Chapter 5

Radiosity Rendering

References

As you read the following, you may find the following summary helpful. In particular it contains some nice illustrations.

http://www.siggraph.org/education/materials/HyperGraph/radiosity/radiosity.htm

Here is an excellent site for understanding hemicubes:

http://freespace.virgin.net/hugo.elias/radiosity/radiosity.htm

The first paper below is the method described in this Chapter.

"The Hemi-cube - a radiosity solution for complex environments"
Michael F Cohen and Donald P Greenberg
ACM SIGGRAPH Computer Graphics 1985 Vol 19 Issue 4

The second paper is the progressive method described at the end of this Chapter.

"A progressive refinement approach to fast radiosity image generation"
Michael F Cohen, Shenchang Eric Chen, John R Wallace and Donald P Greenberg
ACM SIGGRAPH Computer Graphics 1988 Vol 22 Issue 4

The third paper is for interest only. It avoids form factors by using ray-tracing to do the same task.

"A Ray Tracing Algorithm for Progressive Radiosity"
John R. Wallace, Kells A. Elmquist, Eric A. Haines
ACM SIGGRAPH 1989.

5.1 Radiosity

Radiosity is a third way of rendering pictures. It is by far the most computationally expensive, but produces arguably the most realistic effects seen to date. It achieves this by being based

on the physics of light energy. It primarily addresses diffuse lighting, so specular effects (if required) have to be handled by another method. It is also a method in which the lighting effects are part of the modelling process: there is little real-time calculation when rendering or viewing the scene.

It models the effect of light energy being bounced from everywhere in the scene to everywhere in the scene. It is therefore said to use a *global lighting model*. In contrast, Gouraud and Phong shading are purely *local lighting models* because they take no account of illumination from other objects. Ray-tracing has a global element because it can take account of one object being reflected in another, such as one with a shiny surface. However it does not model *energy interchange* between all objects so this effect is purely viewpoint-driven. Simple rendering and lighting techniques do not allow for interchange of light energy between surfaces in a scene. Thus an object next to another bright white object should be at least partly illuminated by the white object and vice versa.

5.1.1 What is Involved?

A radiosity computation for a given scene starts from an assumption of *energy equilibrium* (light levels are not changing), it calculates (to some approximation) the energy reaching each surface due to the light it is receiving from every other surface.

This allows for softer lighting effects, like soft-edged shadows and colour bleeding. The latter is where light coming from (say) a red cube on a white table makes an area of the table look pinkish. It also models non-point sources of light (e.g. strip lighting), which ray-tracing finds difficult.

There are two important features of this approach:

- light sources are treated no differently from any other surface (so we can model their shape), except that they have a positive emission;
- the radiosity equations are solved once, before the scene is rendered, and the scene can be viewed from anywhere without recomputation. Thus there is an initial long computation, but after that multiple views can be produced quickly, in fact using traditional Gouraud shading.

Radiosity is usually associated with surfaces, rather than solids, because the surface is where the light energy interchange takes place. Radiosity tries to do properly what "ambient" lighting approximates. It is more accurate to think of the radiosity calculation as a *modelling* process. We are modelling the illumination and including it in our geometric model. Rendering for radiosity is simple Gouraud shading.

5.1.2 Basic Approach

Imagine a closed 3D world, perhaps the interior of a room with furniture and other objects, illuminated by several light panels. The room is in a state of energy equilibrium in the sense that for any surface (other than the light sources) the amount of energy reaching it is equal to the amount absorbed plus the amount re-radiated. This re-radiated light becomes a fraction of the light reaching the other surfaces in the scene. The total amount of light energy in the room must eventually be scattered towards the eye or absorbed: no scattering is perfect.

A given surface i is potentially illuminated by every other surface in the scene. The light leaving surface i is called its radiosity. It is made up of two components: the light reaching it from other surfaces and scattered by it; the light which it generates directly, if it is a light source.

The reflectivity of the surface tells us what proportion of the first component it scatters. If the material is a purely diffuse reflector, it is what we informally think of as "the colour" of the material itself (i.e. not what it looks like when illuminated but its intrinsic colour).

