Realistic Image Sythesis:

The goal with the radiosity project is to solve the problem of realistic image synthesis.

The radiosity method is a global illumination method to solve the realistic image synthesis problem.

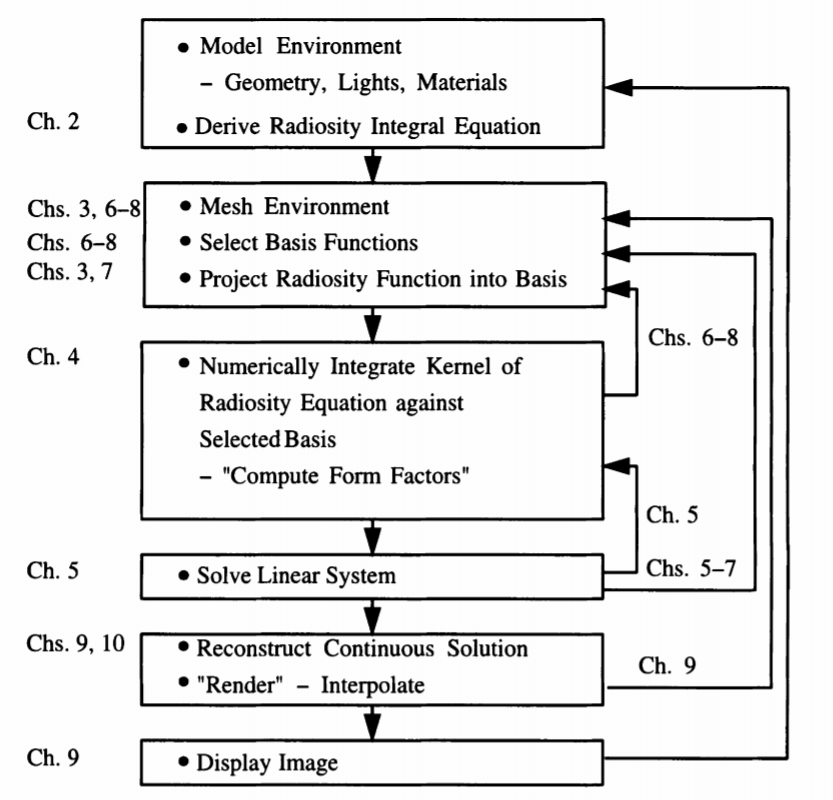
The model of radiosity is based on the physical quantity of the same name:

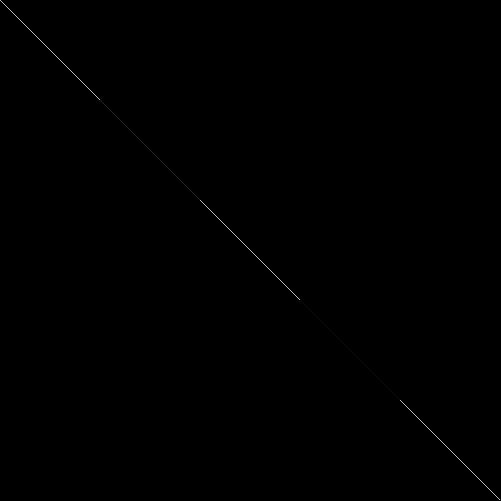
The model is designed under the assumption that all surfaces (and light sources) exhibit Lambertian diffuse reflection and emission.

The radiosity function is piecewise smooth, that is, it is continuous in all derivatives withing regions bounded by discontinuities.

The radiosity model is a Fredholm Integral Equation of the second kind, to solve it we could use analytical methods as the adomian decomposition method, but I needed a numerical method so to make numerical analysis of the results. That is why I chose the finite element method.

The basis for the finite Element Method is to discretize our continues equation [1]. Into a set of finite elements. So we make a transformation of th

The radiosity method has the next steps, to compute a global illumination solution:

This is the matrix P, with reflextivities along the diagonal:

NOTE: that the emissor element, which is the last row has reflectivity equal to zero. This matrix as K has n = 501.

The Form-Factor

The form factor specifies the fraction of the energy leaving one surface which lands in another.

By definition the sum of all the form-factors from a particular point or patch is equal to unity.

For non-occluded environments the form factor for one differential area to another is given by:

By integrating over area j, the form factor from a finite area (or patch) to a differential area can be expressed:

The form factor between finite surfaces (patches) is defined as the area average and is thus:

This expression for the form-factor does not account for the possibility of occluding objects hiding all or part from one patch from another. The complete form is then:

The function takes one If differential area I can see differential area j and zero otherwise.

This double area integral is difficult to solve analytically, so I used the Hemi-Cube algorithm to compute form factors between surfaces (patches).

