UNIVERSITY OF EXETER

Global illumination

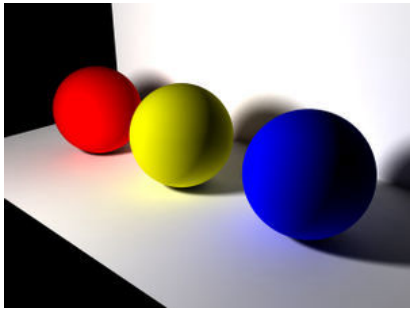
COM3404

Richard Everson

School of Engineering, Computer Science and Mathematics
University of Exeter

R.M.Everson@exeter.ac.uk
http://www.secamlocal.ex.ac.uk/studyres/COM304

Outline



References

- Fundamentals of 3D Computer Graphics. Watt. Chapters 10 & 12.
- Computer Graphics: Principles and Practice. Foley et al (1995).
- http://freespace.virgin.net/hugo.elias/radiosity/radiosity.htm
- "An empirical comparison of radiosity algorithms," A.J. Willmott and P.S. Heckbert, Carnegie Mellon Technical Report CMU-CS-97-115 http://www.cs.cmu.edu/~radiosity/emprad-tr.html

Global illumination

Illumination $I(\mathbf{x}, \mathbf{x}')$ at $\mathbf{x}$ from surface at $\mathbf{x}'$ is sum of

- light emitted by surface at $\mathbf{x}'$, denoted $\epsilon(\mathbf{x}, \mathbf{x}')$
- sum of light scattered from $\mathbf{x}'$ towards $\mathbf{x}$ from other surfaces $\mathbf{x}''$

Mathematically:

$$I(\mathbf{x}, \mathbf{x}') = g(\mathbf{x}, \mathbf{x}') \left[\epsilon(\mathbf{x}, \mathbf{x}') + \int_S \rho(\mathbf{x}, \mathbf{x}', \mathbf{x}'') I(\mathbf{x}', \mathbf{x}'') \, d\mathbf{x}'' \right]$$

$g(\mathbf{x}, \mathbf{x}')$ *visibility function*. 0 if $\mathbf{x}$ is not visible from $\mathbf{x}'$; otherwise $1/\text{dist}(\mathbf{x}, \mathbf{x}')^2$

$\rho(\mathbf{x}, \mathbf{x}', \mathbf{x}'')$ *bi-directional reflectance distribution function*, BRDF.

- Fraction of energy arriving at $\mathbf{x}'$ from $\mathbf{x}''$ that is scattered towards $\mathbf{x}$.
- Appearance depends on illumination (wavelength), the viewing direction, together with material properties
- Phong model is a simple example

Rendering equation

Components

- Model for light emitted by each surface, $\epsilon()$
- A model for the BRDF $\rho()$ of each surface
- A method of evaluating the visibility function

Difficulties

Complexity Integral is so complex that it cannot be evaluated analytically.
- Possible numerical approximations by Monte-Carlo integration.
- Practical algorithms reduce complexity by approximation.

View independence Rendering equation is independent of the view.
- View dependent algorithms (eg. ray tracing, path tracing) reduce complexity by calculating illumination from a particular view.
- Radiosity method models view-independent diffuse illumination.

Recursive Expression for $I(\mathbf{x}, \mathbf{x}')$ incorporates $I(\mathbf{x}', \mathbf{x}'')$. Leads to algorithms that recursively retrace the path of light arriving at a pixel in image plane: ray tracing, path tracing, distributed ray tracing.