Calculating the first component is not easy however. It is basically the sum over all other surfaces of the fraction of their radiosity which reaches surface i. The fraction F_{ij} that reaches surface i from each surface j is called the *form factor*. The form factors depend on the geometry of the scene, including the angle and distances between surfaces i and j. If we assume we can work all this out, we can say:

$$Radiosity_i = Emission_i + Reflectivity_i \int Radiosity_j Form Factor_{ij}$$

where we mean integrate over all other surfaces j. Strictly, that means over every part of every surface: the radiosity will not in general be constant over the surface. It won't be practical to calculate it at "every" mathematical point. For a practical solution, we will have to decide at what approximation we want to calculate the energy distribution. Since a given surface such as a wall may be quite large, it is usual to divide each surface into *patches*. The patches are the unit whose energy level we model. They are assumed to be *energetically uniform*. Then we can put a triangular mesh through the centres of the patches – where we have calculated the radiosity values – and then Gouraud-shade the triangles at render time, to give a continuously varying visual effect. In other words, we get a good estimate of radiosity at the centre of the patch, then linearly interpolate within the triangles. Doing this allows us to model all the soft-edged lighting effects we get in a real room. One advantage of this method is we can have lights of any shape built from patches, not just point sources. Lights are not treated differently from other patches: any patch can scatter and emit light.

However the radiosity of any patch depends on the radiosity of every other patch, as the equation above shows. We have to regard this interchange of energy as a (large) set of simultaneous energy equations, which we solve to get the individual illuminations for each patch.

For efficiency, we use large patches for areas that we expect to have (near) uniform illumination, and smaller patches for more rapidly varying areas. Crucially, we set up the system of equations by determining all the possible interchanges of light between each patch and every other patch and solve them. This light interchange is what radiosity rendering is about. It is costly and slow but takes place at the modelling stage. To render the scene, we just need Gouraud (to blend within and across the edges of the patches), so this part is very fast and easily achieves a real-time walk-through.

5.1.3 Theory

In this section we will work out how to calculate the radiosity at any point on the surface the model. That is, we are not concerned with pixels, triangles or any other practical implementation matter: that will come later.

By definition, *radiosity* is the energy per unit area leaving a surface in unit time.

We are assuming equilibrium, so we can ignore time and consider the instantaneous solution. We will need the following terminology.

- A_i is area of the patch i;
- B_i the radiosity of patch i;
- E_i is the generated energy per unit area per unit time (light source only);
- R_i is the reflection coefficient, the fraction of the incident energy that is reflected

- F_{ij} is the fraction of the energy which leaves patch j and reaches patch i, called the *form factor*

Let us ask what is happening at some patch i. We first ask, how much energy do the other patches radiate towards patch i? Let's consider some other patch j, in particular an infinitesimal area within it. This is emitting $B_j dA_j$ in total. Probably most of this misses patch i but some fraction F_{ij} will hit. That is, the amount which will hit is:

$$F_{ij} B_j dA_j$$

This is the amount arriving from a single infinitesimal area of patch j, so the total reaching patch i from all patches j is:

$$\int F_{ij} B_j dA_j.$$

Some proportion of this is absorbed and the rest is scattered, so the amount scattered from patch i is:

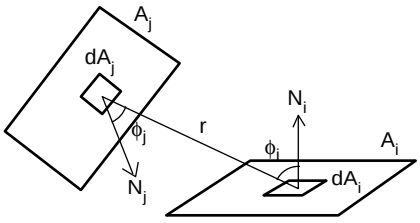
$$R_i \int F_{ij} B_j dA_j$$

Patch i might be a light source, in which case it will also be generating light in its own right. In general then the energy leaving patch i is:

$$B_i dA_i = E_i dA_i + R_i \int F_{ij} B_j dA_j$$

This is the *radiosity equation*. It expresses the complete light interchange between every surface in the model. The non-zero values of E are the lights putting energy into the system. The reflection coefficients R say what is scattered, so $1-R$ is the amount absorbed at each scattering. These two are in balance – the room does not get any brighter are darker – so we have a *steady state* solution.

