Interactive Computer Graphics

Lecture 11: Radiosity - Principles

The reflectance equation

Earlier in the course we introduced the reflectance equation for modelling light reflected from surfaces:

$$I_{reflected} = k_a + I_i k_d \, \boldsymbol{n.s} + I_i K_s (\boldsymbol{r.v})^t$$

Where I is the incident light and the constants represent: k_a the amount of ambient light

k_d the amount of diffuse reflection

k_s the amount of specular reflection

Lighting model for ray tracing

When we used ray tracing we assumed that there was a small number of point light sources.

However, according to the reflectance equation, every surface is reflecting light, and so should also be considered a light source.

So rather than use a constant for ambient light, should we not sum the light received from all other surfaces in the scene?

Ambient light

A better approximation to the reflectance equation is to make the ambient light term a function of the incident light as well:

$$I_{reflected} = I_i k_a + I_i k_d \, \boldsymbol{n.s} + I_i K_s (\boldsymbol{r.v})^t$$

or more simply to write (for a given viewpoint)

$$I_{reflected} = R \, I_i$$

where R is the reflectance function.

Radiosity

Radiosity is defined as the energy per unit area leaving a surface. It is the sum of the emitted energy (if any) and the reflected energy.

For a small area of the surface dA (where the emitted energy can be regarded as constant) we have:

$$B \, dA = E \, dA + R \, I$$

Notice that we now treat light sources as distributed

Collecting energy

The incident energy can now be written (for patch i) as:

$$I_i = \int B_j F_{ij} \, dA_j$$

where the integral is taken over all patches except i

F_{ij} is a constant that links patch i and patch j called the form factor

Discrete formulation

For computer graphics we cannot expect to compute a continuous solution, so we divide all polygons up into patches and replace the integral with a sum:

$$B_i = E_i + R_i \Sigma B_j F_{ij}$$

Where the sum is taken over all patches except i (or alternatively we can set $F_{ii} = 0$)

In matrix form

$$\begin{pmatrix} 1 & -R_1F_{12} & -R_1F_{13} & . & . & -R_1F_{1n} \\ -R_2F_{21} & 1 & -R_2F_{23} & . & . & -R_2F_{2n} \\ -R_3F_{31} & -R_3F_{32} & 1 & . & . & -R_3F_{3n} \\ . \ . & . \ . & . \ . & . & . & . \ . \\ -R_nF_{n1} & -R_nF_{n2} & -R_nF_{n3} & . & . & 1 \end{pmatrix} \begin{pmatrix} B_1 \\ B_2 \\ B_3 \\ . \ . \\ B_n \end{pmatrix} = \begin{pmatrix} E_1 \\ E_2 \\ E_3 \\ . \ . \\ E_n \end{pmatrix}$$

If we can solve this for all Bi then we will be able to render each patch with a correct light model.

However, this is not so easy to do since

- the form factors are not known.
- the reflectance equation is insoluable
- the matrix is big - circa 10000 by 10000

Wavelengths

The radiosity values are wavelength dependent, hence we will need to compute a radiosity value for RG and B.

Each patch will require a separate set of paramaters for R,G and B.

The three radiosity values are the values that the rendered pixels will receive.

Back to the reflectance function

$$I_{reflected} = I_i k_a + I_i k_d \, \boldsymbol{n.s} + I_i K_s (\boldsymbol{r.v})^t$$

Note that the specular term depends on the vector to the light source ***v***.

But now, every patch is a light source!

Specular reflections

Moreover our light sources are no longer points, so we need to collect the incident light in a specular cone to determine the specular reflection.

This is computationally infeasible.

We will return to specularities later, but for the moment we will consider only diffuse radiosity.

Patching Problems

We need to divide our graphics scene into patches for computing the radiosity.

For small polygons we can perhaps use the polygon map, but for large polygons we need to subdivide them.

Since the emitted light will not be constant across a large polygon we will see the subdivisions

Large Polygons

Each Patch will have a different but constant illumination.

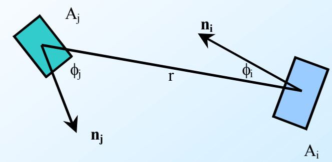
Thus we will see the patches unless either:

Patches project to (sub) pixel size

or We smooth the results (eg by interpolation)

Form Factors

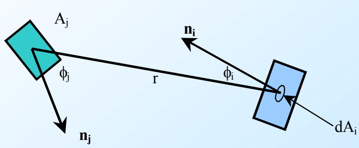
The form factors couple every pair of patches, determining the proportion of radiated energy from one that strikes the other.



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

Simplifying Form Factors

The equation can be simplified if we consider solving just an elemental area dAi of Ai, placed at the centre of Ai. If r is large, the inner integral can be considered constant over dA_i



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

Simplifying form factors

With this assumption the outer integral evaluates to 1, and we assume that the same value persists over the patch.

