UCL

Radiosity

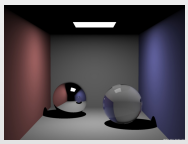
A good book on radiosity:
Radiosity Global Illumination, F. X. Sillion, C. Puech, Morgan Kaufmann publishers, INC., 1994.

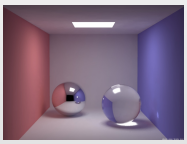
UCL

Introduction

- What is global illumination?

direct lighting
(only from light sources)
+
indirect lighting
(reflections from all surfaces)





UCL

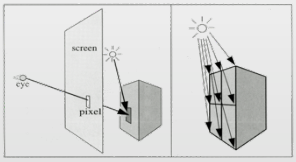
Two approaches for global illumination

- Radiosity
  - View-independent
  - Diffuse only
- Monte-Carlo Ray-tracing
  - Send tons of indirect rays

UCL

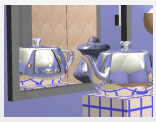
Radiosity vs. Ray Tracing

- Ray tracing is an image-space algorithm
  - Rays are cast through pixels of the viewing window.
  - Can deal with specular and diffuse surfaces
  - If the camera is moved, we have to start over
- Radiosity is computed in object-space
  - View-independent (just don't move the light)
  - Can pre-compute complex lighting to allow interactive walkthroughs



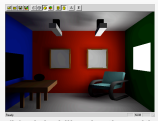
UCL

Global Illumination Approaches



(a) Point light source (ray tracing), global illumination for specular surfaces. (+ textures)

Watt A. Policarpo F. (1998)
"The Computer Image"
ACM Press/Addison-Wesley



(b) global illumination with diffuse surfaces (radiosity)...

(c) … showing colour bleeding (+ textures)

UCL

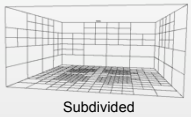
Radiosity methods(I)

- Based on energy exchange principles used in thermal physics.
- *Principle*:
  - Energy is reflected between surfaces.
  - Radiosity method simulates only energy exchange between **diffuse surfaces** (no specularity).
  - Each estimated radiosity quantity is stored with the surface.

UCL

Radiosity methods(II)

- The solution is view independent.
- Based on finite elements.
  - Needs a discrete representation of the 3D scene (mesh).
  - The scene is divided into a set of small areas, or patches.
  - Exchanges will be represented at patches = partitions of the surfaces.
  - Each patch has total energy leaving the surface associated with it (Radiosity). It's constant across the surface.



Subdivided

Exchanges

UCL

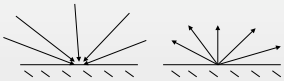
Radiosity Measure

- It is the name of a measure of light energy... (and an algorithm)
  - Radiant energy (flux) = energy flow per unit time across a surface (watts)
  - Radiosity = flux per unit area (a derivative of flux with respect to area) radiated from a surface.
  - These are wavelength-dependent quantities.

UCL

Definitions

- **Radiosity** = outgoing radiant energy from a surface
- **Unit** = energy per unit area per unit time [J/(m^{2}s)]
- **Irradiance** = incoming energy to a surface.



Irradiance Radiosity

- Wavelength dependent
- We often simulate it in RGB channels since it's convenient.

UCL

Radiosity

- Models lighting for diffuse surfaces only.
- Assume polygonal scene
  - polygons divided into n small 'patches'
  - patch i has area A_i
  - radiosity B_i
  - Emits radiosity E_i (if light source)
  - Has surface reflectance ρ_i (diffuse reflection)
    - Diffuse reflectance is $\rho = B/E$
    - E.g., black surface has reflectance of 0 (no light reflected)
      white surface has reflectance of 1 (all light reflected)

UCL

Rendering Equation – Directional to Surface Integration

- Started with integration over directions:

$$L(p, \omega) = L_e(p, \omega) + \int f(p, \omega_i, \omega) L(p^*, -\omega_i) \cos\theta_i \, d\omega_i$$