The Hemi-Cube Algorithm

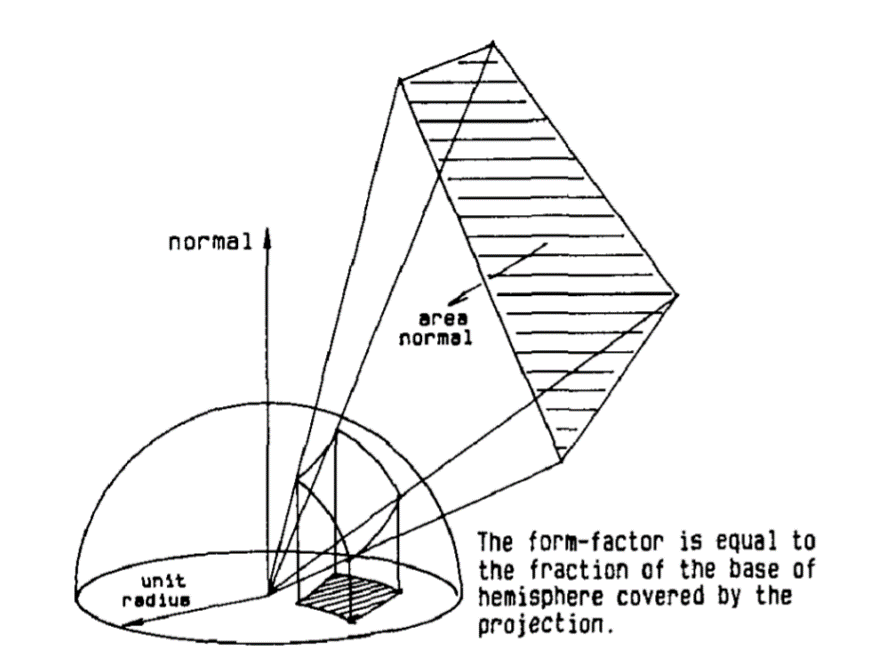
If the distance between two patches is large compared to their size, and they are not partially occluded from one another, it can be seen that integrand from the inner integral remains almost constant. In that case the effect of the outer integral is simply a multiplication by one, and finding a solution to the inner integral will provide a good approximation for the form factor.

The form factor is then calculated using equation “TODO add equation indexing” by using the center point of patch I to represent the average position of patch i.

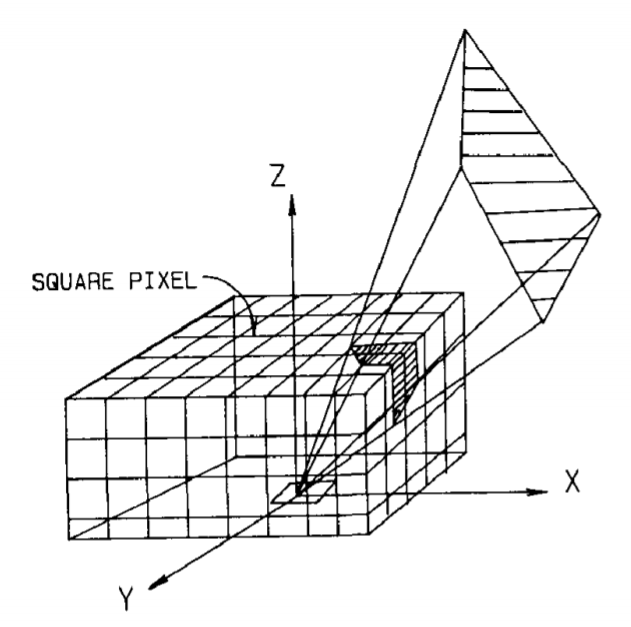
Each patch has as its view of the environment, the hemisphere surrounding its normal.

For a finite area, the form-factor is equivalent to the fraction of the circle (which is the base of the hemisphere) covered by projecting the area onto the hemisphere and then orthographically down onto the circle.

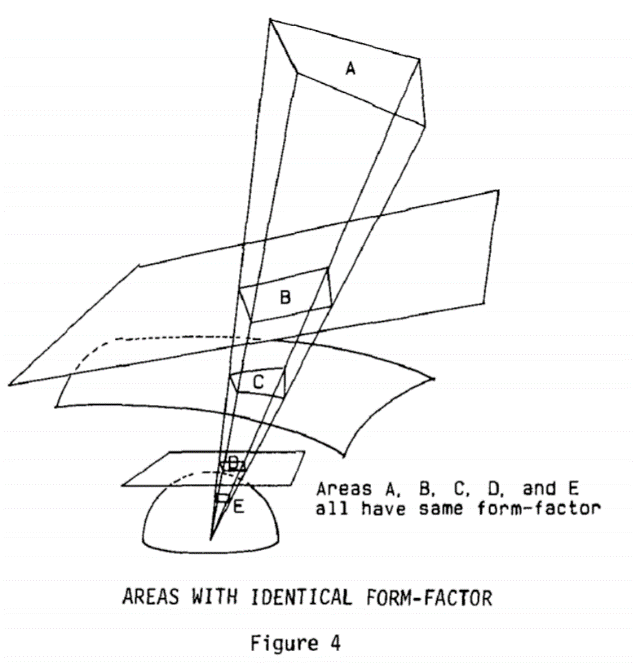
The form-factor is equal to the fraction of the base of hemisphere covered by the projection.