Hence we can write the integral as:

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

Further simplifying

Having assumed that the radius is large compared with patch Ai, it should not be unreasonable to assume that it is also large with respect to the size of Aj. Hence the integrand of

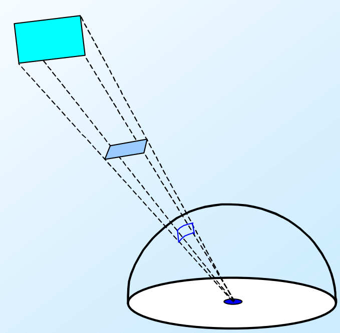
$$F_{ij} = \int_{A_j} \frac{\cos\phi_i \cos\phi_j}{\pi r^2} \, dA_j$$

can similarly be considered constant over Aj

Thus $F_{ij} = \cos\phi_i \cos\phi_j \, Area(A_j) / \pi r^2$

The Hemicube method

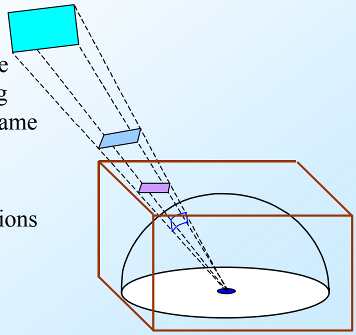
Using a bounding hemisphere it can be shown that all patches that project onto the same area of the hemisphere have the same form factor



The Hemicube method

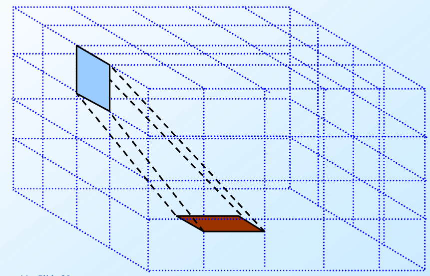
So, all patches that project onto the same area of a surrounding hemicube have the same form factor.

Computing intersections with planes is less computationally demanding



Delta form factors

The hemicube is divided into small “pixel” areas and form factors are computed for each.



Delta form factors

If the area of a hemicube pixel is ΔA, its form factor is:

$$\cos \phi_i \cos \phi_j \, \Delta A / \pi r^2$$

These delta form factors can be computed and stored in a look up table.

Problem Break

Given that the hemicube is defined at the origin, with the z axis vertical what is the delta form factor for the following two hemi cube pixels in the top face, z=1:

{xmin,ymin,xmax,ymax} = {-0.05, -0.05, 0.05,0.05}

and

{xmin,ymin,xmax,ymax} = {-0.45, -0.05, -0.55,0.05}

Solution

For the top face $\cos \phi_i = \cos \phi_j = 1/r$

(The top face is the plane z=1)

Thus, $\cos \phi_i \cos \phi_j \, \Delta A / \pi r^2 = \Delta A/\pi r^4$

ΔA in our example is 0.01 giving $0.01/\pi r^4$

Case 1, r=1, form factor is $0.01/\pi = 0.00318$

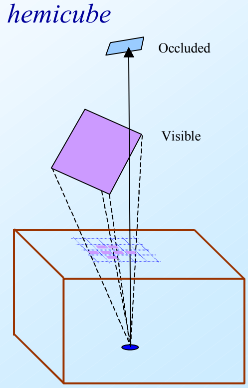
Case 2, $r=\sqrt{1.25}$, form factor is $0.01/1.56\pi = 0.002$

Projection of patches onto the hemicube

We now need to know which patch is visible from each hemicube pixel.

This could be done by ray tracing, or projection.

Further patches are rejected, solving the occlusion problem.



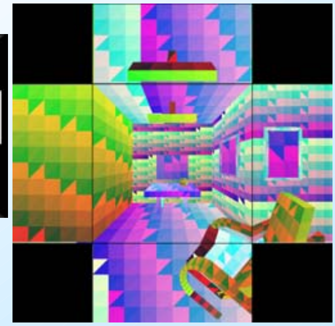
Sum the pixels per patch

Notice that all we need to determine is which patches are visible at each hemicube pixel.

Once this is found we calculate the form factor for each patch by summing the delta form factors of the hemicube pixels to which it projects.

Inside the hemicube

Images from Alan Watt: The Computer Image



In summary

1. Divide the graphics world into discrete patches

2. Compute form factors by the hemicube method

3. Solve the matrix equation for the radiosity of each patch.

4. Average the radiosity values at the corners of each patch

5a. Compute a texture map of each point on the patch

5b. Project to the viewing window and render with interpolation shading.

Radiosity Images

Much of the early work on radiosity was carried out at Cornell University, and images and tutorial material can be found on their web site.

http://www.graphics.cornell.edu/online/research/