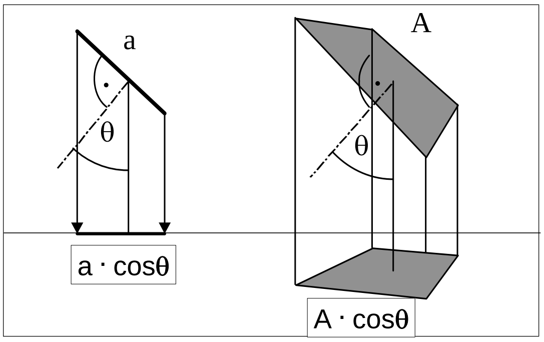
- solving the radiosity equation

$$B_k = E_k + \rho_k \sum_{j \neq k} B_j F_{jk}$$

$$B_k - \rho_k \sum_{j \neq k} B_j F_{jk} = E_k$$

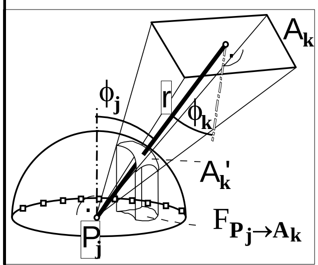
$$\begin{bmatrix} 1 & -\rho_1 F_{21} & \cdots & -\rho_1 F_{n1} \\ -\rho_2 F_{12} & 1 & \cdots & -\rho_2 F_{n2} \\ \vdots & \vdots & & \vdots \\ -\rho_n F_{1n} & -\rho_n F_{2n} & \cdots & 1 \end{bmatrix} \cdot \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_n \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_n \end{bmatrix}$$

Projection of a Polygon



Radiosity: Form Factors

- form factor F_{jk}: contribution of patch j to patch k



$$F_{dA_j, dA_k} = \frac{\cos\phi_j \cos\phi_k dA_k}{\pi r^2}$$

$$\Sigma F_{jk} = 1$$

Radiosity: Form Factors

- form factor F_{jk}: contribution of patch j to patch k

$$F_{dA_j, dA_k} = \frac{\cos\phi_j \cos\phi_k dA_k}{\pi r^2}$$

emitted energy from dA_j on A_k

$$F_{dA_j, A_k} = \int_{A_k} \frac{\cos\phi_j \cos\phi_k}{\pi r^2} dA_k$$

form factor is average over area A_j

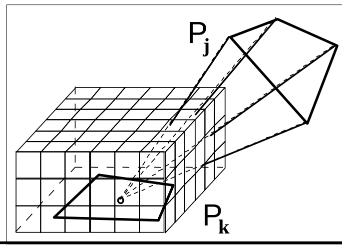
$$F_{jk} = \frac{1}{A_j} \int_{A_j} \int_{A_k} \frac{\cos\phi_j \cos\phi_k}{\pi r^2} dA_k dA_j$$

Radiosity: Form Factors

- form factor properties
  - conservation of energy $\sum_{k=1}^{n} F_{jk} = 1$
  - uniform light reflection $A_j F_{jk} = A_k F_{kj}$
  - no self-incidence $F_{jj} = 0$

Radiosity: Form Factors

- form factor calculation
  - most expensive step in radiosity calculation
  - numerical integration (Monte Carlo methods)
  - hemicube approach (hemisphere replaced by hemicube)



Radiosity Equation

- solving the radiosity equation
  - Gaussian elimination
  - Gauss-Seidel iteration

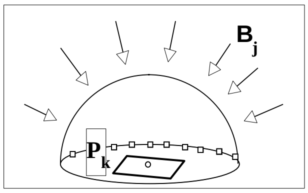
$$\begin{bmatrix} 1 & -\rho_1 F_{21} & \cdots & -\rho_1 F_{n1} \\ -\rho_2 F_{12} & 1 & \cdots & -\rho_2 F_{n2} \\ \vdots & \vdots & & \vdots \\ -\rho_n F_{1n} & -\rho_n F_{2n} & \cdots & 1 \end{bmatrix} \cdot \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_n \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_n \end{bmatrix}$$

- very time and storage intensive

Radiosity Equation

- solving the radiosity equation
  - Gauss-Seidel iteration

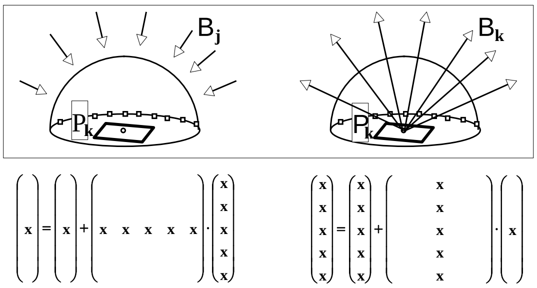
$$B_k^{i+1} = E_k + \rho_k \sum_{j \neq k} B_j^{\,i} F_{jk}$$



"gathering"

Radiosity Equation

- "gathering" vs. "shooting" $B_k^{i+1} = E_k + \rho_k \sum_{j \neq k} B_j^{\,i} F_{jk}$



Progressive Refinement Radiosity (1)

- "shooting"
  - select brightest patch k and distribute its radiosity B_k

$$B_k = E_k + \rho_k \sum_{j \neq k} B_j F_{jk} \Rightarrow \begin{array}{l} B_{k \text{ due to } B_j} = \rho_k B_j F_{jk} \\ B_{j \text{ due to } B_k} = \rho_j B_k F_{kj} \end{array}$$

$$\Downarrow$$

$$\boxed{B_{j \text{ due to } B_k} = \rho_j B_k F_{jk} \frac{A_j}{A_k}} \Leftarrow A_k F_{kj} = A_j F_{jk}$$

Progressive Refinement Radiosity (2) [one refinement step]

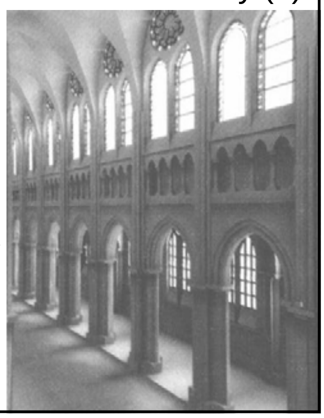
```
for selected patch k
/* set up hemicube,
   calculate form factors Fjk */

for each patch j {
   Δrad := ρj*Bk*Fjk*Aj/Ak
   ΔBj := ΔBj + Δrad
   Bj := Bj + Δrad
}
ΔBk := 0
```


Progressive Refinement Radiosity (3)

- initially $\Delta B_k = B_k = E_k$,
- select patch with highest $\Delta B_k A_k$

cathedral rendered with progressive refinement radiosity

form factors computed with ray-tracing methods



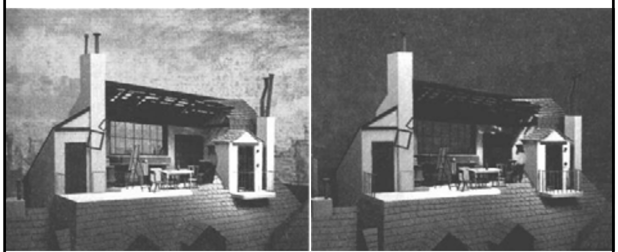
Radiosity Example Images (1)

image of a constructivist museum rendered with progressive refinement radiosity

Radiosity Example Images (2)

stair tower of a building at Cornell University rendered with progressive refinement radiosity

Radiosity Example Images (3)



2 lighting schemes for an opera production:
(left) day view (right) night view