GLOBAL ILLUMINATION

A Traditional Approach

MATRIX RADIOSITY

A third class final project.
Done at the university of Aleppo, faculty of software engineering, by classmates:
Manal Akta'a, Lama Araban and Razmig Keshishian.
Under supervision by Dr. Eng. Mazen Saeed.

Special thanks to:
Shant Keshishian and Peter Juhani Stuart

BACKGROUND

ABSTRACT
Radiosity is a general term for a group of global illumination algorithms which work under the assumption that all reflections are purely diffuse.
Radiosity is used to produce realistic looking illumation for different kinds of spaces. It is not designed to compete with ray tracing, it has quite different approach.
This overview gives an in-depth study of the basic constant-element radiosity algorithms and outlines the basic solution strategies. Two different methods to compute the form factors are also issued in this paper.

I - INTRODUCTION

Radiosity is a computer graphics method to calculate diffuse light distribution and reflection in three dimensional environments (the "global illumination model").
In contrast to *ray tracing*, all variants of the radiosity method have in common, that they separate the process of shading surfaces and the visible-surface determination. They first compute the interaction of light with all surfaces in a scene, and then determine visibility of the already shaded surfaces from different viewpoints.
Radiosity is efficient to compute scenes up to a certain complexity with a lot of light sources, in essence, each object in the environment is treated as a secondary light source. The method provides an accurate representation of the "diffuse" and "ambient" terms found in typical image synthesis algorithms. The phenomena of "color bleeding" from one surface to another, shading within shadow envelopes, and penumbras along shadow boundaries are accurately reproduced.
Its special strength is in providing the illumination data for many different viewpoints (*animation*) of a scene. Computing very large and complex scenes with radiosity methods requires extreme amounts of memory. Specular effects like metallic reflections are difficult to simulate this way.

II - FRAMEWORK

The radiosity method subdivides the surfaces into small elemental surface patches. Supposing these patches are small, their intensity distribution over the surface can be approximated by a constant value which depends on the surface and the direction of the emission. We can get rid of this directional dependency if only diffuse surfaces are allowed, since diffuse surfaces generate the same intensity in all directions. This is exactly the initial assumption of the simplest radiosity model.
Let the energy leaving a unit area of surface i in a unit time in all directions be B_i, and assume that the light density is homogeneous over the surface. This light density plays a crucial role in this model and is also called the radiosity of surface i. The dependence of the intensity on B_i can be expressed by the following argument:
1. Consider a di_erential dA element of surface A. The total energy leaving the surface dA in unit time is $B \cdot dA$, while the flux in the solid angle $d\omega$ is $d\Phi = I \cdot dA \cdot \cos\phi \cdot d\omega$ if ϕ is the angle between the surface normal and the direction concerned.
2. Expressing the total energy as the integration of the energy contributions over the surface in all directions and assuming diffuse reflection only, we get:

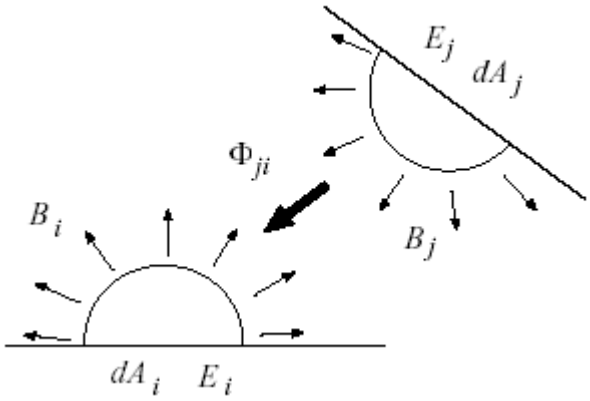
$$B = \frac{1}{dA} \int^{2\pi} \frac{d\Phi}{d\omega} \cdot d\omega = \int^{2\pi} I \cdot \cos\phi \cdot d\omega = I \int_{\theta=0}^{2\pi} \int_{\phi=0}^{\pi/2} \cos\phi \cdot \sin\phi \cdot d\theta d\phi = I \cdot \pi \qquad (1)$$

since $d\omega = \sin\phi \cdot d\theta d\phi$.

Consider the energy transfer of a single surface on a given wavelength. The total energy leaving the surface $B_i \cdot dA_i$ can be divided into its own emission and the diffuse reflection of the radiance coming from other surfaces.

The emission term is $E_i \cdot dA_i$ if E_i is the emission density which is also assumed to be constant on the surface.

The diffuse reflection is the multiplication of the diffuse coefficient ρ_i and that part of the energy of other surfaces which actually reaches surface i.



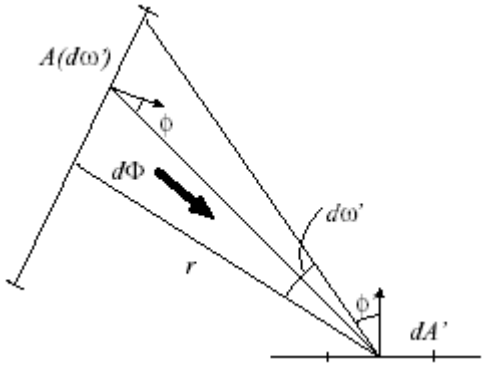
Let F_{ji} be a factor, called the *form factor*, which determines that fraction of the total energy leaving surface j which actually reaches surface i.

