Radiosity

Ray tracing handles only some global illumination.

- It does simulate global specular reflections.
- It does *not* simulate global diffuse reflections.

Diffuse reflections of light are related to heat reflections.

Two groups adapted thermal-engineering methods to produce *radiosity* algorithms for global illumination:

- Goral, Torrance, Greenberg, Battaile in 1984
- Nishita, Nakamae in 1985

Radiosity is a *preprocess* for a rendering system.

- Assume all objects have diffuse surfaces.
- The preprocess simulates the global diffuse reflections to find each surface's illuminated color.
- The rendering system then produces the image.
  - It finds the visible surfaces and scan-converts them.
  - It uses the colors from the preprocess and does not use an illumination model of its own.

If the surfaces do not change, the preprocessed colors can be used for many different renderings.

- All surfaces are diffuse, so their illuminated colors do not change with the camera position.
- The results of radiosity are *view independent*.

Radiosity does not handle specular reflection, though.

Radiosity

Break all surfaces down into small *patches*.

Let n be the total number of patches in the scene.

Let A_i be the area of patch i.

Consider illumination in each primary color separately.

Any patch can emit light.

- E_i is the power per unit area emitted by patch i.
- $A_i E_i$ is the total power emitted by patch i.
- Emission is uniform in all directions and constant over the patch.

A patch can also reflect light emitted by other patches.

- ρ_i is the *reflectivity* of patch i, the fraction of the power incident on patch i that it reflects.
- Reflectivity is a property of the patch's material.
- Reflection is uniform in all directions and constant over the patch.

Radiosity

The total power produced by a patch sums its emission and reflection.

- B_i is the *radiosity* of patch i, its total power per unit area.
- $A_i B_i$ is the total radiosity for patch i.
- Radiosity is uniform in all directions and constant over the patch.

The radiosity B_i involves all n other patches.

$$A_i B_i = A_i E_i + \rho_i \sum_{1 \le j \le n} A_j B_j F_{j \to i}$$

How much light from patch j is reflected by patch i depends on the patches visibility to each other.

- It depends on relative position, orientation, and occlusion. Models radiative exchange between patches.
- $F_{j \to i}$ is the *form factor*, the fraction of the power leaving patch j that reaches patch i.
- $F_{j \to i} = 0$ if the patches are completely occluded.
- The form factor is constant over patch i.

Radiosity

The reciprocity relationship holds for form factors:

$$A_i F_{i \to j} = A_j F_{j \to i} \qquad F_{i \to j} = (A_j / A_i)\, F_{j \to i}$$

- Say $A_i = 2A_j$ and the patches are arranged simply.
- Patch i receives all of patch j's power.
- But patch j receives only half of patch i's power.

It is customary to use this relationship when defining B_i:

$$\frac{1}{A_i}\left(A_i B_i = A_i E_i + \rho_i \sum_{1 \le j \le n} A_j B_j F_{j \to i} \right)$$

$$B_i = E_i + \rho_i \sum_{1 \le j \le n} B_j F_{i \to j}$$

The terms can be rearranged:

$$B_i - \rho_i \sum_{1 \le j \le n} B_j F_{i \to j} = E_i$$

There are now n linear equations in the n unknown Bs.

$$\begin{bmatrix} 1 - \rho_1 F_{1 \to 1} & -\rho_1 F_{1 \to 2} & \dots & -\rho_1 F_{1 \to n} \\ -\rho_2 F_{2 \to 1} & 1 - \rho_2 F_{2 \to 2} & \dots & -\rho_2 F_{2 \to n} \\ \dots & \dots & \dots & \dots \\ -\rho_n F_{n \to 1} & -\rho_n F_{n \to 2} & \dots & 1 - \rho_n F_{n \to n} \end{bmatrix} \cdot \begin{bmatrix} B_1 \\ B_2 \\ \dots \\ B_n \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ \dots \\ E_n \end{bmatrix}$$

Iterative techniques will solve for the unknown Bs.

Form factors

The form factor $F_{i \to j}$ is the fraction of the power leaving patch *i* that reaches patch *j (radiative energy striking j from j to all radiative energy leaving i, in a hemisphere).*

It depends on geometry between patches only and (partial) oclusion.

$$F_{i \to j} = \int\limits_{A_i}\int\limits_{A_j} \frac{\cos\phi_i \cdot \cos\phi_j}{\pi \cdot r^2} dA_i dA_j$$

Properties: reciptocity, conservation of enery, convexity

This intergral is difficult to solve (also has to multiply by occluding factor depending if area elements on both surfaces are visible to each other or not).

- The patch-to-patch form factor is approximated by area-element-to-patch form factor, when patch areas are small relative to distance.
- Under these conditions , that area-element-to-patch form factor is equal to the form factor between the area element and *the projection* of A_j on the *hemisphere* surrounding the erea element for patch *i,* dA_i.
- Also, patches that produce the same projections on the hemisphere have the same form factors.

- Thus, we can approximate the form factor from i of any patch *j* by considering not the patch itself, but its projection on the faces of a *hemicube* - this is the justification for the hemicube method.

Hemicube is constructed around the center of each patch.

The hemicube Z axis and the patch normal coincide.

The faces of the hemicube are devided into *pixels.*

All other patches are projected onto the hemicube.

If projections of two patches overlap on a pixel , rge further patch is rejected (ocluded) .

Notice analogy to Z-buffer algorithm.