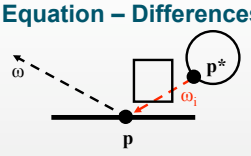
- Change to integration over all surface points p*

$$L(p,\omega) = L_e(p,\omega) + \int f(p,\omega_i,\omega) L(p^*,\omega_i) G(p,p^*) V(p,p^*) dA(p^*)$$

  - G(p,p*) is the geometric term, takes distance and orientation into account
  - V(p,p*) is the visibility term, 0: not visible, 1: visible

UCL

Rendering Equation – Differences to before

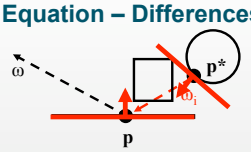


$$L(p,\omega) = L_e(p,\omega) + \int f(p,\omega_i,\omega) L(p^*,\omega_i) G(p,p^*) V(p,p^*) dA(p^*)$$

For each x, compute V(x,x'),
the visibility between x and x':
1 when the surfaces are unobstructed
along the direction ω, 0 otherwise

UCL

Rendering Equation – Differences to before



$$L(p,\omega) = L_e(p,\omega) + \int f(p,\omega_i,\omega)L(p^*,\omega_i)G(p,p^*)V(p,p^*)dA(p^*)$$

For each p, compute G(p, p*), which describes the on the geometric relationship between the two surfaces at p and p*

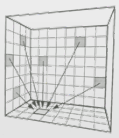
UCL

Radiosity Equation

$$L(p,\omega) = L_e(p,\omega) + \int f(p,\omega_i,\omega)L(p^*,\omega_i)G(p,p^*)V(p,p^*)dA(p^*)$$

Radiosity assumption:
perfectly diffuse surfaces (not directional)

$$B_p = E_p + \rho_p \int B_{p^*} G'(p,p^*)V(p,p^*)\, dA(p^*)$$



Note:
G'() now includes a 1/pi factor

UCL

Relation between Radiance and Radiosity

$$L_{diffuse}(p,\omega) = \int_{\Omega} \frac{\rho}{\pi} L_{in}(p,\omega_i)\cos\theta_i \, d\omega_i$$

$$\rho = \frac{B}{E}$$

$$B(p) = \pi L_{diffuse}(p)$$

UCL

Continuous Radiosity Equation

reflectance

$$B_p = E_p + \rho_p \int G'(p,p^*)\, V(p,p^*)\, B_{p^*}$$

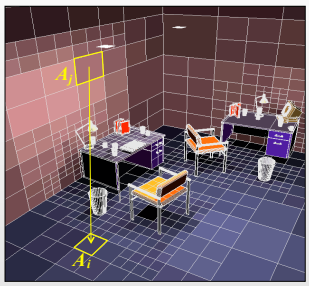
form factor

G: geometry term
V: visibility term

No analytical solution,
even for simple configurations

UCL

Discrete Radiosity Equation



Discretize the scene into *n* patches, over which the radiosity is constant

reflectance

$$B_i = E_i + \rho_i \sum_{j=1}^{n} F_{ij}\, B_j$$

form factor

- discrete representation
- iterative solution
- costly geometric/visibility calculations

UCL

Properties of the form factors

- Form Factors
  - F_{ij} = Form factor between patch i and patch j
  - Fraction of energy leaving patch i which arrives at patch j
- $F_{ii} = 0$ for any convex patch
- Energy conservation:

$$\sum_{j=0}^{n} F_{ij} = 1$$

- Reciprocity relation between form factors

$$F_{ij} A_i = F_{ji} A_j$$

UCL

Discrete Radiosity Equation (2nd)

$$A_i B_i = A_i E_i + \rho_i \sum_{j=1}^{n} A_j B_j F_{ji}$$

Simplification:

$$B_i = E_i + \rho_i \sum_{j=1}^{n} B_j F_{ij} \qquad (i = 1,2,...,n)$$

since $A_i F_{ij} = A_j F_{ji}$

UCL

Radiosity equation becomes

- The radiosity equation becomes:

$$B_i = E_i + \rho_i \sum_{j=0}^{n} F_{ij} B_j$$

or: $B_i - \rho_i \sum_{j=0}^{n} F_{ij} B_j = E_i$

- Which can be written as a matrix form

FB = E

where $F = \delta_{ij} - \rho_i F_{ij}$
where $\delta_{ij} = 1$ when i=j, 0 otherwise

UCL

Solution of Equation

- The B_i are unknown
- Assume all else is known
- Then can be rewritten as system of n linear equations in n unknowns.
- Hence patches can be rendered
  - ideally with smooth shading.
- One set of equations for each wavelength!