Considering all the surfaces, their contributions should be integrated, which leads to the following formula of the radiosity of surface i:

$$B_i \cdot dA_i = E_i \cdot dA_i + \rho_i \cdot \int B_j \cdot F_{ji} \cdot dA_j \qquad (2)$$

Before analyzing this formula any further, some time will be devoted to the meaning and the properties of the form factors. The *fundamental law of photometry* (shown below) expresses the energy transfer between two differential surfaces if they are visible from one another.

$$d\Phi = I \cdot \frac{dA \cdot \cos\phi \cdot dA' \cdot \cos\phi'}{r^2}$$



Replacing the intensity by the radiosity using equation 1, we get:

$$d\Phi = I \cdot \frac{dA_i \cdot \cos\phi_i \cdot dA_j \cdot \cos\phi_j}{r^2} = B_j \cdot \frac{dA_i \cdot \cos\phi_i \cdot dA_j \cdot \cos\phi_j}{\pi \cdot r^2} \qquad (3)$$

If dA_i is not visible from dA_j , that is, another surface is obscuring it from dA_j or it is visible only from the "inner side" of the surface, the energy flux is obviously zero. These two cases can be handled similarly if an indicator variable H_{ij} is introduced:

$$H_{ij} = \begin{cases} 1 \text{ if } dA_i \text{ is visible from } dA_j \\ 0 \text{ otherwise} \end{cases} \qquad (4)$$

Since our goal is to calculate the energy transferred from one finite surface ΔA_j to another ΔA_i in unit time, both surfaces are divided into infinitesimal elements and their energy transfer is summed or integrated, thus:

$$\Delta\Phi_{ji} = \int_{\Delta A_i} \int_{\Delta A_j} B_j \cdot H_{ij} \cdot \frac{dA_i \cdot \cos\phi_i \cdot dA_j \cdot \cos\phi_j}{\pi \cdot r^2} \qquad (5)$$

By definition, the form factor F_{ji} is a fraction of this energy and the total energy leaving surface $j(B_j \cdot \Delta A_j)$:

$$F_{ji} = \frac{1}{\Delta A_j} \cdot \int_{\Delta A_i} \int_{\Delta A_j} H_{ij} \cdot \frac{dA_i \cdot \cos\phi_i \cdot dA_j \cdot \cos\phi_j}{\pi \cdot r^2} \qquad (6)$$

It is important to note that the expression of $F_{ji} \cdot \Delta A_j$ is symmetrical with the exchange of *i* and *j*, which is known as the *reciprocity relationship*:

$$F_{ji} \cdot \Delta A_j = F_{ij} \cdot \Delta A_i \qquad (7)$$

We can now return to the basic radiosity equation.

Taking advantage of the homogeneous property of the surface patches, the integral can be replaced by a finite sum:

$$B_i \cdot \Delta A_i = E_i \cdot \Delta A_i + \rho_i \cdot \sum_j B_j \cdot F_{ji} \cdot \Delta A_j \qquad (8)$$

Applying the reciprocity relationship:

$$B_i \cdot \Delta A_i = E_i \cdot \Delta A_i + \rho_i \cdot \sum_j B_j \cdot F_{ij} \cdot \Delta A_i \qquad (9)$$

Dividing by the area of surface i, we get:

$$B_i = E_i + \rho_i \cdot \sum_j B_j \cdot F_{ij} \qquad (10)$$

This equation can be written for all surfaces, yielding a linear equation where the unknown components are the surface radiosities *(B_i)*:

$$\begin{bmatrix} (1-\rho_1 F_{11}) & -\rho_1 F_{12} & \cdots & -\rho_1 F_{1N} \\ -\rho_2 F_{21} & (1-\rho_2 F_{22}) & \cdots & -\rho_2 F_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ -\rho_N F_{N1} & -\rho_N F_{N2} & \cdots & (1-\rho_N F_{NN}) \end{bmatrix} \cdot \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_N \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_N \end{bmatrix} \qquad (11)$$

The meaning of F_{ii} is the fraction of the energy reaching the very same surface. Since in practical applications the elemental surface patches are planar polygons, F_{ii} is 0.

Both the number of unknown variables and the number of equations are equal to the number of surfaces *(N)*. The solution of this linear equation is, at least theoretically, straightforward.

It can be performed with any standard equation solver. However, a *Gauss-Siedel* iterative approach has a number of advantages.The matrix is well suited to this technique due to the fact that i t is diagonally dominant (the sum of the absolute values of each row is less than the main diagonal term). This is always true since, by definition, the sum of any row of form-factors is equal to unity. In the matrix to be solved, each form factor term is multiplied by its surface reflectivity , which is also less than unity.

Thus the summation of the absolute values of all terms in any row exclusive of the main diagonal term is also less than one. Since, the main diagonal term is always equal to one, the matrix is strictly diagonally dominant, and guaranteed to converge rapidly to a solution.

An initial guess for the radiosities, which must be supplied for the first iteration , is simply the emission of each patch (only the primary light sources have any initial radiosity). During each iteration each radiosity is solved for , using the previously found values of the other radiosities. Iterations continue until no radiosity value changes by more than a preselected small percentage.

The calculated B_i radiosities represent the light density of the surface on a given wavelength. Recalling *Grassman's laws*, to generate color pictures at least three independent wavelengths should be selected (say red, green and blue), and the color information will come from the results of the three different calculations.

Thus, to sum up, the basic steps of the radiosity method are these:

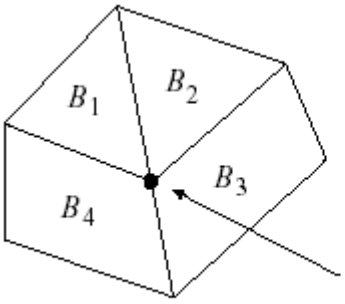
1. F_{ii} form factor calculation.