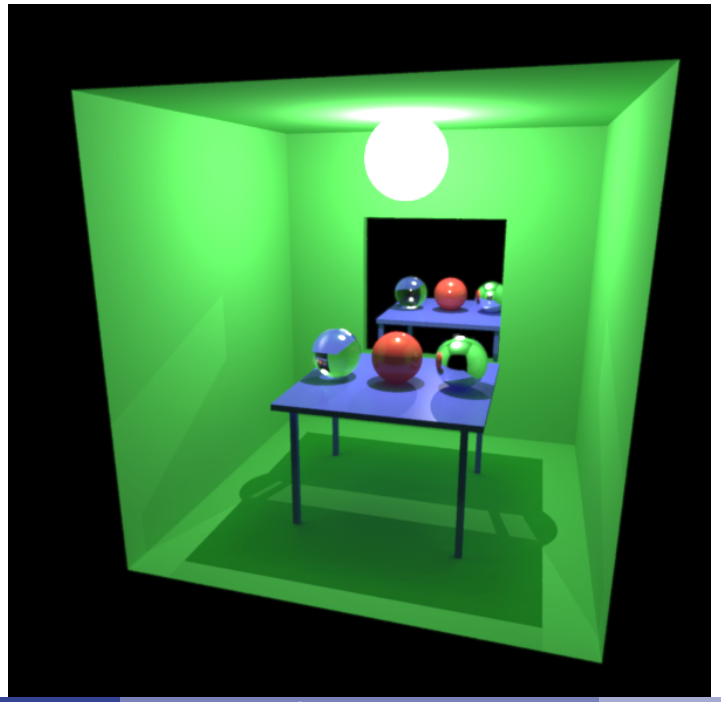
Path notation

Four types of light transfer between diffuse and specularly reflecting surfaces:

Diffuse – Diffuse	A diffuse surface reflects light onto a second diffuse surface
Diffuse – Specular	A diffuse surface reflects light onto a specular surface
Specular – Diffuse	A specular surface reflects light onto a diffuse surface
Specular – Specular	A specular surface reflects light onto a second specular surface

Categorise all paths from light **L** to eye **E** as **L(D|S)*E**

Local reflection models permit LD|SE only



Path notation

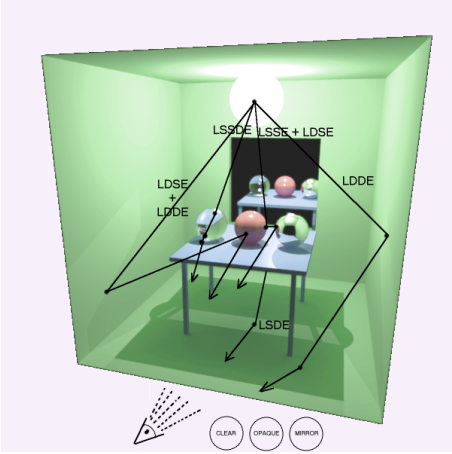
LDDE Shadow of table: light – wall – floor – eye

LSE + LDE Top of opaque ball: light – (specular|diffuse) – eye

LDSE + LDDE Bottom of opaque ball: light – wall – (specular|diffuse) – eye

LSDE Lighter shadow below table: light – mirror – floor – eye

LSSDE Caustic on table top: light – refraction through transparent sphere, S – refraction emerging from sphere, S – reflection at table top – eye



Clear, opaque and mirror balls on a table

Whitted ray tracing

Approximation

- Treat lights as *point* sources
- ρ becomes law for perfect reflection and refraction
- Integral over scene reduced to sum over the path of a single ray

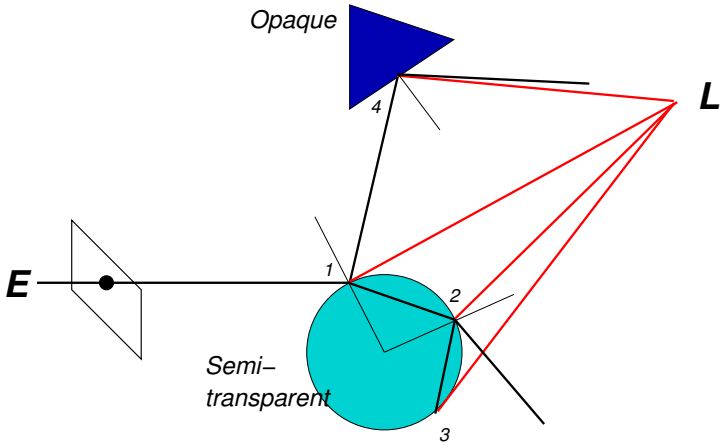
View dependent

Calculation depends on the particular view

Whitted ray tracing

Trace a ray back from each pixel

At each surface: reflected and transmitted rays recursively traced back.
Local specular and diffuse reflection from lights



