Global Illumination and Radiosity

OUTLINE:

What is Global Illumination?

Radiosity Methods

Radiosity Equations

Progressive Radiosity

The Math Behind Radiosity

Global Illumination

Observation: light comes from other surfaces, not just designated light sources.

Goal: simulate interreflection of light in 3-D scenes.

***Difficulty*: you can no longer shade surfaces one at a time, since they're now interrelated!**

Two general classes of algorithms:

1. **ray tracing methods**: simulate motion of photons one by one, tracing photon paths either backwards ("eye ray tracing") or forwards ("light ray tracing") -- good for specular scenes

2. **radiosity methods**: set up a system of linear equations whose solution is the light distribution -- good for diffuse scenes

Classical Radiosity Method

Definitions:

surfaces are divided into **elements**

radiosity = integral of emitted radiance plus reflected radiance over a hemisphere. units: [power/area]

Assumptions:

- no participating media (no fog) → *shade surfaces only, not vols.*
- opaque surfaces (no transmission)
- **reflection and emission are diffuse** → ***radiance is direction-indep.***, *radiance is a function of 2-D surface parameters and λ*
- reflection and emission are independent of λ within each of several wavelength bands; typically use 3 bands: R,G,B → *solve 3 linear systems of equations*
- radiosity is constant across each element → *one RGB radiosity per element*

Typically (but not exclusively):

- surfaces are polygons, elements are quadrilaterals or triangles

The Unit of Radiosity

Radiance (a.k.a. **intensity**) is power from/to an area in a given direction.

units: power / (area × solid angle)

Radiosity is outgoing power per unit area due to emission or reflection over a hemisphere of directions.

units: power / area

radiosity = radiance × integral of [cos(polar angle) × d(solid angle)] over a hemisphere = π × radiance

So radiosity and radiance are linearly interrelated.

Thus, "radiosity" is both a unit of light and an algorithm.

Radiant emitted flux density is the unit for light emission.

units: power / area

Deriving Radiosity Equations, 1

$$\begin{pmatrix}\text{outgoing}\\ \text{power}\\ \text{of elem } i\end{pmatrix} = \begin{pmatrix}\text{power}\\ \text{emitted}\\ \text{by elem } i\end{pmatrix} + \begin{pmatrix}\text{power}\\ \text{reflected}\\ \text{by elem } i\end{pmatrix}$$

$$\begin{pmatrix}\text{outgoing}\\ \text{power}\\ \text{of elem } i\end{pmatrix} = \begin{pmatrix}\text{power}\\ \text{emitted}\\ \text{by elem } i\end{pmatrix} + \begin{pmatrix}\text{reflectance}\\ \text{of elem } i\end{pmatrix} \times \sum_{\text{elem } j} \begin{pmatrix}\text{fraction of power}\\ \text{leaving elem } j \text{ that}\\ \text{arrives at elem } i\end{pmatrix} \times \begin{pmatrix}\text{outgoing}\\ \text{power}\\ \text{of elem } j\end{pmatrix}$$

Let

A_i = area of element i (computable)

e_i = radiant emitted flux density of element i (given)

ρ_i = reflectance of element i (given)

b_i = radiosity of element i (unknown)

F_{ij} = form factor from i to j = fraction of power leaving i that arrives at j (computable)

So the equation above can be rewritten :

$$A_i b_i = A_i e_i + \rho_i \sum_{j=1}^{n} F_{ji} A_j b_j$$

Form Factors

Define the **form factor** F_{ij} to be the fraction of light leaving element i that arrives at element j

$$F_{ij} = \frac{1}{A_i} \int_{A_i}\int_{A_j} \frac{\cos\theta_i \cos\theta_j}{\pi r^2} v_{ij}\, dA_j\, dA_i$$

Where

v_{ij} is a boolean **visibility** function: 0 if point on i is occluded with respect to point on j, 1 if unoccluded.

This is a double area integral. Difficult! We end up approximating it.

dA_i and dA_j are infinitesimal areas on elements i and j, respectively

θ_i and θ_j are polar angles: the angles between ray and normals on elements i and j, respectively

Projected area of dA_i from j is $\cos\theta_i dA_i$, hence the cosines

r is distance from point on i to point on j

Reciprocity law: $A_i F_{ij} = A_j F_{ji}$.

Deriving Radiosity Equations, 2

Earlier, we had : $A_i b_i = A_i e_i + \rho_i \sum_{j=1}^{n} F_{ji} A_j b_j$

Dividing by A_i : $b_i = e_i + \rho_i \sum_{j=1}^{n} F_{ji} \frac{A_j}{A_i} b_j$

By the reciprocity law, $F_{ji}(A_j / A_i) = F_{ij}$, so $b_i = e_i + \rho_i \sum_{j=1}^{n} F_{ij} b_j$ for all elems i

Or, in matrix/vector notation :
b = e + Kb
where **b** is the vector of unknown radiosities, **e** is the vector of known emissions,
and **K** is a square matrix of reflectance times form factor : $K_{ij} = \rho_i F_{ij}$
Subtracting **Kb** from both sides, we get **b - Kb = e**, or
(I - K)b = e
where **I** is an identity matrix.
This is a linear system of n equations in n unknowns (the b_i).