2. Describe the light emission E_i on the representative wavelengths, or in the simplified case on the wavelength of red, green and blue colors.

Solve the linear equation for each representative wavelength, yielding $B_1, B_2, \ldots, B_N$

3. Generate the picture taking into account the camera parameters by any known hidden surface algorithm. If it turns out that surface i is visible in a pixel, the color of the pixel will be proportional to the calculated radiosity, since the intensity of a diffuse surface is proportional to its radiosity (equ. 1) and is independent of the direction of the camera.

Constant color of surfaces results in the annoying effect of faceted objects, since the eye psychologically accentuates the discontinuities of the color distribution. To create the appearance of smooth surfaces, the tricks of *Gouraud shading* can be applied to replace the jumps of color by linear changes. In contrast to *Gouraud shading* as used in incremental methods, in this case vertex colors are not available to form a set of knot points for interpolation. These vertex colors, however, can be approximated by averaging the colors of adjacent polygons:



$$B_v = \frac{B_1 + B_2 + B_3 + B_4}{4}$$

Note that the first two steps of the radiosity method are independent of the actual view, and the form factor calculation depends only on the geometry of the surface elements.

In camera animation, or when the scene is viewed from different perspectives, only the third step has to be repeated; the computationally expensive form factor calculation and the solution of the linear equation should be carried out only once for a whole sequence.

In addition to this, the same form factor matrix can be used for sequences, when the lightsources have time varying characteristics.

III - FORM FACTOR CALCULATION

The most critical issue in the radiosity method is efficient form factor calculation, and thus it is not surprising that considerable research effort has gone into various algorithms to evaluate or approximate the formula which defines the form factors:

$$F_{ij} = \frac{1}{\Delta Ai} \cdot \int_{\Delta A_i} \int_{\Delta A_j} H_{ij} \cdot \frac{dA_i \cdot \cos\phi_i \cdot dA_j \cdot \cos\phi_j}{\pi \cdot r^2} \qquad (12)$$

Different solutions represent different compromises between the conflicting objectives of high calculation speed, accuracy and algorithmic simplicity.
We conisider the geometrical means of approximating the above double integral.

GEOMETRIC ANALOGUES

The following algorithms focus first on the inner section of the double integration, then estimate the outer integration. The inner integration is given some geometric interpretation which is going to be the base of the calculation. This inner integration has the following form:

$$d_i F_{ij} = \int_{\Delta A_j} H_{ij} \cdot \frac{\cos\phi_i \cdot \cos\phi_j}{\pi \cdot r^2} \cdot dA_j \qquad (20)$$

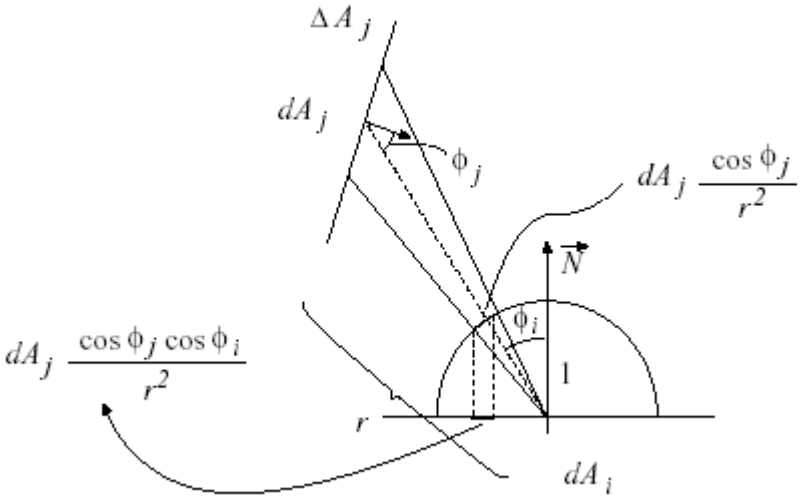
Nusselt has realized that this formula can be interpreted as projecting the visible parts of ΔAj onto the unit hemisphere centered above dA_i, then projecting the result orthographically onto the base circle of this hemisphere in the plane of dA_i, and finally calculating the ratio of the doubly projected area and the area of the unit circle (π). Due to the central role of the unit hemisphere, this method is called the *hemisphere algorithm*.
Later Cohen and Greenberg have shown that the projection calculation can be simplified, and more importantly, supported by image synthesis hardware, if the hemisphere is replaced by a half cube. Their method is called the *hemicube algorithm*.
Beran-Koehn and Pavicic have demonstrated in their recent publication that the necessary calculations can be significantly decreased if a cubic tetrahedron is used instead of the hemicube.

Having calculated the inner section of the integral, the outer part must be evaluated. The simplest way is to suppose that it is nearly constant on ΔA_i, so the outer integral is estimated as the multiplication of the inner integral at the middle of ΔA_i and the area of this surface element:

$$F_{ij} = \frac{1}{\Delta A_i} \int_{\Delta A_i} d_i F_{ij} \cdot dA_i \approx d_i F_{ij} = \int_{\Delta A_j} H_{ij} \cdot \frac{\cos\phi_i \cdot \cos\phi_j}{\pi \cdot r^2} \cdot dA_j \qquad (21)$$

More accurate computations require the evaluation of the inner integral in several points on ΔA_i i and some sort of numerical integration technique should be used for the integral calculation.