Whitted ray tracing

1

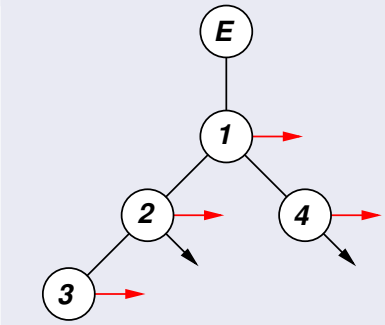
- Reflected ray from triangle traced back
- Refracted ray emerging from 2 traced back
- Local specular and diffuse reflection from L

2

- Reflected ray from 3 traced back
- Refracted ray traced, but hits nothing
- Local specular illumination from L

3

- Local specular illumination



Red arrows indicate light, **L**

4

- Reflected ray traced, but hits nothing
- No refraction/transmission
- Local specular/diffuse

Whitted ray tracing

Global model

Specular reflections from every surface computed and traced

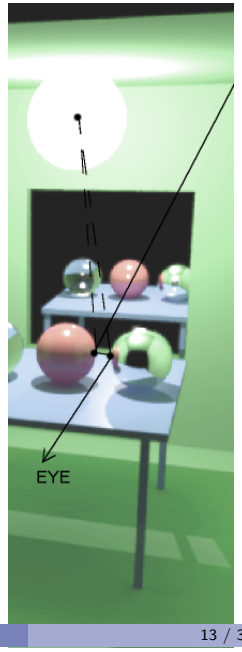
Local model

- Diffuse and specular reflection from light sources computed at every surface
- But diffuse reflections not propagated

Opaque and diffuse spheres

LSSE Global: Light–Opaque–Mirror–Eye
LDSE Local: Light–Opaque–Mirror–Eye
LSE Global: Light–Mirror–Eye

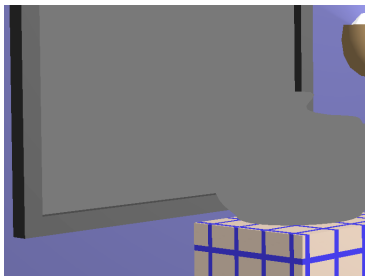
LDS*E + LS*E paths modelled



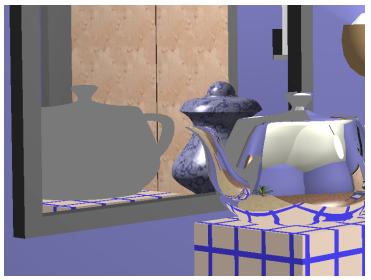
Recursion depth

Recursion Stopped when

- ray hits perfectly diffuse surface
- rays meets scene boundary or bounding sphere.
- recursion depth too great