UCL

The Radiosity Equation

$$\mathbf{FB} = \mathbf{E}$$

where $\mathbf{B} = \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_n \end{bmatrix}$ and $\mathbf{E} = \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_n \end{bmatrix}$

and $\mathbf{F} = \begin{bmatrix} 1-\rho_1 F_{11} & -\rho_1 F_{12} & \cdots & -\rho_1 F_{1n} \\ -\rho_2 F_{21} & 1-\rho_2 F_{22} & \cdots & -\rho_2 F_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -\rho_n F_{n1} & -\rho_n F_{n2} & \cdots & 1-\rho_n F_{nn} \end{bmatrix}$

UCL

The Radiosity Equation

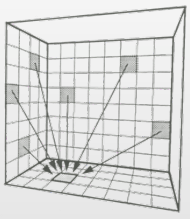
$$\begin{bmatrix} 1-\rho_1 F_{11} & -\rho_1 F_{12} & \cdots & -\rho_1 F_{1n} \\ -\rho_2 F_{21} & 1-\rho_2 F_{22} & & \\ \vdots & & \ddots & \\ -\rho_n F_{n1} & \cdots & \cdots & 1-\rho_n F_{nn} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_n \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_n \end{bmatrix}$$

UCL

Solving the Radiosity Matrix

- Compute **F**
- Radiosity of a single patch *i* is updated iteratively by *gathering* radiosities from all other patches:

$$\begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_i \\ \vdots \\ B_n \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_i \\ \vdots \\ E_n \end{bmatrix} + \begin{bmatrix} & & & \\ \rho_i F_{i1} & \rho_i F_{i1} & \cdots & \rho_i F_{in} \\ & & & \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_j \\ \vdots \\ B_n \end{bmatrix}$$



This method is fundamentally a Gauss-Seidel relaxation

UCL

Form Factors

UCL

Computing the Form Factors

- Form factor between two surface elements, including the visibility

$$F_{dA_i\,dA_j} = \frac{\cos\alpha_i \cos\alpha_j}{\pi r^2} V(dA_i, dA_j)$$

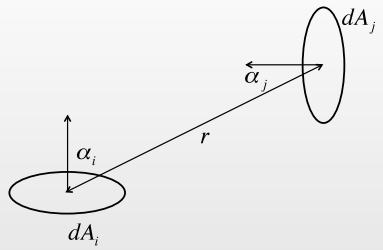
- $V(dA_i, dA_j)$ is the visibility value between patch I and patch j.

$$V(dA_i, dA_j) = \begin{cases} 1 \text{ if } dA_j \text{ is visible from } dA_i \\ 0 \text{ otherwise} \end{cases}$$

- Effect of V = shadows

UCL

Unoccluded Form-Factor



Patch i
to patch j

Differential
Form-Factor

$$F_{dA_i\,dA_j} = \frac{\cos\alpha_i \cos\alpha_j}{\pi r^2}$$

UCL

Analytic Form Factors

$$F_{dA_i A_j} = \int_{A_j} \frac{\cos\alpha_i \cos\alpha_j}{\pi r^2} dA_j$$

From differential area to a patch

$$F_{A_i A_j} = \frac{1}{A_i} \int_{A_i} \int_{A_j} \frac{\cos\alpha_i \cos\alpha_j}{\pi r^2} dA_j dA_i$$

From patch to patch

$A_i F_{ij} = A_j F_{ji}$ follows.

In practice form factors cannot be derived analytically!

Must also take into account visibility between patches!

UCL

Solving the analytic equation?