HEMISPHERE ALGORITHM



The result of Nusselt is proven using the above figure, which shows that the inner form factor integral can be calculated by a double projection of ΔAj, first onto the unit hemisphere centered above dA_i, then to the base circle of this hemisphere in the plane of dA_i, and finally by calculating the ratio of the double projected area and the area of the unit circle (π). By geometric arguments, or by the definition of solid angles, the projected area of a differential area dA_j on the surface of the hemisphere is $dA_j \cdot \cos\phi_j / r^2$. This area is orthographically projected onto the plane of dA_i, multiplying the area by factor $\cos\phi_j$. The ratio of the double projected area and the area of the base circle is:

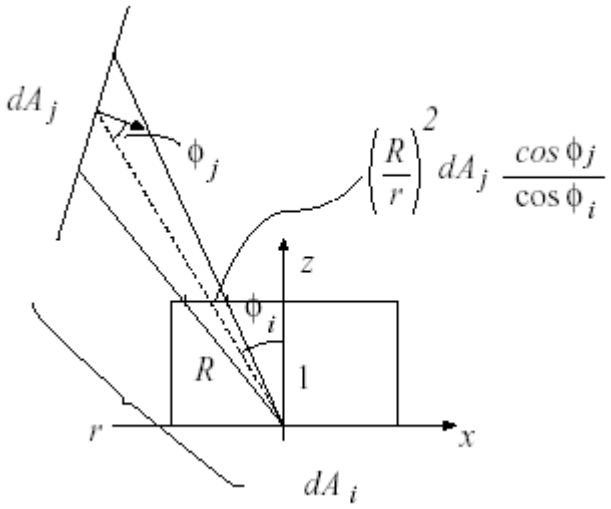
$$\frac{\cos\phi_i \cdot \cos\phi_j}{\pi \cdot r^2} \cdot dA_j \qquad (22)$$

Since the double projection is a one-to-one mapping, if surface ΔAj is above the plane of A_i, the portion, taking the whole ΔAj surface into account, is:

$$\int_{\Delta A_j} H_{ij} \cdot \frac{\cos\phi_i \cdot \cos\phi_j}{\pi \cdot r^2} \cdot dA_j = d_i F_{ij} \qquad (23)$$

This is exactly the formula of an inner form factor integral.

HEMICUBE ALGORITHM



The hemicube algorithm is based on the fact that it is easier to project onto a planar rectangle than onto a spherical surface. Due to the change of the underlying geometry, the double projection cannot be expected to provide the same results as for a hemisphere, so in order to evaluate the inner form factor integral some corrections must be made during the calculation. These correction parameters are generated by comparing the needed terms and the terms resulting from the hemicube projections.

Let us examine the projection onto the top of the hemicube.

Using geometric arguments and the notations of the above figure, the projected area of a differential patch dA_j is:

$$T(dA_j) = H_{ij} \cdot \left(\frac{R}{r}\right)^2 \cdot dA_j \cdot \frac{\cos\phi_j}{\cos\phi_i} = H_{ij} \cdot \frac{dA_j \cdot \cos\phi_i \cdot \cos\phi_j}{\pi \cdot r^2} \cdot \frac{\pi}{\cos^4\phi_i} \qquad (24)$$

since $R = 1/\cos\phi_i$.

Looking at the form factor formula, we notice that a weighted area is to be calculated, where the weight function compensates for the unexpected $\pi/\cos^4\phi_i$ term. Introducing the compensating function ω_z valid on the top of the hemicube, and expressing it by geometric considerations of the figure which supposes an *(x; y; z)* coordinate system attached to the dA_i, with axes parallel with the sides of the hemicube, we get:

$$\omega_z(x,y) = \frac{\cos^4\phi_i}{\pi} = \frac{1}{\pi\left(x^2 + y^2 + 1\right)^2} \qquad (25)$$

Similar considerations can lead to the calculation of the correction terms of the projection on the side faces of the hemicube:

If the side face is perpendicular to the *y* axis, then:

$$\omega_y(x,z) = \frac{\cos^4\phi_i}{\pi} = \frac{z}{\pi\left(x^2 + z^2 + 1\right)^2} \qquad (26)$$

or if the side face is perpendicular to the *x* axis:

$$\omega_x(y,z) = \frac{\cos^4\phi_i}{\pi} = \frac{z}{\pi\left(y^2 + z^2 + 1\right)^2} \qquad (27)$$

The weighted area defining the inner form factor is an area integral of a weight function. If ΔAj has a projection onto the top of the hemicube only, then:

$$d_i F_{ij} = \int_{\Delta Aj} T(dA_j) \cdot \frac{\cos^4\phi_i}{\pi} \qquad (28)$$

Instead of integrating over ΔAj, the same integral can also be calculated on the top of the hemicube in an *x; y; z* coordinate system:

$$d_i F_{ij}^{top} = \int_{T(\Delta Aj)} H_{ij}(x,y) \cdot \frac{1}{\pi\left(x^2 + y^2 + 1\right)^2} \cdot dxdy \qquad (29)$$

since $\cos\phi_i = 1\Big/\left(x^2 + y^2 + 1\right)^{\frac{1}{2}}$. Indicator $H_{ij}(x; y)$ shows whether ΔAj is really visible through hemicube point *(x; y; 1)* from ΔA_i or if it is obscured.