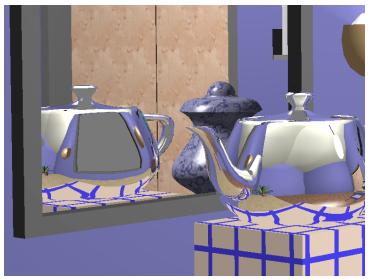


Depth 0

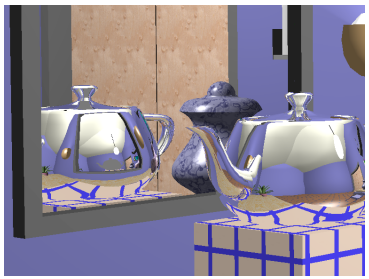


Depth 1

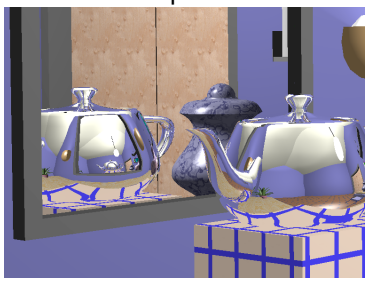
Recursion depth



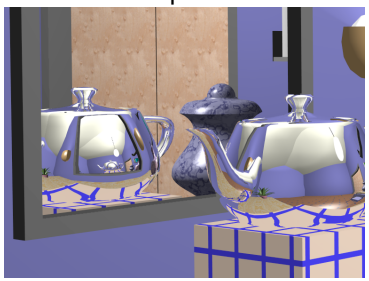
Depth 2



Depth 3



Depth 2



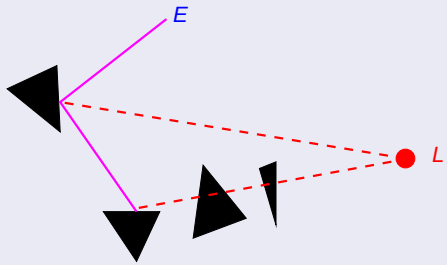
Depth 7 anti-aliased

Shadows

Since light sources are point sources, only perfect, hard-edged shadows – the umbra – are calculated.

Calculate shadows by reducing illumination at a intersection **x** if *shadow ray* from **x** to light intersects another object.

Shadow ray

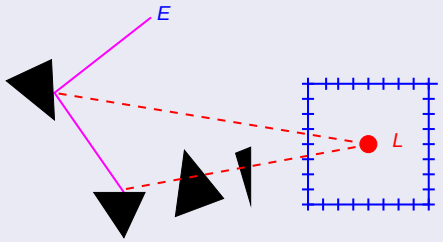


Shadows: Haines-Greenberg buffer

Intersection tests are expensive.

- Does not use coherence
- Reduced by storing polygons that intersect light rays in Haines-Greenberg light buffer.
- Each cell contains a list of polygons/objects and depths intersected by a ray leaving the light through the cell.
- When shadow testing, keep the opaque object which shadowed each light for each ray tree node. Check this object first at that node on the next shadow test as it is likely to be the shadowing object again.
- Do not calculate surface normals etc until it is verified the object is not in shadow.

Haines-Greenberg light buffer



Computation

Recursive algorithm calculating the illumination at **x** as

$$I(\mathbf{x}) = I_{local}(\mathbf{x}) + k_{rg} I(\mathbf{x}_r) + k_{tg} I(\mathbf{x}_t)$$

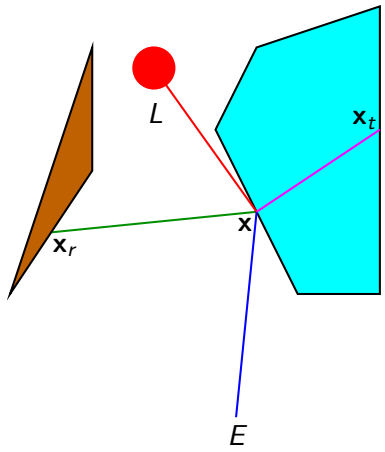
$I_{local}(\mathbf{x})$ Specular and diffuse reflection from lights

$\mathbf{x}_r$ Hit point discovered by tracing reflected ray from **x**

$\mathbf{x}_t$ Hit point discovered by tracing transmitted ray from **x**

k_{rg} Global reflection coefficient

k_{tg} Global transmission coefficient



Computation

ShootRay(`ray`)

- intersection test
- if `ray` intersects an object
- get normal at intersection
- calculate local intensity I_{local}
- `depth--`
- if `depth > 0`
- calculate and shoot reflected ray
- calculate and shoot refracted ray