- No! Too long and difficult
- Simpler:
  - Assume that surfaces are very small compared to the distance r (i.e., use differential to differential/area FF)

$$F_{ij} \approx A_j \frac{\cos\alpha_i \cos\alpha_j}{\pi r^2}$$

UCL

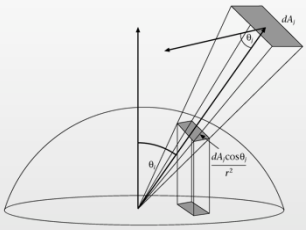
Projection Methods

- Compute the form factor with simplifying assumptions:
  - The distance between two patches is large compared to their area
  - The inner integral ($F_{dA_i A_j}$) does not vary much over the surface A_i.
- Use differential-to-patch form-factor:

$$F_{ij} \approx F_{dA_i A_j} = \int_{A_j} \frac{\cos\alpha_i \cos\alpha_j}{\pi r^2} dA_j$$

UCL

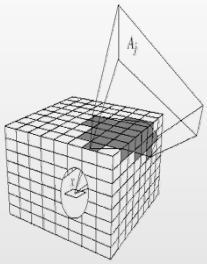
Projection on a Hemi-Sphere



- Nusselt Analogy
- Proportion of projected area on circular base = F_{dAiAj}
- Note - all patches with the same projected area have same form-factor.

UCL

Hemi-Cube Approximation



- Start with patch i
- Place a hemi-cube on its centre.
- Project all other patches onto this hemi-cube.
- The hemi-cube is 'pixelised', with a ΔF_q pre-computed for each grid element and stored in a look-up table
- Use z-buffer for each face – storing which patches remain visible.

UCL

Delta Form-Factors

$$\frac{1}{\pi(x^2 + y^2 + 1)^2}$$

E.g., consider the top face.

The exact form factor between a 'pixel' to the origin can be derived analytically.

Same is true for all such pixels.

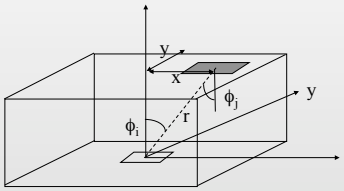
The form-factor is approximated by sum of all delta form-factors of pixels covered by j.

UCL

Delta Form-Factors

$$\Delta F_q = \frac{\cos\phi_i \cos\phi_j}{\pi r^2} \Delta A$$

$$\cos\phi_i = \cos\phi_j = \frac{z}{r} = \frac{1}{\sqrt{x^2 + y^2 + 1}}$$



$$\frac{1}{\pi(x^2 + y^2 + 1)^2}$$

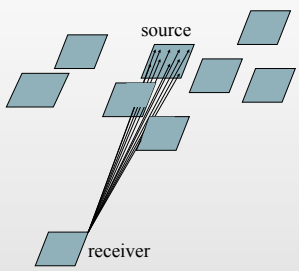
UCL

Problems with Hemi-Cube

- Crude sampling method
  - more ideal would be a uniform subdivision of the hemisphere around a patch centre
    - more computationally intensive (has been tried).
- Leads to aliasing - depending on size of ‘pixels’.
- Gives form-factors (and hence radiosities) at patch centers - not ideal for smooth shading.

UCL

Use Ray Casting



Take a sample of n points on a source patch.

Trace rays to a vertex on receiver patch.

Sum the contributions of all visible rays.

UCL

Ray Casting

- Advantages
  - Can easily obtain the form-factors and radiosities at the vertices (smooth shading)
  - Can vary the sampling scheme
  - Visibility naturally taken into account
- Disadvantages
  - Slower to compute

UCL

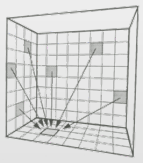
Computing a Radiosity Solution

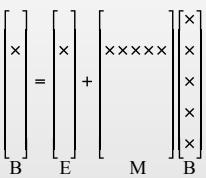
UCL

Gauss-Seidel: Gathering

- Jacobi or Gauss-Seidel algorithm
- At each iteration, compute for all patches a new radiosity
- This method is also called *gathering:* the light incoming from all other surfaces is gathered at a patch

$$B_i^{(k+1)} = E_i + \sum_{j=0}^{n} \rho_i F_{ij} B_j^{(k)}$$





UCL

Gathering: Drawbacks

- If you compute for every patch at each iteration,
  - it is slow: all the form factors have to be computed and the matrix has to be solved before getting an image.
  - it has an excessive storage: all the form factor needs to be retained in memory.