This integral is approximated by a finite sum having generated a $P \times P$ raster mesh on the top of the hemicube. Typical values of P are within the range 50- 200.

$$d_i F_{ij}^{top} = \int_{T(\Delta Aj)} H_{ij}(x,y) \cdot \frac{1}{\pi\left(x^2 + y^2 + 1\right)^2} \cdot dxdy \approx \sum_{x=-P/2}^{P/2-1} \sum_{y=-P/2}^{P/2-1} H_{ij}(X,Y) \cdot \omega_z(X,Y) \cdot \frac{1}{P^2} \quad (30)$$

The only unknown term here is H_{ij} , which tells us whether or not surface j is visible through the raster cell called "pixel" $(X;Y)$. Thanks to the research that has been carried out into hidden surface problems there are many effective algorithms available which can also be used here. An obvious solution is the application of simple ray tracing. The center of dA_i and the pixel defines a ray which may intersect several other surfaces. If the closest intersection is on the surface j, then $H_{ij}(X;Y)$ is 1, otherwise it is 0.

A faster solution is provided by the z-buffer method, since it's supported by video display hardware.

REFERENCES

1. L. Szirmay-Kalos (editor). *Theory of Three Dimensional Computer Graphics*. Akademia Kiado, Budapest,1995.

2. Michael Cohen and Donald Greenberg. *The hemi-cube, a radiosity solution for complex environments*. In Proceedings of SIGGRAPH '85, pages 31-40, 1985.

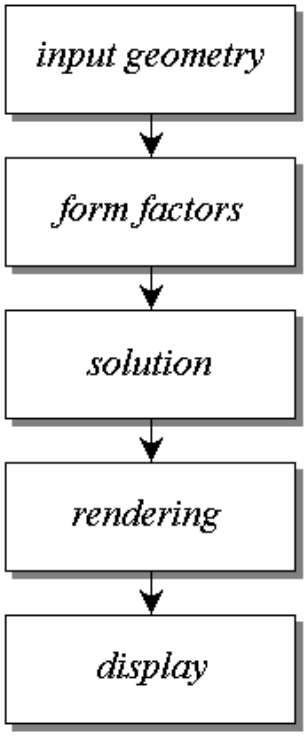
3. Ville kaksonen.*Radiosity*. Telecommunications Software and Multimedia Laboratory, Helsinki University of Technology.

4. Hugo Elias. *Radiosity* (online).Available at: http://freespace.virgin.net/hugo.elias/radiosity/radiosity.htm.

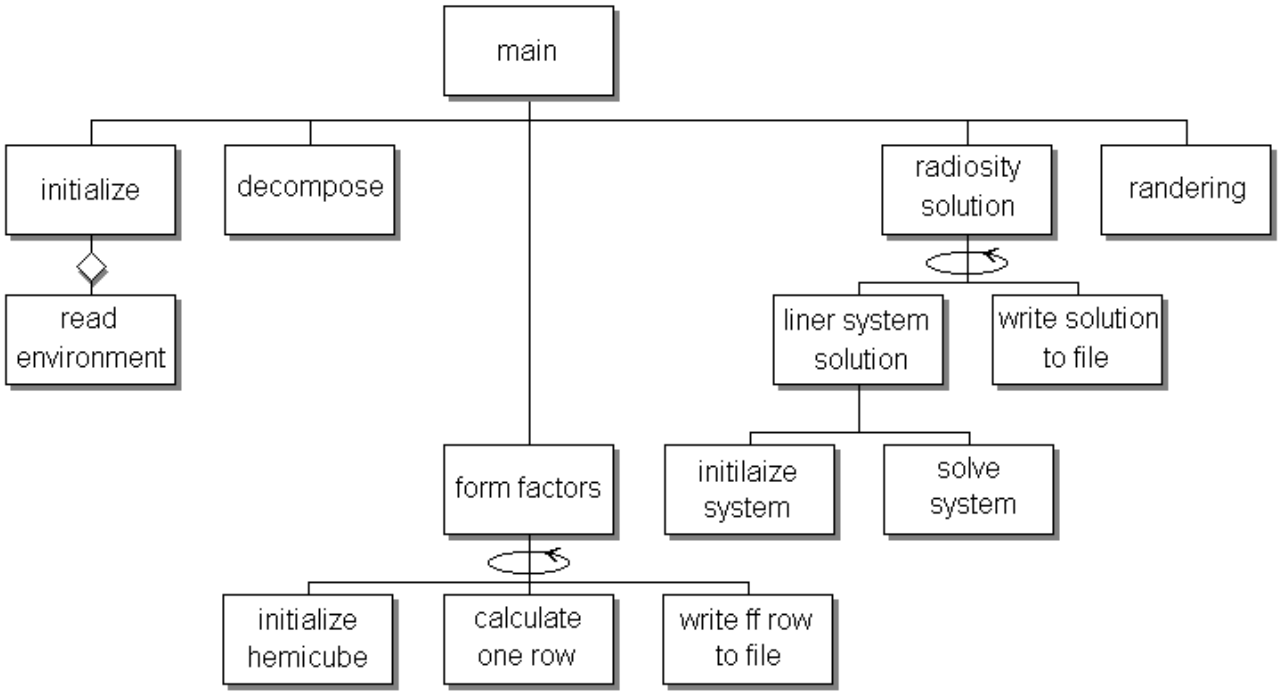
IMPLEMENTATION

OUTLINE

1. Input:
   The environment definition is input from a file containing polygon vertex coordinates, and associated reflectivity and emission values for each color band and a parameter for subdividing the polygon into patches.
2. Form-factors:
   After decomposing the environment into finite patches, the hemi-cube is set around the center of each patch with it's orientation, and a row of form-factors is calculated from that patch to all others through their projections onto the cube. A file containing the matrix of form-factors is output.
3. Radiosity-solution:
   For each color band a linear system of simultaneous equations is constructed using the form-factors and the corresponding set of reflectivity and emission values. The system is solved for patch radiosities. A file containing the radiosities for each color band is output.
4. Rendering:
   An eye position, view direction, and a perspective view frustum angle are specified from which an image is rendered.