Computation

calculate and shoot reflected ray

- calculate reflection vector; store in `reflected-ray` structure
- RayOrigin := intersection point
- Attenuate ray: multiply k_{rg} by previous value
- ShootRay(`reflected-ray`)
- if `reflected-ray` intersects an object
- combine colours $k_{rg}I$ with I_{local}

Computation

calculate and shoot refracted ray

- if `ray` is entering object
- accumulate refractive index
- increment number of objects ray is currently inside
- calculate refraction vector in `refracted-ray`
- else
- de-accumulate refractive index
- decrement number of objects ray is currently inside
- calculate refraction vector in `refracted-ray`
- RayOrigin := intersection point
- Attenuate ray by k_{tg}
- if `refracted-ray` intersects an object
- combine colours $k_{tg}I$ with I_{local}

Efficiency

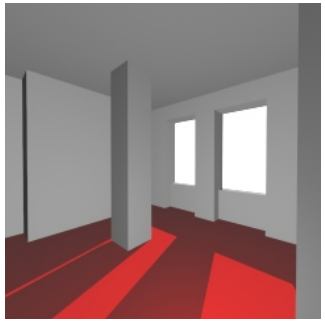
Adaptive depth control
If product of transmission and reflection coefficients is small, stop tracing as contribution of ray is small.

Eliminate first hit intersection calculations
In many scenes rays make only one or two intersections, so using a modified Z-buffer algorithm pre-compute first intersection objects for each pixel in image plane.

Bound objects with simple shapes
Only make detailed intersection calculations if ray is known to enter a simple bounding volume.

Octrees allow rapid location of objects close to the ray's path so intersections need only be calculated for these likely objects. Exploits spatial coherence of objects.

Radiosity



Ray traced

- Spot light outside window, ambient light
- Surfaces are diffuse reflectors
- Some faces invisible
- Hard shadows

Radiosity

- Light source is image of the sky outside room
- Entire room lit and visible
- Soft shadows
- Graduated changes in light across faces
- Colours "bleed" across surfaces

Radiosity

Radiosity: energy per unit area leaving a patch per unit time

- Sum of emitted and reflected energy
- For ith patch

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

radiosity $\times$ area $=$ emitted $+$ reflected from all other patches

B_i radiosity of patch i

dA_i area of patch i

E_i emitted energy per unit area

R_i reflectance of patch i

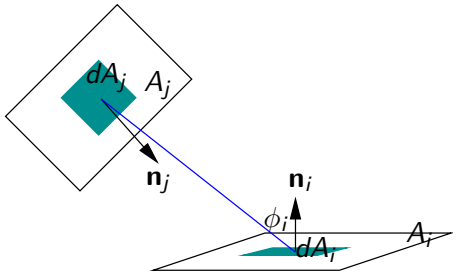
F_{ij} form factor: expresses geometric relationship between patches i and j

Form factor

$$F_{ij} = \frac{\text{Radiative energy leaving } A_i \text{ that arrives at } A_j}{\text{Radiative energy leaving } A_i \text{ in all directions}}$$

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

- Should be modified with the visibility function if patches are obscured
- Function of geometry; computed once for stationary scenes



Discrete approximation

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

- $F_{ij}\,dA_i = F_{ji}\,dA_j$
- Approximate integral by sum over discrete patches

$$B_i = E_i + R_i \sum_{j\neq i} B_j F_{ij}$$

In matrix form:

$$\begin{bmatrix} 1-R_1F_{11} & R_1F_{12} & \dots & -R_1F_{1N} \\ 1-R_2F_{21} & R_2F_{22} & \dots & -R_2F_{2N} \\ \vdots & \vdots & & \vdots \\ 1-R_NF_{N1} & R_NF_{N2} & \dots & -R_NF_{NN} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_N \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_N \end{bmatrix}$$