- Each pixel on the hemicube is considered as a small patch,and corresponding form factors (*delta form factors)* are precalculated and stored in a look-up table
- For any patch *j,* the form factor i to j, is a pproximated by summing the corresponding form factors for the pixels on which the *j*th patch projected on the hemicube. 3D version of Sutherland-Hodgman clipper is used to clip patch projections against the hemicube.
- Form factor calculation becomes: projections onto mutually porthogonal planes and then summing

Radiosity method summary

1. Compute form factors
2. Solve radiosity matrix equation
3. Use the obtained results for "vertex color" standard rendering scheme (e.g. using hidden surface removal and Gouraud type shading). Can reuse the results from previous step for rendering from different view points.

- Comments:Precompute FF, store , reuse
  - Form factors depend only on gemetry, calculated once and reused if geometry does not change. They can be used for different reflectivity and light source values. For example, can render same scene with some light sources off.
  - Form factors recalculations is neccessary only if geometry changes.
  - The solution in 2 is view-independent. If light is changed, but geometry not, then only step 2 is recalculated, not 1.
  - In step 3, can reuse the results from previous step for rendering from different view points. This is useful in animating walkthrough a building interior
  - Difficult to code! Expensive: stodage and calculation of form factors.
  - Accuracy: depends on hemicubical resolution. Aliasing could be a problem.

Improving radiosity

Many researchers have improved radiosity's accuracy and efficiency.

Two interesting improvements:

- *Progressive refinement*: iteratively improve the solution so the quality is proportional to the processing time, displaying results as the solution progresses. Light from each patch is shot into the environment and the entire scene is updated for each iteration:
  - find the patch with gratest radiosity or emitted energy
  - evaluate form factors from this patch to every other patch in the environment
  - update the radiocity of each of the receiving patches
  - update current ambient term (included so early iterations are voisible) as a function of sum of the differences of the values calculated in current and previous itterations
- *Discontinuity meshing*: look for shadow boundaries to avoid blurring them.

Combining ray tracing and radiosity

It would be nice to have one algorithm that handled:

- global specular reflection like ray tracing
- global diffuse reflection like radiosity

The following approach might seem reasonable:

- First, use radiosity to find the global diffuse reflection.
- Then, use ray tracing to add the global specular reflection, using the radiosity solution instead of local diffuse and ambient terms.

Unfortunately, this approach does not correctly handle some categories of reflections:

- light reflected off a mirror onto a diffuse surface
- that light reflected further onto another mirror

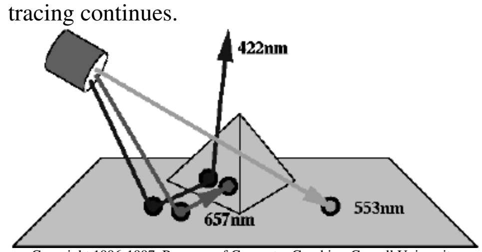
Most attempts to overcome these problems are complicated and very computationally expensive.

Global illumination by density estimation

This algorithm accounts for specular and diffuse reflection by tracing "light particles" emitted by surfaces.

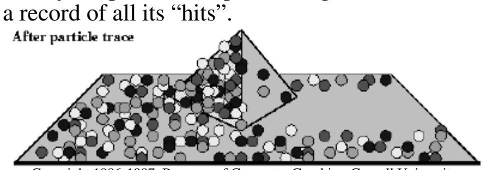
When a particle hits a surface:

- The surface keeps a record of the "hit."
- The surface may reflect the particle, in which case the tracing continues.



- More reflective surfaces are more likely to reflect.
- The direction of reflection is probabilistic, depending on how specular or diffuse the surface is.

Eventually, all particles stop reflecting and each surface has a record of all its "hits".



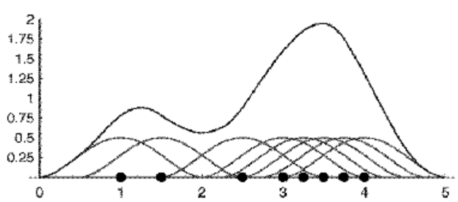
Global illumination by density estimation

The algorithm reconstructs a continuous illumination function on each surface from its "hits".

- Illumination is higher where there are more "hits."
- Illumination is lower where there are fewer "hits."

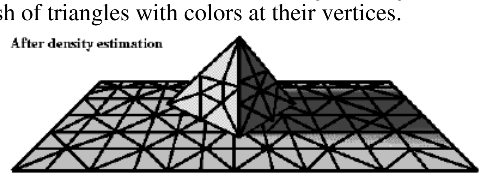
The statistical technique of *density estimation* performs the reconstruction.

- Intuitively, each "hit" contributes a "lump."
- The sum of the "lumps" is the reconstructed function.



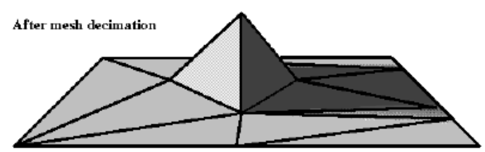
- There are also more sophisticated approaches.

From the continuous function, the algorithm generates a mesh of triangles with colors at their vertices.



Global illumination by density estimation

Finally the algorithm simplifies the mesh of triangles (as if generating LODs).



This approach has several advantages:

- It correctly accounts for all forms of reflection.
- It uses less memory in return for using more disk storage.
- It works well on a parallel-processing system.