CURRENT IMPLEMENTATION'S STRUCTURE CHART



MAIN ALGORITHMS

1. Environment decomposition

Given $0 \le u \le u_{\max}$ and $0 \le v \le v_{\max}$:

```
DecomposeQuad (n, m)
      S = {} ; // the resulting set of quadrilateral patches
      u[ i ] = 0 ;
      for i = 1 to n do
            u[ i + 1 ] = uMax * i / n ;
            v[ j ] = 0 ;
            for j = 1 to m do
                  v[ j + 1 ] = vMax * j / m ;
                  add the quadrilateral { ( u[ i ], v[ j ] ),
                                          ( u[ i + 1 ], v[ j ] ),
                                          ( u[ i + 1 ], v[ j + 1 ] ),
                                          ( u[ i ], v[ j + 1 ] ) } to S ;
                  v[ j ] = v[ j + 1 ] ;
            endfor
            u[ i ] = u[ i + 1 ] ;
      endfor
      return S ;
end
```

for mathematical details on surface parameters, see the appendices.

2. Form factor calculation

```
for i = 1 to patchCount do
      for j = 1 to patchCount do F[ i ][ j ] = 0;
for i = 1 to patchCount do
      place hemicube on patch i ;
      project environment onto hemicube ;
      for k = 0 to pixelCount do
            if ( pixel[ k ] > 0 ) then
                  F[ i ][ pixel[ k ] ] += deltaFormFactor [ k ] ;
      endfor
endfor
```

3. Radosity solution

Using Gauss-Seidel iterative approach, after guaranteeing a convergent solution process

```
while ( not converged ) do
      limit = smallest floating point value ;
      for i = 1 to patchCount do
            previous = radiosity [ i ] ;
            radiosity[ i ] = emission[ i ] ;
            for j = 1 to patchCount do
                  radiosity[ i ] += coefficient[ i ][ j ] * radiosity[ j ] ;
            limit = max ( (radiosity[i] - previous)/radiosity[i]), limit ) ;
      endfor
      if ( limit < epsilon )
            solution converged ;
endwhile
```

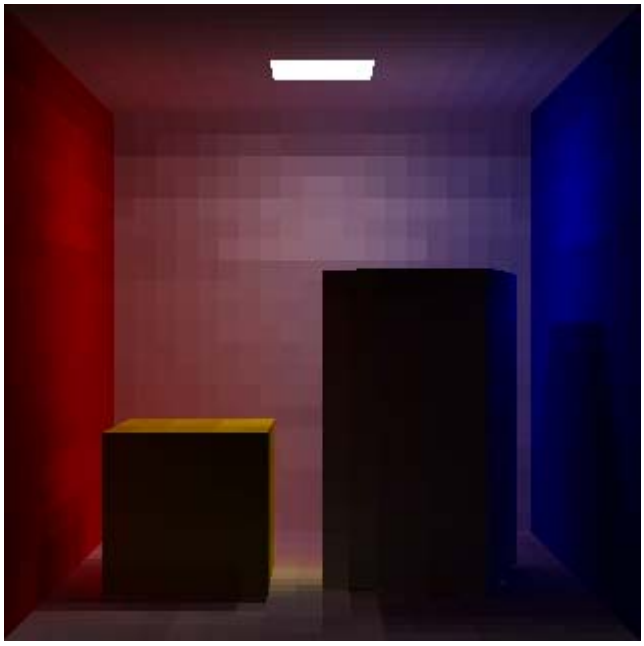
for mathematical details on linear system solution procedures, see the appendices.

RESULTS

1. Cornell box

Cornell box – vertex radiosities

2520 patches in the environment
Hemicube dimensions 100x100 pixels
Form factor calculation : 570 seconds
Radiosity solution : 22 seconds



Cornell box – patch radiosities

Computation was done on an Intel Pentium III at 500 MHz machine.
Images were displayed on a 800x600 framebuffer with 32-bit color depth.

Smooth rendering is accomplished by transferring radiosity values from patches to vertices. Vertex radiosity values are calculated by means of averaging adjacent patch radiosities for a shared vertex.

2. Dining room

Dining room – angle*1*

13218 patches in the environment
Hemicube dimensions 100x100 pixels
Form factor calculation : 13200 seconds

Dining room – angle*2*

Computation was done on an Intel Pentium IV at 1.7 GHz machine.
Images were displayed on a 1024x768 framebuffer with 16-bit color depth.