Radiosity method

$$\begin{bmatrix} 1-R_1F_{11} & R_1F_{12} & \dots & -R_1F_{1N} \\ 1-R_2F_{21} & R_2F_{22} & \dots & -R_2F_{2N} \\ \vdots & \vdots & & \vdots \\ 1-R_NF_{N1} & R_NF_{N2} & \dots & -R_NF_{NN} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_N \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_N \end{bmatrix}$$

$$\mathbf{RB} = \mathbf{E}$$

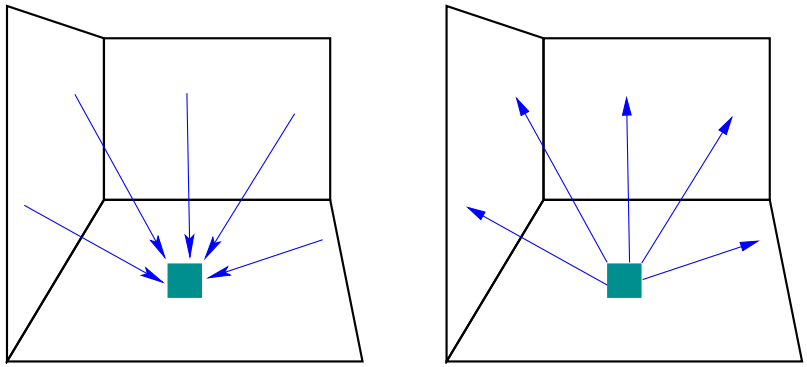
- Emissions E_i are defined by the model
  Only non-zero where for light sources
- Elements of $\mathbf{R}$ are defined by the model and BDRF for this wavelength
- Radiosities are solution to equations: $\mathbf{B} = \mathbf{R}^{-1}\mathbf{E}$

- Separate system of equations for each wavelength or colour of light
- After radiosities are computed patches are rendered with Gouraud/Phong shading to interpolate across patches

Progressive refinement: gathering and shooting

- Direct solution of $\mathbf{B} = \mathbf{R}^{-1}\mathbf{E}$ is expensive in space and time
- Efficient solution by successive refinements of an approximate solution

Repeat until convergence:
  for each patch:
    Gather contributions for patch i from all other patches
    Shoot radiosity from patch i to all other patches



Gathering and shooting

Repeat until convergence:
  for each patch i:
  Gather Contribution of patch j to patch i is

$$B_i = E_i + R_i \sum_{j \neq i} B_j F_{ij}$$

so new estimate at iteration $k+1$ is

$$B_i^{(k+1)} = E_i + R_i \sum_{j=1}^{N} F_{ij} B_j^{(k)}$$

Shoot Contribution of patch i to B_j is $R_j B_i F_{ji}$
Update all patches j with change due to change in radiosity at ith patch, $\Delta B_i = B_i^{(k+1)} - B_i^{(k)}$

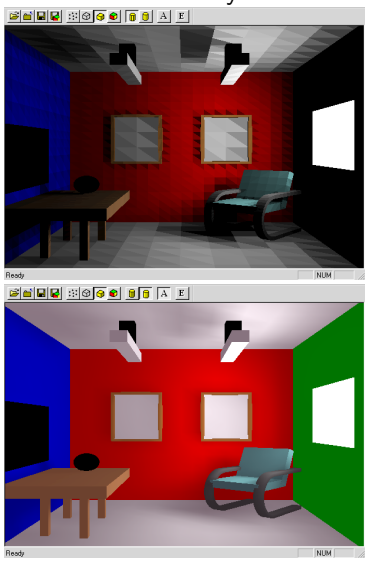
$$B_j^{(k+1)} = B_j^{(k)} + RF_{ji}\Delta B_i$$