There are three such systems of equations, one for the red channel, one for green, and one for blue. The variables ρ_i, e_i, and b_i are RGB vectors.

Computing Visibility for Form Factors

Computing visibility in the form factor integral is like solving a hidden surface problem from the point of view of each surface in the scene.

Two methods:

ray tracing: easy to implement, but can be slow without spatial subdiv.

hemicube: exploit speed of z-buffer algorithm, compute visibility between one element and all other elements. Good when you have z-buffer hardware, but some tricky issues regarding hemicube resolution

You end up approximating the double area integral with a double summation, just like numerical methods for approximating integrals.

When two elements are known to be inter-visible (no occluders), you can use analytic form factor formulas and skip all this.

Two Radiosity Algorithms

Matrix radiosity: Compute form factors and store the matrix **I-K**, solve system of equations, then display.

Solving can be done in several ways (Successive Overrelaxation, Gauss-Seidel, Conjugate Gradient, ...)

Time and memory cost: O(n^2), where n=#elements.

n is commonly many thousand, so the memory cost is excessive.

Progressive radiosity: Solves system of equations incrementally as matrix of form factors is computed, one column at a time. Partial solutions can be displayed, yielding *progressive refinement* of image.

Time cost: O(ns) where s=#shooting steps

Memory cost: O(n)

This is much more commonly used.

Progressive Radiosity, Intuitively

- Turn hemicube around so that you're shooting light out to other elements, not gathering it in from other elements.
- Shoot from light sources initially, then shoot from reflective surfaces in decreasing order of brightness, treating reflective surfaces as *secondary light sources*, like repeated application of a shadow algorithm.
- Display an approximate picture as you go (optional).
- Converges to correct solution in the limit as you shoot an infinite number of times, and hopefully it will reach an acceptable approximation very quickly.
- If you stop after only lights have "shot", you've simulated shadows only, not interreflection.

Progressive Radiosity Algorithm

generate mesh by subdividing polygons into elements
for each element i
    $b_i \leftarrow e_i$ *radiosity*
    $\Delta b_i \leftarrow e_i$ *unshot radiosity*
until convergence   *(quantitative, or user gets impatient)*
    i = index of element with maximum "*unshot power*" $A_i \Delta b_i$
    Compute F_{ij} for all elements j using hemicube or ray tracing
    for each element j
        $\Delta Rad \leftarrow \rho_j \Delta b_i F_{ij} A_i / A_j$ *incremental radiosity shot from i to j*
        $b_j \leftarrow b_j + \Delta Rad$ *update total radiosity of element j*
        $\Delta b_j \leftarrow \Delta b_j + \Delta Rad$ *update unshot radiosity of element j*
    $\Delta b_i \leftarrow 0$ *reset unshot radiosity for element i to zero*
    display scene using radiosities b_j, if desired

This is progressive radiosity without substructuring. Substructuring (shooting from polygons to elements, not from elements to elements) speeds things up.

Systems Issues

All algorithms require the following operations:

1. Input scene (geometry, emissions, reflectances).
2. Choose mesh (important!), subdividing polygons into elements.
3. Compute form factors using ray tracing or hemicube for visibility (expensive).
4. Solve system of equations (indirectly, if progressive radiosity).
5. Display picture.

If mesh is chosen too coarse, approximation is poor, you get blocky shadows.

If mesh is chosen too fine, algorithm is slow. A good mesh is critical!

In radiosity simulations, because scene is assumed diffuse, surfaces' radiance will be view-independent.

Changes in viewpoint require only visibility computations, not shading. Do with z-buffer hardware for speed. This is commonly used for "architectural walkthroughs" and virtual reality.

Changes in scene geometry or reflectance require a new radiosity simulation.

Summary of Radiosity Algorithms

Radiosity algorithms allow indirect lighting to be simulated.

Classical radiosity algorithms:

- Generality: limited to diffuse, polygonal scenes.
- Realism: acceptable for simple scenes; blocky shadows on complex scenes. Trial and error is used to find the right mesh.
- Speed: good for simple scenes. If all form factors are computed, $O(n^2)$, but if progressive radiosity or newer "hierarchical radiosity" algorithms are used, sometimes $O(n)$.

Generalizations:

- **curved surfaces**: easy - radiosity samples are like a surface texture.
- **non-diffuse (specular or general) reflectance**: much harder; radiosity is now a function of not just 2-D position on surface, but 2-D position and 2-D direction. Lots of memory required, but it can be done.

The Math Behind Radiosity

Radiosity methods are the discrete way to think about global illumination.

The continuous way is:

$$\text{radiosity}(x) = \text{emitted}(x) + \text{reflectance}(x) \times \int_{\substack{\text{other}\\\text{surface}\\\text{points } t}} \text{formfactor}(x,t) \times \text{radiosity}(t)\, dt$$

where x and t are surface points

This is called an **integral equation** because the unknown function "radiosity(x)" appears inside an integral.

Can be solved by radiosity methods or randomized "Monte Carlo" techniques also, by simulating millions of photon paths.

References

Some good books and papers on the radiosity method:

[Cohen & Wallace, Radiosity and Realistic Image Synthesis]
[Sillion & Puech, Radiosity & Global Illumination]
[Ashdown, Radiosity: A Programmer's Perspective]

[Cohen, SIGGRAPH 88] -- paper on progressive radiosity algorithm