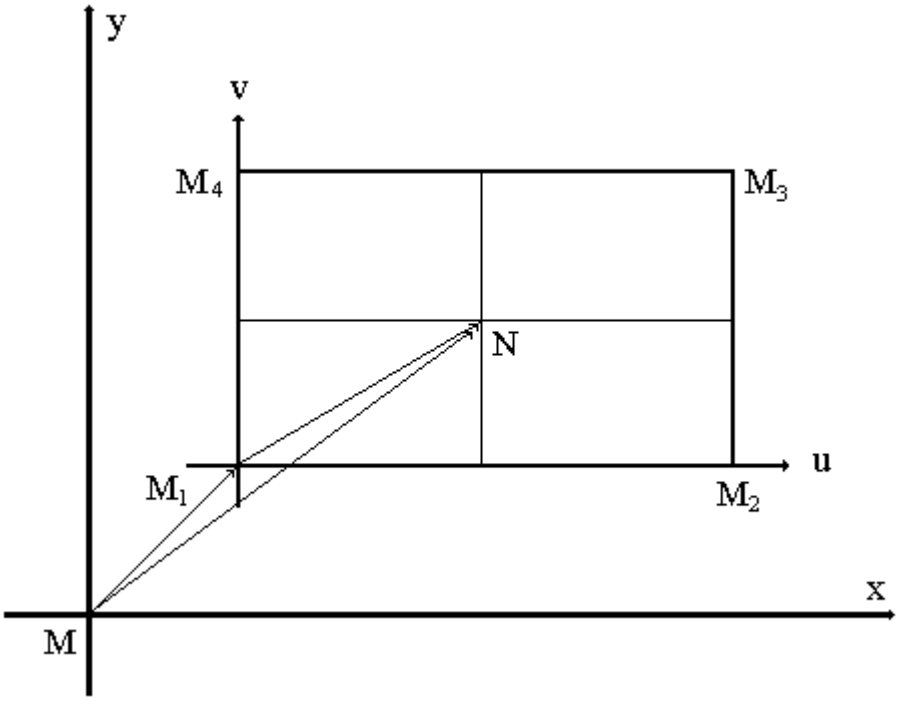
IMPLEMENTATION LIMITATIONS

Considering performance and time frame restrictions, current implementation of radiosity rendering system is only able to handle convex quadrilaterals as it's input.

APPENDIX

SURFACE PARAMETERS

1. Cartesian to parametric conversion:

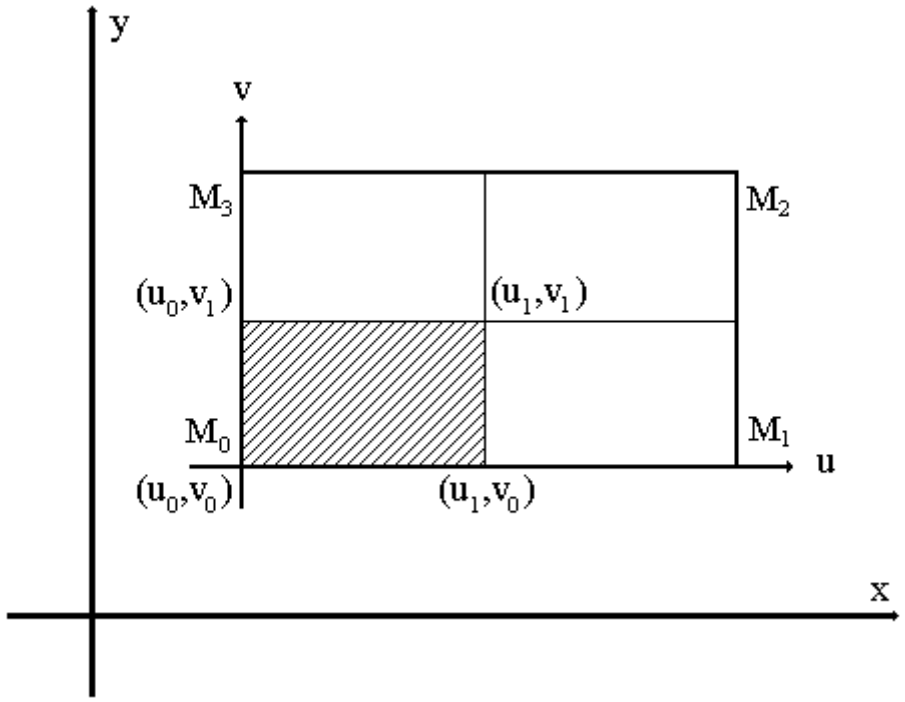


$$\begin{cases} u = f(x,y,z) \\ v = f(x,y,z) \end{cases}$$

$$u = \frac{\overrightarrow{M_1N} \cdot \overrightarrow{M_1M_2}}{\|M_1M_2\|} = \frac{[(x-x_1)(x_2-x_1)+(y-y_1)(y_2-y_1)+(z-z_1)(z_2-z_1)]}{\sqrt{(x_2-x_1)^2+(y_2-y_1)^2+(z_2-z_1)^2}}$$

$$v = \frac{\overrightarrow{M_1N} \cdot \overrightarrow{M_1M_4}}{\|M_1M_4\|} = \frac{[(x-x_1)(x_4-x_1)+(y-y_1)(y_4-y_1)+(z-z_1)(z_4-z_1)]}{\sqrt{(x_4-x_1)^2+(y_4-y_1)^2+(z_4-z_1)^2}}$$