Gathering and shooting

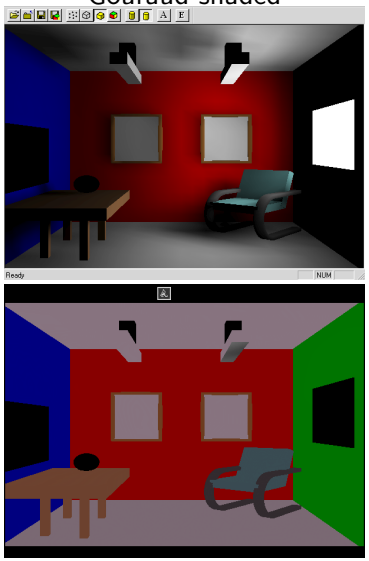
- Treat patches in order of emitting patches first, followed by those that have received a lot of light, etc.
- In the early stages of solution ambient light can be added to make darker parts of the solution visible. Ambient contribution removed as solution is refined.

Example: 20 iterations

Radiosity



Gouraud shaded

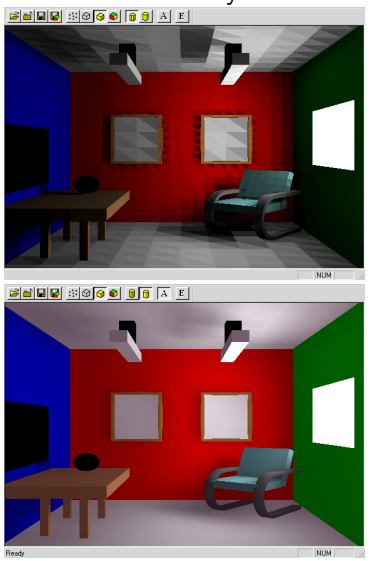


With ambient

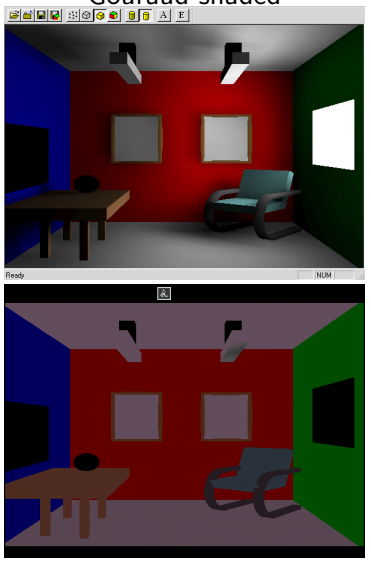
Difference ΔB_i

Example: 250 iterations

Radiosity



Gouraud shaded

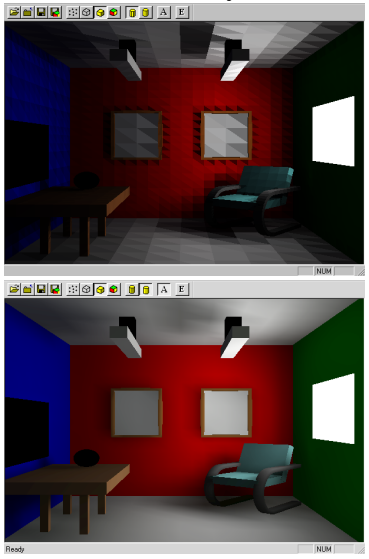


With ambient

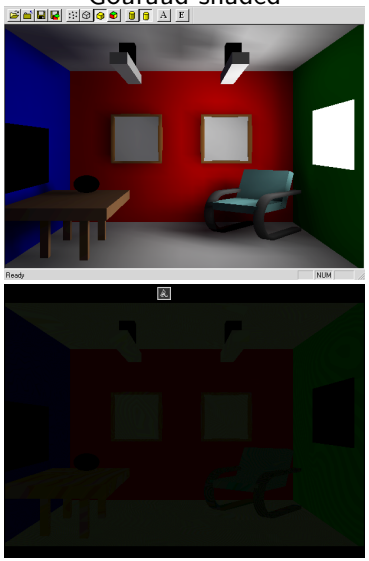
Difference ΔB_i

Example: 5000 iterations

Radiosity



Gouraud shaded

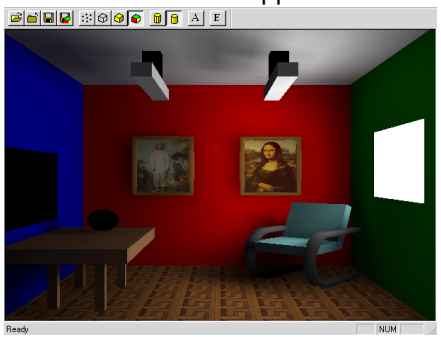


With ambient

Difference ΔB_i

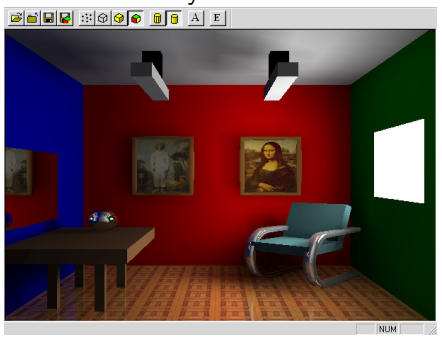
Example:

Texture mapped



Pictures, carpet, etc.

Ray traced



Specular reflections from mirror, ball, shiny surfaces

Form factors

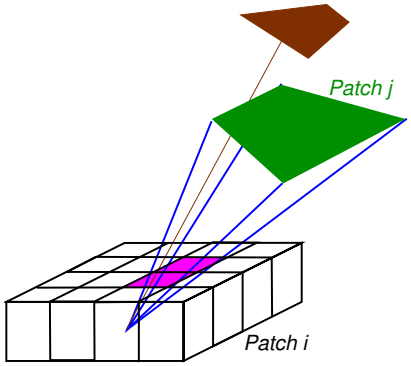
$$F_{ij} = \frac{\text{Radiative energy leaving } A_i \text{ that arrives at } A_j}{\text{Radiative energy leaving } A_i \text{ in all directions}} \approx \int_{A_j} \frac{\cos\phi_i \cos\phi_j}{\pi r^2} \, dA_j$$