The only items not known in this equation are the values of the form factors. We need to understand how to calculate them from the geometry of the patches.



We make the assumption that the energy reaching i from j is uniform over the area of i. (This is not strictly true because we are not using infinitesimal patches but, as we said earlier, when we come to render the patch we will be assuming that anyway.) This means we just consider a point on i. We note that i can receive energy from any direction within a hemisphere centred on this point.

From the diagram we can argue that the amount energy leaving a point on patch j *in the direction of* a point on patch i is reduced by the angle subtended to that direction by patch j. Specifically it will be reduced to $cos(\phi_j)$. By a symmetric argument, the amount of this hitting patch i will be reduced by its angle, to $cos(\phi_i)$.

As we are assuming that the patches themselves are small enough to be approximately uniform in energy, then we need do no more than add up all the contributions, each of which will be as just described. This means the overall form factor is:

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

We note that, at equilibrium, $F_{ji}A_i = F_{ij}A_j$. Finally, note there are two cosines here, one associated with each of the two patches. At heart this is Lambert's law used twice. It is used once for the angle of the surface "sending" the radiation (because its angle to its incoming radiation determines how much energy it can scatter) and once for the angle of the surface "receiving" the radiation (because its ability to capture and scatter the energy coming from the first patch depends on its angle to that energy).

5.2 Practical approximations and form factors

If we want to make pictures using this method, then we need to write a program. We have already noted that we cannot calculate radiosity values at "every" point on the surface of the model: infinity is too big a calculation! We make two approximations to the geometry. The first is that we treat the receiving patch as energetically uniform. This means that we can reduce the double integral to a single one. Then we assume (as we indicated earlier) that we will work with finite-size patches instead of infinitesimals. So, we can approximate the radiosity equation integral with the more tractable summation:

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

(You will see that this is the same as the wordy equation we introduced right at the start.) For monochrome images, we need one of these equations for every patch in our model and they must be solved as simultaneous equations. For colour every patch needs three similar equations, one each of red, green, and blue.

Notice how each B_i is dependent on all the other B values. For example, if j were only 3, and we consider just B_1, we have:

$$B_1 = E_1 + R_1(B_1F_{11} + B_2F_{21} + B_3F_{31})$$

Furthermore, if we re-express this for E, we get:

$$B_1(1 - R_1F_{11}) + B_2(-R_1F_{21}) + B_3(-R_1F_{31}) = E_1$$

This gives us the idea of how to write the whole set of simultaneous linear equations as:

$$\begin{pmatrix} 1 - R_1F_{11} & -R_1F_{21} & \cdots & -R_1F_{n1} \\ -R_2F_{12} & 1 - R_2F_{22} & \cdots & -R_2F_{n2} \\ \vdots & \vdots & \vdots & \vdots \\ -R_nF_{1n} & -R_nF_{2n} & \cdots & 1 - R_nF_{nn} \end{pmatrix} \begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_n \end{pmatrix} = \begin{pmatrix} E_1 \\ E_2 \\ \vdots \\ E_n \end{pmatrix}.$$

The E_i are zero except for the light-emitting patches. The non-zero values thus represent the energy entering the system. The R_i are known from the surface characteristics. The F_{ij} must be calculated from the geometry of the scene. As patches are flat, no energy leaving a patch

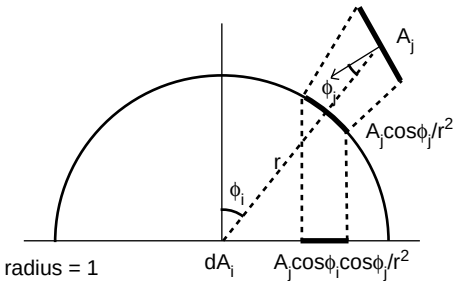
hits itself, so $F_{ii} = 0$. Computing the other F_{ij} is very time intensive. For a practical scene, we usually expect tens of thousands of patches.

To solve the form factor integral (given earlier) is difficult, even for simple surfaces. Nusselt introduced a geometric approach giving the same solution. This is the *hemisphere* method.