2. Parametric to Cartesian conversion:



$$\begin{cases} x = f(u,v) \\ y = f(u,v) \\ z = f(u,v) \end{cases}$$

(u_0, v_0)	(u_1, v_0)	(u_0, v_1)	(u_1, v_1)
$x = x_0$	$x = x_0 + x_u$	$x = x_0 + x_v$	$x = x_0 + x_u + x_v$
$y = y_0$	$y = y_0 + y_u$	$y = y_0 + y_v$	$y = y_0 + y_u + y_v$
$z = z_0$	$z = z_0 + z_u$	$z = z_0 + z_v$	$z = z_0 + z_u + z_v$

on $\overrightarrow{u_0u}$*:*

$$\frac{(x_{M_1} - x_{M_0})}{x_u} = \frac{\|M_0M_1\|}{u}$$

$$\frac{(y_{M_1} - y_{M_0})}{y_u} = \frac{\|M_0M_1\|}{u}$$

$$\frac{(z_{M_1} - z_{M_0})}{z_u} = \frac{\|M_0M_1\|}{u}$$

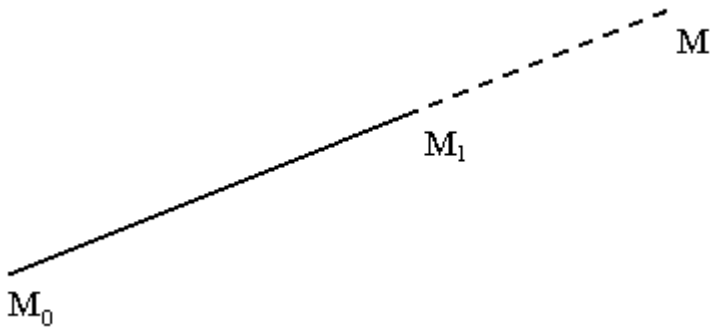
on $\overrightarrow{v_0v}$*:*

$$\frac{(x_{M_4} - x_{M_0})}{x_v} = \frac{\|M_0M_4\|}{v}$$

$$\frac{(y_{M_4} - y_{M_0})}{y_v} = \frac{\|M_0M_4\|}{v}$$

$$\frac{(z_{M_4} - z_{M_0})}{z_v} = \frac{\|M_0M_4\|}{v}$$

VECTOR-PLANE INTERSECTION



$$\overrightarrow{M_0M_1} = (x_1 - x_0, y_1 - y_0, z_1 - z_0)$$
$$\overrightarrow{M_0M} = (x - x_0, y - y_0, z - z_0)$$

It's clear that
$$\overrightarrow{M_0M} = t \cdot \overrightarrow{M_0M_1}$$
therefore :
$$x - x_0 = t \cdot (x_1 - x_0) \Rightarrow x = x_0 + t \cdot (x_1 - x_0)$$
$$y - y_0 = t \cdot (y_1 - y_0) \Rightarrow y = y_0 + t \cdot (y_1 - y_0) \qquad *$$
$$z - z_0 = t \cdot (z_1 - z_0) \Rightarrow z = z_0 + t \cdot (z_1 - z_0)$$

Cartesian plane equation:
$$Ax + By + Cz + D = 0$$
*substituting from * we get:*
$$A(x_0 + t \cdot (x_1 - x_0)) + B(y_0 + t \cdot (y_1 - y_0)) + C(z_0 + t \cdot (z_1 - z_0)) + D = 0$$
$$t = -\frac{(Ax_0 + By_0 + Cz_0 + D)}{(A(x_1 - x_0) + B(y_1 - y_0) + C(z_1 - z_0))}$$

A vanishing denominator indicates a no intersection case
*Substituting t's value in equations * we get the Cartesian coordinates of the intersection point*

LINEAR SYSTEM SOLUTION

Gauss - Jakobi iteration

$$A \cdot X = B$$

In detail:

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1$$
$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2$$
$$\vdots$$
$$a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n$$

Can be written in the form:

$$x_1 = \beta_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n$$
$$x_2 = \beta_2 + a_{21}x_1 + a_{23}x_3 + \cdots + a_{2n}x_n$$
$$\vdots$$
$$x_n = \beta_n + a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn-1}x_{n-1}$$

where:

$$\beta_1 = \frac{b_i}{a_{ii}}$$

$$a_{ij} = \begin{cases} -\dfrac{a_{ij}}{a_{ii}}; j \neq i \\ 0; j = i \end{cases}$$

$$i = 1,2,\cdots,n$$

$$j = 1,2,\cdots,n$$

Assuming that the initial solution $x^{(0)}$ equals zero for all unknowns, we get the next solution:

$$x_1^{(1)} = \beta_1, x_2^{(1)} = \beta_2, \cdots, x_n^{(1)} = \beta_n$$

Generally, solution (k + 1) is obtained from:

$$x_i^{(k+1)} = \beta_i + \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij}x_j^{(k)}$$

$$i = 1,2,\cdots,n$$

$$k = 1,2,\cdots,n$$

Loop count is either predefined, or it's entered and stopped when the following condition holds for any small positive ε

$$MAX_{1 \leq i \leq n} \frac{\left|x_i^{(k+1)} - x_i^{(k)}\right|}{\left|x_i^{(k+1)}\right|} < \varepsilon$$

To apply Gauss-Jakobi method efficiently, the given system should fulfill convergence criterion:

$$\sum_{j=1}^{n} \left|a_{ij}\right| < 1$$

$$i = 1,2,\cdots,n$$

generally the looping process is guaranteed to be convergent when the following inequalities fulfill

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} \left|a_{ij}\right| < \left|a_{ii}\right|$$

$$i = 1,2,\cdots,n$$

Gauss - Seidel approach

Gauss-Seidel iteration is considered to achieve better estimation, and is centered around the following idea:

The calculation of the (k+1) order solution for the unknown x_n is being done by the use of the same order (k+1) solutions for the unknowns $x_1, x_2, \ldots, x_{i-1}$, and by the use of the previous solutions (of order k) for the unknowns $x_{i+1}, \ldots, x_n$

Thus:

$$x_i^{(k+1)} = \beta_i + \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} + \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}$$

$$i = 1,2,\cdots,n$$

$$k = 1,2,\cdots,n$$

The initial solution is the zero vector, and the above convergence conditions apply.