UCL

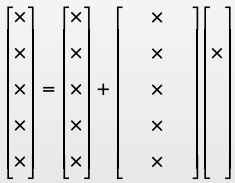
Matrix Solution Impractical

- Suppose there are 10,000 polygons.
  - Each divided into 10 patches (say).
  - n = 100,000
  - n^2 = 10,000,000,000
  - Each form factor 4 bytes (at least)
  - 40,000,000,000 bytes = 40Gb for the matrix.

UCL

Progressive Refinement (PR)

- Solve the equation row by row, without ever storing the full matrix



$$B_j^{(k+1)} = B_j^{(k)} + \rho_j F_{ij} \frac{A_i}{A_j} B_i$$

UCL

Consider Again

- Original formulation:

$$B_i = E_i + \rho_i \sum_{j=1}^{n} B_j F_{ij}$$

- A single term determines influence of j on i:

$$(B_i \text{ due to } B_j) = \rho_i B_j F_{ij}$$

UCL

What is the Contribution of B_i?

Using $A_i F_{ij} = A_j F_{ji}$ and swapping i with j:

$$\left(B_j \text{ due to } B_i\right) = \rho_j B_i F_{ji} = \rho_j B_i F_{ij} \frac{A_i}{A_j}$$

For all patches j: $\left(B_j \text{ due to } B_i\right) = \rho_j B_i F_{ij} \frac{A_i}{A_j}$

UCL

Shooting

- We 'shoot' light out from patch i, allowing it to make a contribution to all other patches j
  - adjusting the radiosities of each of these other patches.
- Note that the form-factors needed (F_{ij}) are all from the i-th row of the matrix - e.g., one hemicube.

UCL

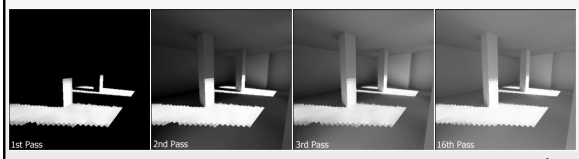
B_i , $\Delta B_i = 0$ for all non-light sources

B_i , $\Delta B_i = E_i$ for sources. *unshot radiosity* ΔB_i

```
for each iteration {
        sort in descending order of A_i ΔB_i ;
        for each patch  i { /*in the descending order*/
                compute the form factors F_ij using a hemi-cube at i;
                for each patch j≠i {
                        ΔRad = ρ_j * ΔB_i * F_ij * (A_i/A_j) ;
                        ΔB_j = ΔB_j + ΔRad;
                        B_j = B_j + ΔRad;
                }
                ΔB_i  = 0;
        }
        /*render scene here if desired*/
}/*until convergence of the B_i */
```

UCL

Example



from wikimedia

UCL

Remarks

1. Choosing the patch which has the greatest residual energy helps to have visual results close to the final solution from the first few iterations.
   Usually, we start with light sources.
2. Note that a patch can shoot energy several times.
3. Substructuring can be used here.
4. An ambient term can be used to help to get a better impression of the final result from the first iterations.
5. The final solution with the 'shooting' algorithm is the same as the one computed with the 'gathering' algorithm.

UCL

Ambient Term

UCL

Ambient Term in PR

- Initially images are very dark due to large amount of un-shot energy
- Un-shot energy is approximated and used in an ambient term
- Purely for display, does not contribute to the final solution

UCL

Ambient Term

- Average un-shot energy: $U = \frac{\sum_{i=0}^{n} \Delta B_i A_i}{\sum_{i=0}^{n} A_i}$
- Average reflectance: $\rho_{avg} = \frac{\sum_{i=0}^{n} \rho_i A_i}{\sum_{i=0}^{n} A_i}$
- Take infinite inter-reflections into account:

$$R = 1 + \rho_{avg} + \rho_{avg}^2 + \rho_{avg}^3 + \rho_{avg}^4 + \ldots = \frac{1}{1 - \rho_{avg}}$$