5.2.1 The hemisphere method

With the approximations stated earlier, we are effectively treating our chosen patch i as a point, for purposes of gathering the energy coming its way. So one way to calculate the energy arriving from other patches would be to place a unit-radius hemisphere on that point (i.e. somewhere inside the patch), with the hemisphere oriented to face away from the surface. Then we ask what energy passes into the hemisphere: this will be the energy arriving at patch i. If we project the other patches onto this hemisphere, the projections will allow us to compute the form factors for those other patches.

We will need to account for the angle of patch i as well as that of patch j. We first project the shape of patch j onto i's hemisphere, using the centre of i as the centre of projection. The resulting area projected on the hemisphere allows for the orientation of patch j: if it is square-on, we get a bigger result than if it is edge-on. However this makes no allowance for the orientation of patch i (because we projected towards the centre point, which has no direction). Clearly, if this contribution hits patch i square-on, it will add more than if it hits obliquely. We can allow for this by weighting the contribution according to where it is on the hemisphere: high weighting near the centre, low weighting near the edge. So the projected *area* is the incoming energy due to patch j and its orientation; but the *position* on the hemisphere decides how much the orientation of patch i causes it to receive. If we project downwards from the surface of the hemisphere, onto the underlying disc, the projected area will be reduced by the cosine of the orientation, exactly what we want: high weighting near the centre, low weighting near the edge. What we have just described is purely geometric. All we now have to do is recognise that the energy is proportional both to the area we have just calculated and to the inverse square of the distance between the patches. Let's work through this step by step.



In the above, r is distance between dA_i and the centre of A_j; and the hemisphere is of radius 1 and so its circular base has area π.

What we are going to do first is project patch j onto the hemisphere, taking account of the angle of patch j (because if it is edge-on it will contribute nothing). This calculation takes no account of the orientation of patch i, so we will then need to project from the hemisphere onto the disc below it, allowing for the orientation of patch i; again, if this is edge-on to the section of hemisphere, it will receive little. The resulting area, as a fraction of the total disc area π, is proportional to the form factor.

Note: This is little more than the perspective calculation, expressed for areas rather than for coordinate points. We have centrally-projected the patch onto the hemisphere, rather than onto a flat screen. However, if I were to stand at the centre of projection and look outwards, what

I saw on the 2D surface of the hemisphere would be the same as if I looked at the 3D scene instead, which is a characteristic of perspective projection. The inverse square law makes it the correct size at the distance of the hemispherical "screen".

Note: Why are we doing this central projection? We want to estimate the contribution from all the other patches out there, at a point. They are at different distances and at different orientations. By projecting them onto a unit hemisphere, we take out that 3D variability while losing none of the important information about their contributions. This depends only on the solid angle they present: this angle is unchanged by this projection. We have therefore reduced a complicated 3D problem to a simpler 2D one.

If we project area A_j onto the hemisphere, we get $A_j \frac{\cos\phi_j}{r^2}$. The cosine comes from the angle of A_j while the inverse square law for energy expanding from a point source gives the r^2. [Remember it is the *area* we are interested in, not just the length in the illustration.] The cosine tells us that if A_j lies along a radius, it sends no energy to patch i; whereas if it is at right angles to the radius, it sends maximum energy. This is analogous to the argument used in Lambert's law because we are considering the projected size of the patch as a measure of its energy contribution.

Note: For all the detail differences, this is the same argument we used to get the form factor double integral, earlier.

We can now forget the patches themselves and concentrate on the contributions made to patch i by their elements projected on the hemisphere. So we project the element on the hemisphere directly down onto A_i, getting a multiplier of $\cos\phi_i$. This is the calculation which allows for the orientation of patch i, relative to the incoming energy from patch/element j.

We are arguing that the element's contribution is proportional to the area of this second projection. If the element is at right angles to the patch i (i.e. low on the horizon of the hemisphere), then it contributes nothing because the receiving patch is edge on to the incoming radiation. If it is parallel to patch i (i.e. overhead), it contributes its maximum amount. The energy associated with the element has already taken account of the orientation of the patch j, so this second projection is effectively taking account of the orientation of patch i.