This hemisphere can be broken into small delta solid angles, difficulty on creating equal sized areas on a sphere as well as creating a set of linear coordinates to uniquely describe locations on its surfaces makes projecting all the surfaces on the environment onto hemisphere impractical, instead we can project onto an hemi-cube of half-length equal to one.



Any two patches that project to the same delta area in the hemicube will have the same form factor as described in the next image:



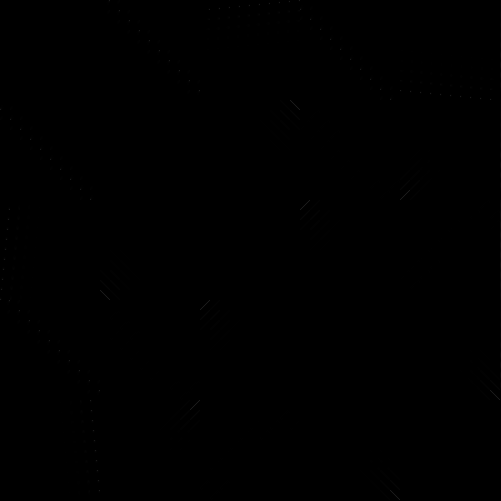
The hemicube contains 5 faces, one full face and 4 half faces. These faces are divided into pixels at a given resolution (I used 100 by 100 pixels per face). And the environment is projected onto the 5 planar surfaces.

If two patches project onto the same pixel on the cube, a depth determination is made as to which patch is seen from the surface with the hemi-cube.

The contribution of each pixel on the cube’s surface to the form-factor value varies and is dependent on the pixel location and orientation. A specific delta form factor value is found from equation 3 for the differential area to differential area formulation and added to the respective Form Factor matrix entry.

This is the resulting Form Factor Matrix computed in radiosity code. Each row i represents an element and each column represents the geometric relationship each element j has with element i.

It is computed using the Hemi-Cube algorithm. It is possible that the values cannot be seen because they are relatively low values. But that is good because the sum of each row cannot be greater than one, if not the system does not converge.

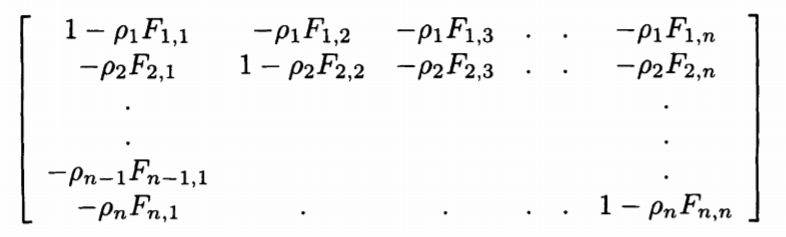


The Energy Balance solution

Now having the form-factor matrix we can solve the system of linear equations:

Where K is defined to be:

M is the identity matrix; P is a diagonal matrix with reflectivities on its diagonal and F is the matrix of form factors. Thus, K looks like:



Using the form factor matrix computed before we can do matrix operations to compute which leads to this matrix:



The matrix K is diagonally dominant; thus, it is well suited to iterative methods as Gauss-Seidel, which is the method I use to solve the system of linear equations.

The Gauss-Seidel Method is really fast with K matrix (only one iteration). Now to make it converge the spectral radius of K must be less than 1. Therefore, the largest absolute value for an eigenvalue of K must be less than one.