UCL

Ambient Term

- The ambient term is computed as:

$$Ambient = R \cdot U$$

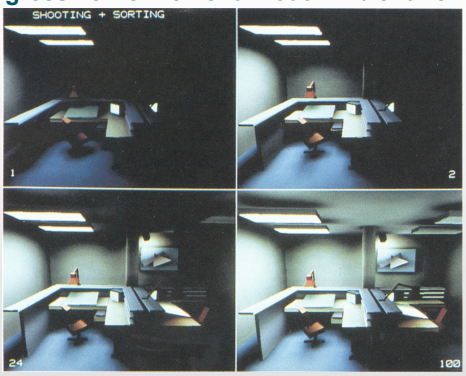
- For the display adjust the radiosities:

$$B_i^{'} = B_i + \rho_i Ambient$$

- As the accuracy solution improves with each iteration, the ambient will decrease and tend to zero

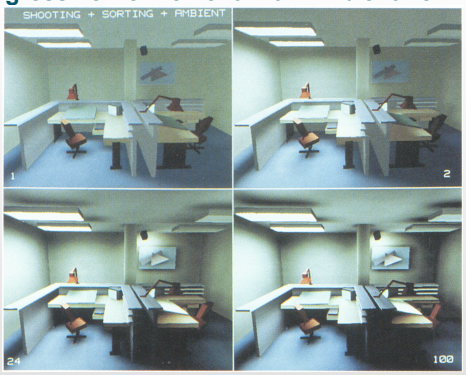
UCL

Progressive Refinement w/out Ambient Term



UCL

Progressive Refinement with Ambient Term



UCL

Meshing

UCL

Meshing

- Choosing a mesh for the patches is the most problematic aspect of radiosity.
- Too coarse, and sharp gradients (shadows) are missed, and the effects are blocky.
- Too fine - increases time/space and oversamples for relatively low gradients.
- Uniform meshing is simplest, suffers from above.

UCL

Adaptive Meshing

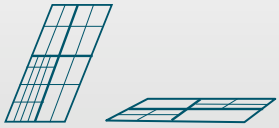
- Only the **receivers** of light need to be finely subdivided
  - and only where there is a sharp radiosity gradient.
- Use an adaptive subdivision scheme:
  - subdivide patch into elements
  - if radiosity for the elements are slowly changing, finish
  - else subdivide the elements and repeat.

UCL

Substructuring

- Create a coarse patch mesh.
- Compute radiosity on patch mesh.
- Find areas of high radiosity gradient.
- Subdivide patches into elements.
- Compute radiosity values on elements from the patch radiosity values.

Patch boundaries
Element boundaries



UCL

Radiosity for the elements

- Patch to patch form factor is

$F_{iq,j}$ = form factor between element q of patch i and j
A_{iq} = Area of element q of patch i
R = Number of elements for patch i

$$F_{ij} = \frac{1}{A_i} \sum_{q=0}^{R} F_{i_q,j} A_{i_q}$$

- Radiosity of element q of patch i

$$B_{i_q} = E_{i_q} + \rho_{i_q} \sum_{j=0}^{n} F_{i_q,j} B_j$$

UCL

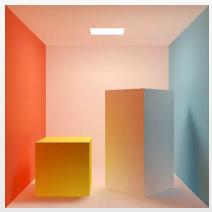
Substructuring: Discussion

- Radiosity is shot only from patches, but is computed at elements.
- Only NxM form factor computations (instead of MxM) where N is the number of patches and M the number of elements.
- Although the obtained solution is comparable to a MxM solution.

UCL

Meshing: Cornell Box

Example using the hemi-cube method. Does not use smooth shading.



Final rendering, with smooth shading and sub-structuring.

UCL

Rendering

- Radiosity is a view-independent solution.
- Could flat shade each patch with colour depending on radiosity at the center (bad solution!)
- Instead obtain radiosities at the vertices of the polygons
  - Use Gouraud smooth shading (interpolation)
  - Available very cheaply on graphics hardware.

UCL

Fin