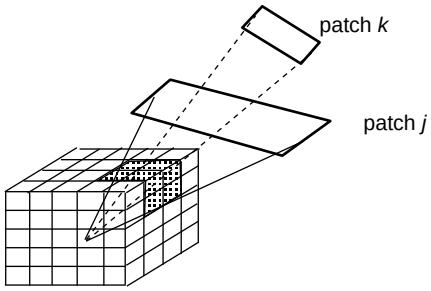
So consider a small area dA_i on patch i, and another patch A_j. We can approximate the form factor by using the geometry in the diagram and then integrating over the hemisphere:

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

This is a single integral, whereas as our theoretical solution is a double integral. This is because we have treated patch i as a point: we assumed it was energetically uniform.

5.2.2 The hemicube method

The integral shown is again a fully continuous solution, so we will estimate it by using a finite number of areas. A numerical method to compute form factors was published in 1985 by Cohen and Greenberg, called the *hemicube method*. What Cohen and Greenberg did was approximate the hemisphere with a hemicube, as follows.



Projecting onto planes is easy. We can also divide the hemicube surface into small cells: the more we use, the better the accuracy. We project the other patches onto the hemicube surfaces: some patches project onto one face of the cube, others onto several faces. This is treated by taking one face of the hemicube at a time and clipping the patches against that face before projecting. Also we can use hidden surface computations (z-buffering) to determine which is the nearest patch when several project onto the same area. Graphics cards support z-buffering, so this part can be fast.

All the hemicube is doing is simplifying the projecting of the *shape* of patch j. We now need to think about the energy it contributes.

When we moved from the hemisphere to the hemicube we lost the ability to weight contribution with angle: the hemicube only has two angles, vertical and horizontal. However, we know which cell on the surface approximates which part of the hemisphere, so we can give each cell a weighting to correct for this. A cell in the centre of a face can be seen directly at the centre of patch i, while one near an edge of the hemicube is seen obliquely (i.e. the usual cosine argument applies). Thus weightings are incorporated to give more importance to central cells. These can be precomputed when generating the hemicube.

Once all of this is done, we can finally compute the radiosity values for each patch. As you can tell, this can take many hours.

The good news is, to view the scene, we choose a viewpoint and simply Gouraud shade to blend the patches. Moving to another viewpoint is simple and fast; we don't have to recompute the radiosity as long as the scene itself is unchanged.

The expense of radiosity tends to lead people not to use it for cases other than when accuracy is more important than time; and when the scene itself is fixed but viewed many times. In fact most of its good features can be emulated by clever ray tracing or surface rendering techniques, albeit at lower accuracy.

5.3 Progressive Radiosity

The major problem of the standard approach is the $O(N^2)$ cost of calculating all the form factors and indeed the $O(N^2)$ cost storing them. Progressive radiosity is a popular implementation variant, which reduces both costs to $O(N)$.

Instead of working on solutions to all patches simultaneously, the effects of one specific patch on all the others are calculated in full. Then the process is looped over all patches, accumulating contributions. If we iterate this process "for ever", we will arrive at the same solution as with the complete matrix of equations. The method has the advantage that intermediate results can be viewed at each overall iteration. Moreover, "important" surfaces can be processed first (e.g. light sources; generally choose the ones with highest unshot radiosity) and the process can be stopped when little change is happening. As described, this is sometimes called the "shooting" variant because the energy from one patch is shot out to all others. We can of course do it the other way round and calculate the effect of all other patches on the chosen patch, when it is

called the "gathering" variant.

When working iteratively, we have to be careful not to count energies more than once. We initialise all radiosities to zero, except for those of the light source patches. The calculation we make is the *change* in radiosity at all patches j due to patch i, which requires calculation of the form factor using the hemicube sat on patch i. We then repeatedly update these change values by running over all i. Now we can update the actual radiosity values and show the intermediate result, resetting the change values to zero in case we do another complete iteration.

Further improvements continue to appear from time to time, to reduce costs or extend the application areas. Radiosity cannot handle highlights so is often combined with ray-tracing or lightmap techniques where these are visually important. In general, radiosity remains an active general research area, while ray-tracing is essentially fully evolved, except for specialised work and parallelism.