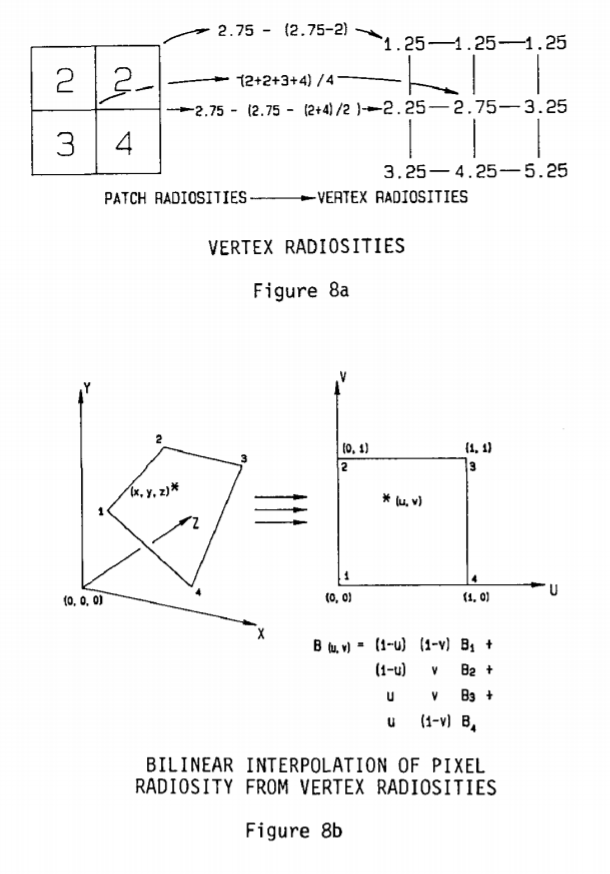
Stimuli-Dependent

There is a system of linear equations per stimuli required, so for the radiosity code I solved three systems of linear equations (red, green and blue).

Rendering

To render an image the discretized radiosity information is

This bilinear interpolation of radiosities insures first order continuity at patch edges. In order to perform the interpolation, radiosity values must be transferred from the patches themselves to each vertex of the patches.



The radiosity solution is view-independent, therefore changing the view position does not need a new computation of radiosities nor form factors.

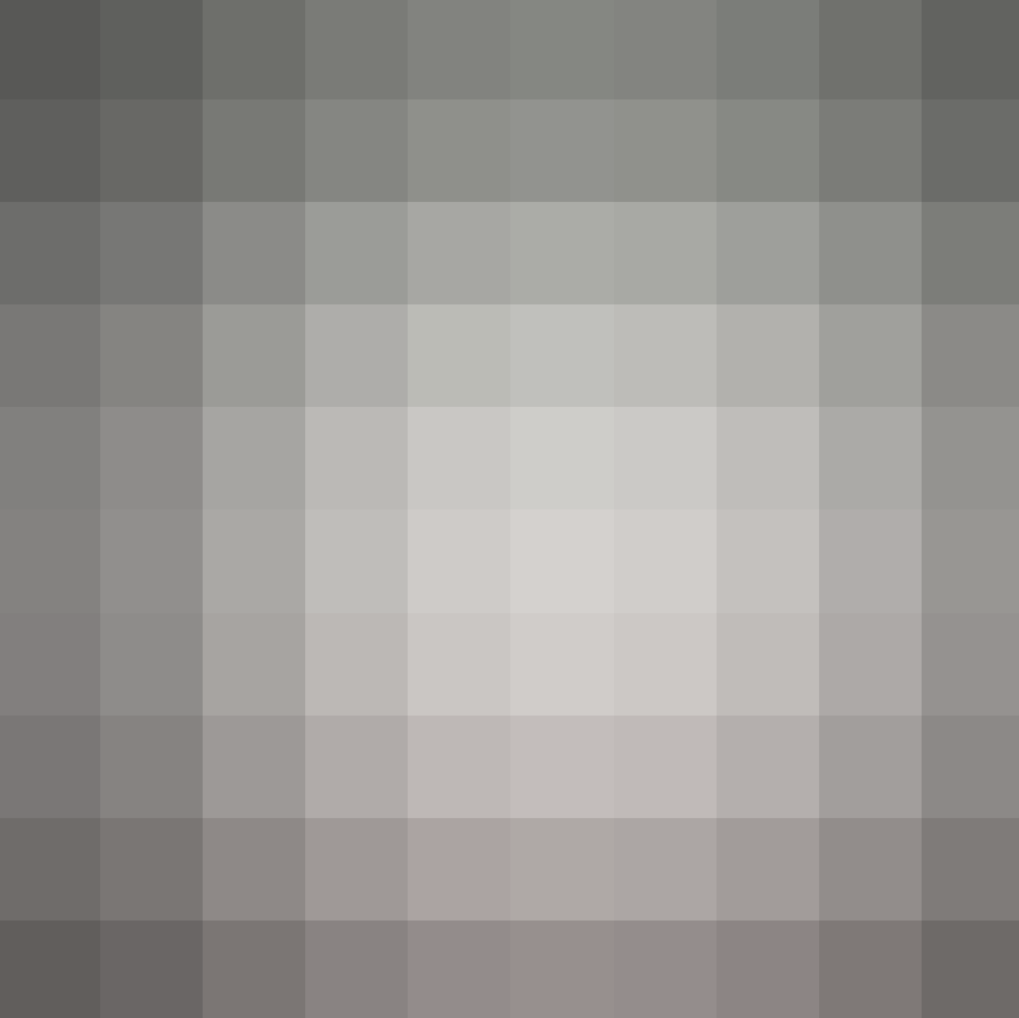
These are vertex colors for face xy with z = 0 without interpolation.



Now with interpolation:



And this is face xz with y equal 0:



Now with interpolation: