Johns Hopkins
Engineering for Professionals

605.767 Applied Computer Graphics

Brian Russin

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

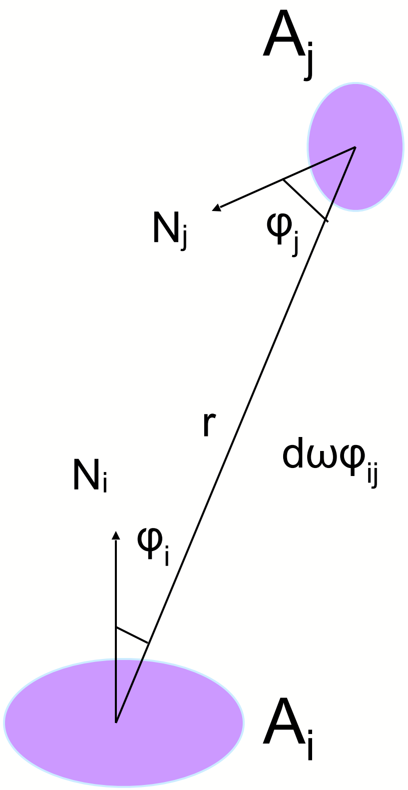
Module 4B
Implementing Radiosity

Problems with Form Factors

- Form factor between two surfaces is the percentage of energy emanating from area A_j that is incident on A_i
- n^2 form factors must be calculated for an environment
  - Estimated an order of magnitude more computation than solving the radiosity matrix or rendering the solution
  - Storage space may limit scene complexity
    - 10,000 patches requires 400,000,000 bytes to store the form factors
      - 4 bytes per form factor
    - Even though form factor matrix may be 90% sparse
      - Most patches cannot see each other
- Form factor calculation between 2 patches must take into account any intervening patches

Form Factors

- Form factors between 2 infinitesimal surfaces with differential areas dA_i and dA_j
  - $d\omega_{ij} = \frac{\cos\varphi_j}{r^2} dA_j$
    - Angle seen from dA_i
  - Energy leaving dA_j that reaches dA_i is:
- $dE_i dA_j = \frac{I_i \cos\varphi_i \cos\varphi_j}{r^2} dA_i dA_j$
- Form factor is defined as:
  - $F dA_i dA_j$ = Radiative energy reaching dA_i from dA_j / Total energy leaving dA_i in all directions
  - For ideal Lambertian surface: $E_i = I_i \pi$
    - Intensity is independent of direction
- $F dE_i dA_i = \frac{\cos\varphi_i \cos\varphi_j}{\pi r^2} dA_j$



Form Factors (cont)

- Integrate: $F_{dA_i A_j} = \int_{A_i} \frac{\cos\varphi_i \cos\varphi_j}{\pi r^2} dA_j$
  - Computes the form factor from the differential area dA_i to the patch A_j
- Patch to patch form factor:
  - F_{ij} = Radiative energy reaching A_i from A_j / Total energy leaving A_i in all directions
  - $$F_{ij} = \frac{1}{A_i} \int_{A_i} \int_{A_j} \frac{\cos\varphi_i \cos\varphi_j}{\pi r^2} dA_j dA_i$$
    - Integrate over A_i and take the area average
    - Solving requires two integrals: difficult to perform analytically!
- Sum of form factors = 1
  - Form factors are fractions of total energy leaving the patch
  - In closed environment that energy must end up in other patches

Form Factor Calculation Methods

- Watt and Watt list several methods of calculating form factors
  - Numerical or analytical
    - Earliest approach used direct numerical integration
    - Converts the double area integral into a double contour integral (Stoke's theorem)
    - Computationally expensive
    - Difficult to handle intervening patches
  - Hemicube
    - Most common method
    - Efficient but approximate
    - Easily handles intervening patches
  - Ray tracing
    - Rays are traced through hemisphere
  - Hybrid approach
    - Use hemicube for most calculations
    - Analytical approach when hemicube approximation is not as good: for light sources and when patches are close together

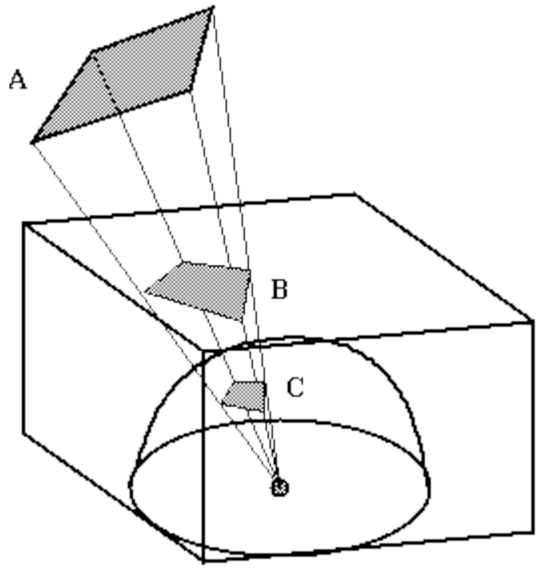
Hemicube Form Factor Determination

- Hemicube is an efficient but approximate method
  - Solves the visibility problem (intervening patches)
- Two justifications:
  - If distance between two patches (r) is large compared to the area of the patch then the value of the inner integral does not change much over the range of the outer integral
    - Effect of the outer integral is approximately multiplication by 1
    - That is the area to area form factor is approximated by the differential area to area form factor.
  - Any patch that has the same projection onto the surface of the hemisphere has the same form factor
    - Can project onto a cube rather than a hemisphere
    - **Nusselt analogue**

Nusselt Analogue

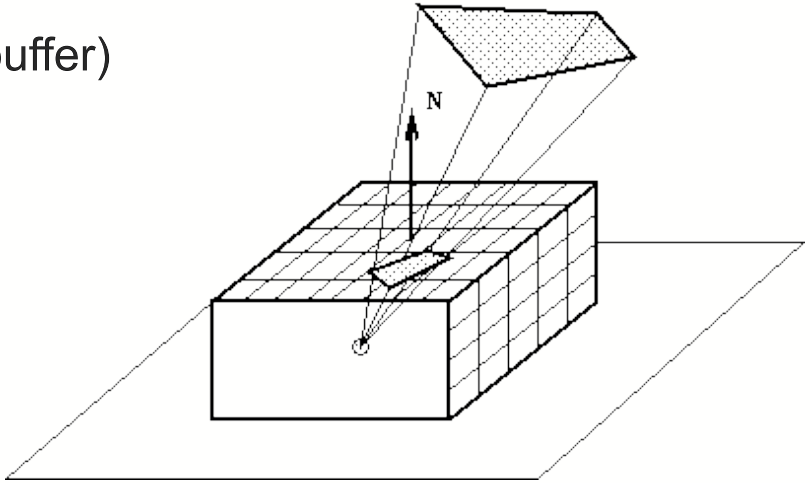
The **Nusselt analogue** states that considering the projection of a patch A_j onto the surface of a hemisphere surrounding dA_i is equivalent to considering the patch itself. In addition patches that produce the same projection on the hemisphere have the same form factor.

Thus the patches A, B and C below all have the same form factor.



Hemicube Form Factor Determination

- Method
  - Hemicube is constructed around each patch
    - Hemicube z axis and the patch normal are coincident
    - Faces of the cube are divided into elements called pixels
  - Every other patch is projected onto the hemicube
  - Form factor is obtained by summing the form factors of the pixels onto which patch j projects
  - https://education.siggraph.org/static/HyperGraph/radiosity/overview_2.htm
- Solves problem of intervening patches
  - For each pixel label which patch is closest (kind of a patch Z buffer)



Delta Form Factor

- Each pixel on the hemicube is considered a small patch
  - Delta form factor is precalculated
    - Differential to the finite area form factor

$$\Delta F_p = \frac{\cos\theta_i \cos\theta_j}{\pi r^2} \Delta A$$

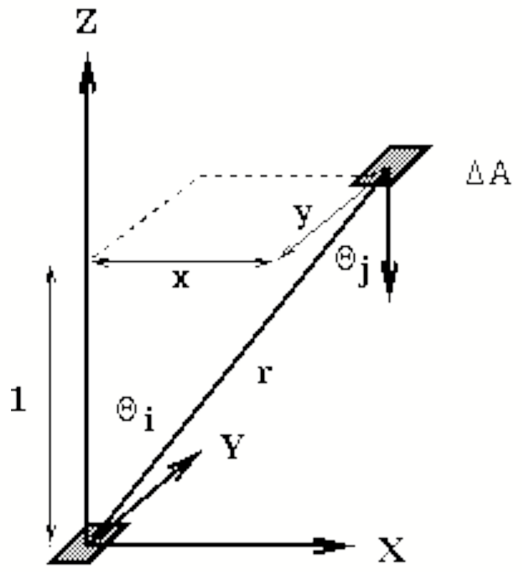
- Finding a form factor of a patch entails:
  - Project it onto faces of the hemicube
  - Find which pixels it intersects
  - Sum the form factors for each intersected pixel
- For top surface:

$$\cos\theta_i = \cos\theta_j = \frac{1}{r}$$

$$\Delta F_p = \frac{1}{\pi\left(x^2 + y^2 + 1\right)^2} \Delta A$$

- For side surface:

$$\Delta F_p = \frac{z}{\pi\left(y^2 + z^2 + 1\right)^2} \Delta A$$



Radiosity Algorithm Classification

- **Gathering**: Precompute a matrix of form factors, store, and solve radiosity matrix using Gauss-Seidel method, Successive Over Relaxation, or other method to solve system of equations.
  - Radiosity of a single patch is updated for each iteration by gathering radiosities from all other patches
- **Shooting**: Light from each patch i is distributed to all other patches and the entire scene is updated.
  - Visually optimized by ordering patches by amount of energy they are likely to radiate
- **Shooting and ambient**: An ambient term is included so early approximations are visible.
  - As successive iterations improve image quality the ambient term is reduced

Progressive Refinement

- Radiosity has large storage and processing requirements
  - Mostly to calculate form factors
  - Likely the reason it has not gained widespread use in the graphics community
- Progressive refinement speeds calculations and reduces memory requirements at each iteration
- Can display the rendered surfaces at each step to provide a sequence of increasing quality views
  - Dark to full lighting

Shooting and Ambient Progressive Refinement

- Contribution to the radiosity of patch j due to that from patch i:
  - B_i (from B_j) = $R_i B_i F_{ji}$
  - By reciprocity: B_j (from B_i) = $R_j B_i F_{ij} A_i / A_j$
    - Determine contribution from each patch j from a single patch i
- Image dark at first and light will be added at each iteration
  - Strategy is to start with patches with greatest radiosity (the light sources)
  - Use temporary ambient light factors
    - Reduce with each iteration B_i
- Four stages of the progressive refinement radiosity method:
  - 1) Find the patch with the greatest radiosity (emitted energy)
  - 2) Evaluate a column of form factors
    - Form factors from the patch to every other patch in the environment
  - 3) Update the radiosity of each of the receiving patches
  - 4) Reduce the temporary ambient term

Meshing Strategies

- Efficiency and accuracy of the radiosity solution depends on the quality of the meshing
  - Since each patch has equal radiosity (color) the patches must be defined to represent the distribution of light in a scene
  - For storage and performance want:
    - Large patches where light is constant
    - Small patches where light rapidly changes
      - Shadow boundaries, near light sources
- Two strategies
  - A priori – meshing is performed before radiosity solution is computed
    - Predict where discontinuities are going to occur and divide mesh accordingly
  - Adaptive meshing – a starting mesh is provided and the mesh itself is refined as the solution progresses

Progressive Refinement with Substructuring

- Watt: Section 11.7.1
- Progressive refinement with substructuring is a common radiosity strategy
  - For performance and quality reasons

Radiosity and Ray-Tracing Hybrid

- Can create a 2 pass renderer
  - 1st pass uses an enhanced radiosity method
    - Accounts for diffuse mechanisms
  - 2nd pass : ray tracing to deal with specular interactions
- Watt and Watt state that merging radiosity and ray tracing is not straightforward
  - Radiosity alone does not produce acceptable rendering in most cases due to its inability to deal with specular interactions
  - Still an area of research

Performance

- "An Empirical Comparison of Radiosity Algorithms" - Carnegie Mellon
  - Andrew H. Willmott and Paul S. Heckbert
  - https://www.cs.cmu.edu/~radiosity/emprad-tr.html
- "Matrix" method is only feasible for simple scenes
  - Storage is $O(n^2)$, n is the number of patches
  - Computing the matrix is most costly step
    - $O(n^2 v)$ to calculate the form factors, v is cost of determining visibility
- Progressive radiosity
  - For s shooting steps, the time cost is O(snv)
    - Converges faster than matrix radiosity so s is much less than n
  - Storage is only O(n): matrix columns are discarded after use
  - Substructuring is recommended
- Wavelet radiosity and hierarchical radiosity
  - Use adaptive, multilevel meshes
  - More complex but can get better performance on moderate complexity scenes

Radiosity