if A_i is approximated by a small patch at its centre.

- Calculate F_{ij} by projecting patch j onto a *hemicube* of pixels centred on patch i
- Form factor for each hemicube pixel

$$\Delta F_a = \frac{\cos\phi_i \cos\phi_j}{\pi r^2} \Delta A_a$$

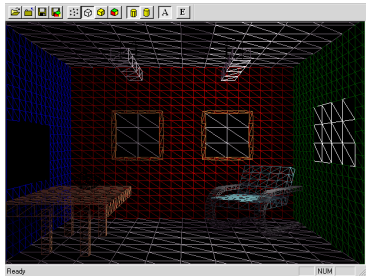
- Projections calculated by modified Z-buffer algorithm

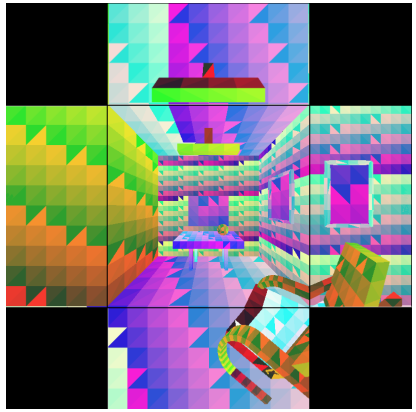


Hemicube

$$F_{ij} \approx \sum \Delta F_a$$

sum is over all pixels in the hemicube to which patch j projects





Hemicube for a patch on the window

Summary

Global illumination method

- Handles diffuse light sources
- Renders shadows naturally
- View independent
- Form factors need only be calculated once for static scenes

Disadvantages

- Computationally expensive, even with hemicube approximation
  - Many iterations required before scene emerges
  - Form factors must be recalculated for moving objects
- Artifacts may result from poor (too coarse) meshing
  - Fine meshes needed where light intensity changes rapidly
  - Adaptive schemes to refine mesh where gradient is high