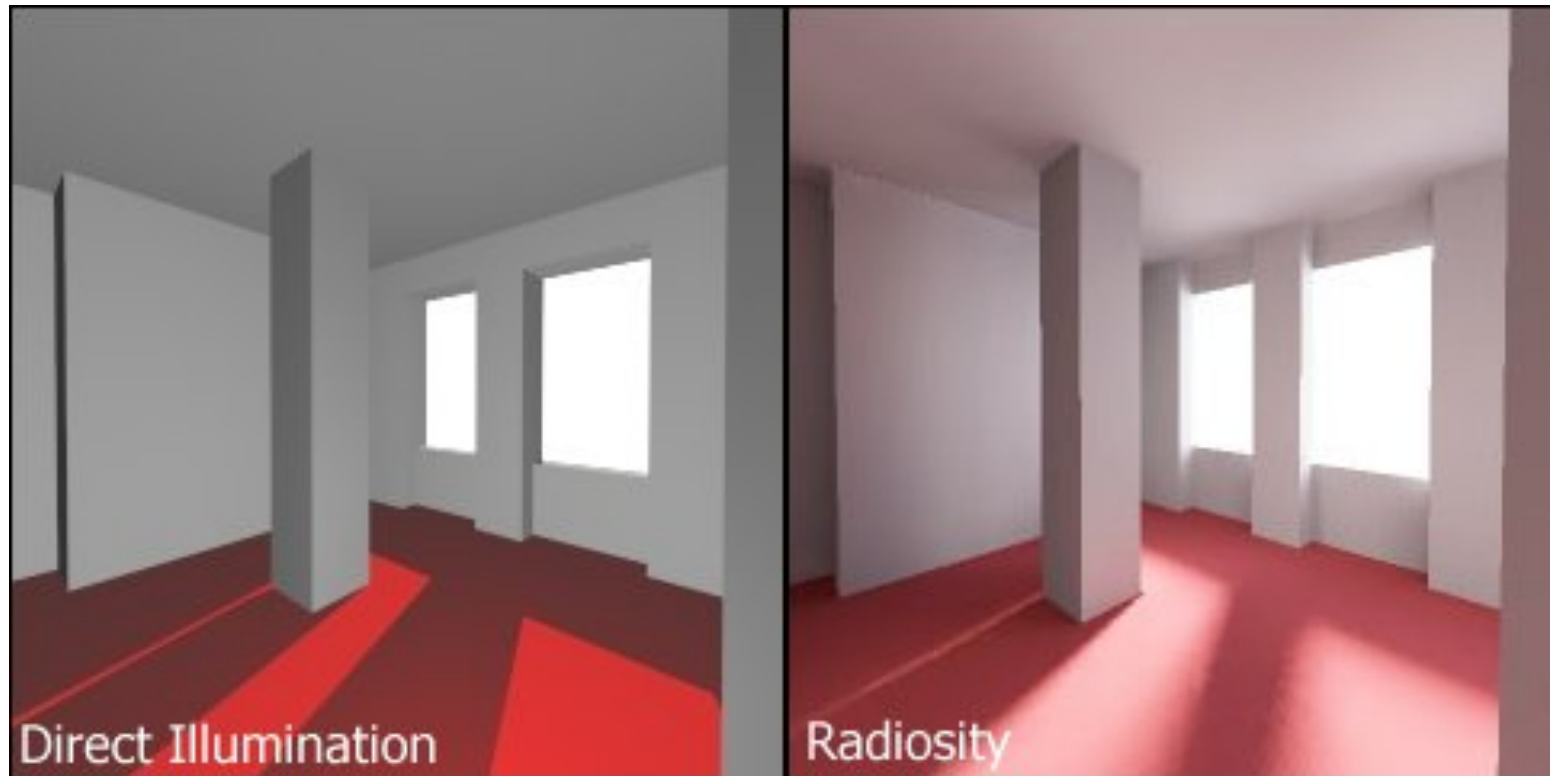


*Interactive Computer Graphics:
Lecture 13*

Radiosity

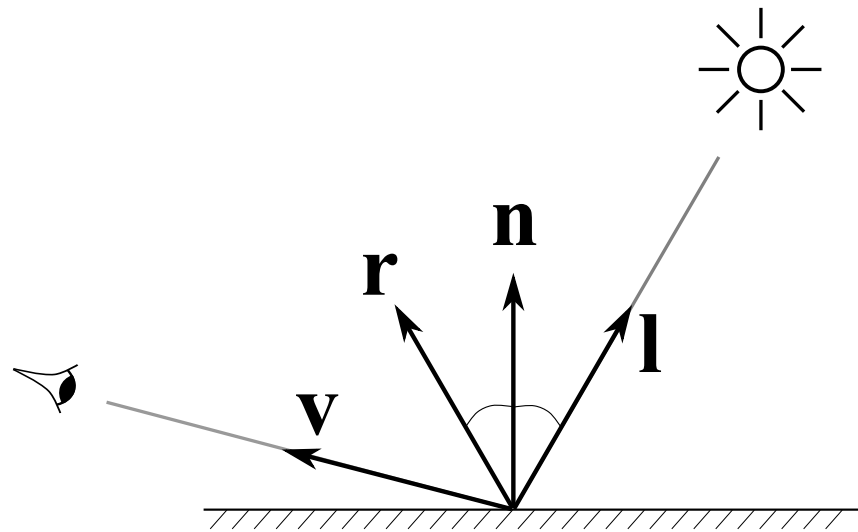




The reflectance equation

- Recall the reflectance equation: models reflected light from a surface:

$$I_{reflected} = k_a + k_d (\mathbf{n} \cdot \mathbf{l}) I_{incident} + k_s (\mathbf{r} \cdot \mathbf{v})^q I_{incident}$$

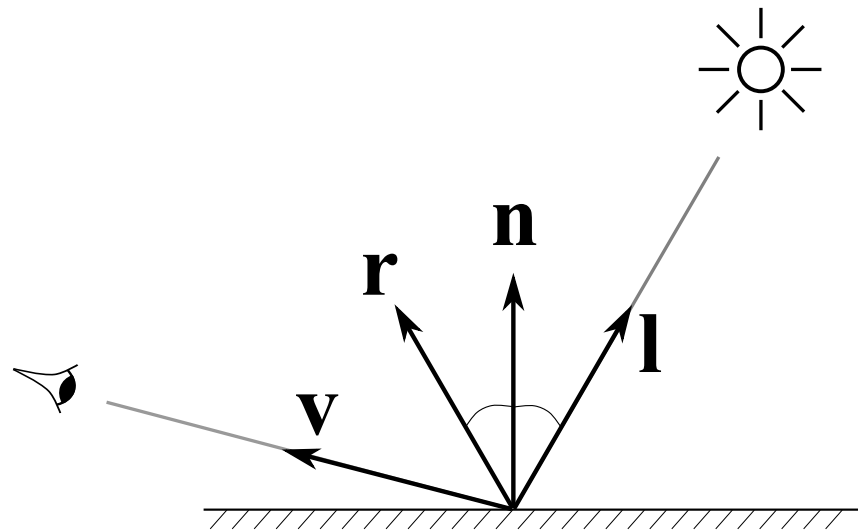


The reflectance equation

- Recall the reflectance equation: models reflected light from a surface:

$$I_{reflected} = k_a + k_d (\mathbf{n} \cdot \mathbf{l}) I_{incident} + k_s (\mathbf{r} \cdot \mathbf{v})^q I_{incident}$$

- $I_{incident}$: light intensity

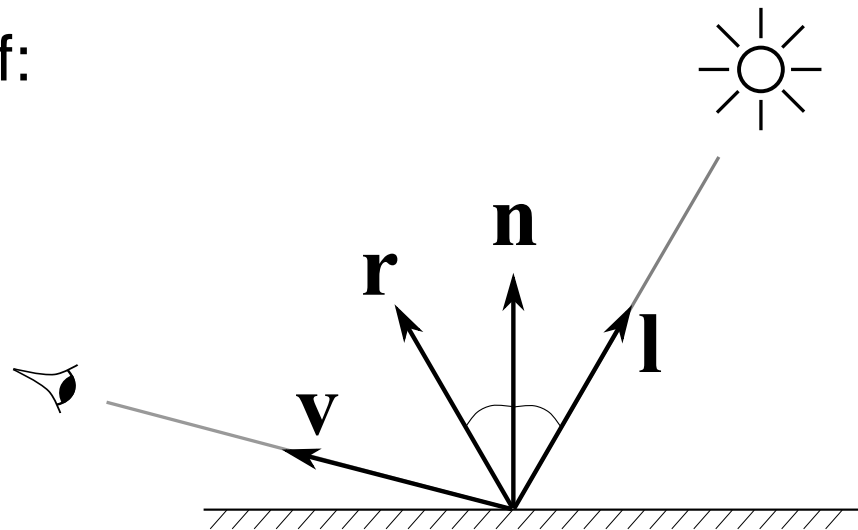


The reflectance equation

- Recall the reflectance equation: models reflected light from a surface:

$$I_{reflected} = k_a + k_d (\mathbf{n} \cdot \mathbf{l}) I_{incident} + k_s (\mathbf{r} \cdot \mathbf{v})^q I_{incident}$$

- $I_{incident}$: light intensity
- $k_{\{a,d,s\}}$: control amounts of:
 - k_a : ambient light
 - k_d : diffuse light
 - k_s, q : specular reflection



Lighting model for ray tracing

For ray tracing we assumed that there were 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, shouldn't we 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

$$I_{reflected} = k_a I_{incident} + k_d (\mathbf{n} \cdot \mathbf{l}) I_{incident} + k_s (\mathbf{r} \cdot \mathbf{v})^q I_{incident}$$

- Or, more simply, to write (for a given viewpoint)

$$I_{reflected} = R I_{incident}$$

- where R is the (viewpoint dependent) reflectance function.

Radiosity

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

Radiosity

- Divide the scene into patches $i = 1, \dots, n$
- For the i -th patch, let:
 - B_i = total energy leaving the patch
 - E_i = total energy emitted by patch itself
 - R_i = reflectance value
 - I_i = incident light energy arriving at the patch
- With this notation, the above equation can be re-written

$$B_i = E_i + R_i I_i$$

Collecting energy

We can estimate the incident energy for patch i as:

$$I_i = \sum_{j=1}^n B_j F_{ij}$$

where the sum is taken over all surface patches of the scene

The B_j 's in the sum represent the energy leaving all the other patches in the scene

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

We can assume that $F_{ii} = 0$

Final formulation

- We can substitute this expression for incident light into the previous equation
- We obtain a discrete approximation for the energy leaving the i -th patch:

$$B_i = E_i + R_i I_i \quad \text{becomes} \quad B_i = E_i + R_i \sum_j B_j F_{ij}$$

- The form factors F_{ij} take into account
 - Patch areas
 - The angle at which they ‘face’ each other
- They control the amount of energy leaving patch i that reaches patch i

In matrix form

- Re-write the equation for the i -th patch

$$B_i - \sum_j R_i B_j F_{ij} = E_i$$

- Joining the equations for all patches, we can formulate the matrix equation:

$$\begin{pmatrix} 1 & -R_1 F_{12} & -R_1 F_{13} & \cdot & \cdot & -R_1 F_{1n} \\ -R_2 F_{21} & 1 & -R_2 F_{23} & \cdot & \cdot & -R_2 F_{2n} \\ -R_3 F_{31} & -R_3 F_{32} & 1 & \cdot & \cdot & -R_3 F_{3n} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ -R_n F_{n1} & -R_n F_{n2} & -R_n F_{n3} & \cdot & \cdot & 1 \end{pmatrix} \begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_n \end{pmatrix} = \begin{pmatrix} E_1 \\ E_2 \\ \vdots \\ E_n \end{pmatrix}$$

In matrix form

$$\begin{pmatrix} 1 & -R_1 F_{12} & -R_1 F_{13} & \cdot & \cdot & -R_1 F_{1n} \\ -R_2 F_{21} & 1 & -R_2 F_{23} & \cdot & \cdot & -R_2 F_{2n} \\ -R_3 F_{31} & -R_3 F_{32} & 1 & \cdot & \cdot & -R_3 F_{3n} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ -R_n F_{n1} & -R_n F_{n2} & -R_n F_{n3} & \cdot & \cdot & 1 \end{pmatrix} \begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_n \end{pmatrix} = \begin{pmatrix} E_1 \\ E_2 \\ \vdots \\ E_n \end{pmatrix}$$

- If we can solve this for every B_i 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 need to be found
 - the full reflectance equation is insolvable
 - the matrix is big - minimum 10000 by 10000

Wavelengths

- The radiosity values are wavelength dependent, hence we will need to compute a radiosity value for R , G and B .
- Each patch will require a separate set of parameters for R , G and B .
- The three radiosity values are the values that the rendered pixels will receive.

Back to the reflectance function

$$I_{reflected} = k_a I_{incident} + k_d (\mathbf{n} \cdot \mathbf{l}) I_{incident} + k_s (\mathbf{r} \cdot \mathbf{v})^q I_{incident}$$

Note that the specular term depends on the relative positions of the viewpoint and each light source \mathbf{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

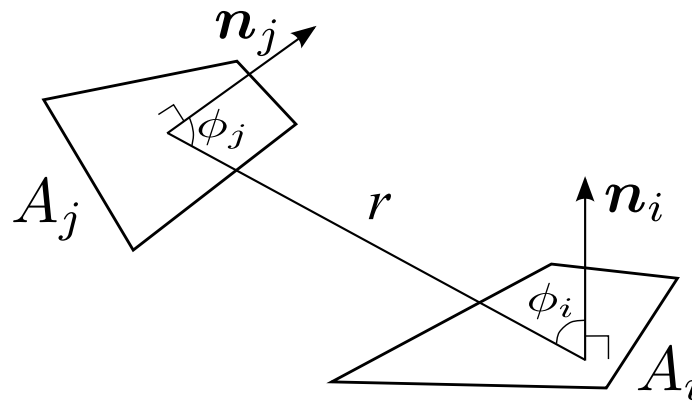
- We need to divide our graphics scene into patches for computing the radiosity.
- For small polygons we can perhaps use the polygon map, but for large polygons we need to subdivide them.
- Since the emitted light will not be constant across a large polygon we will see the subdivisions

Large Polygons

- Each patch will have a different but constant illumination.
- Thus we will see the patch boundaries 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.

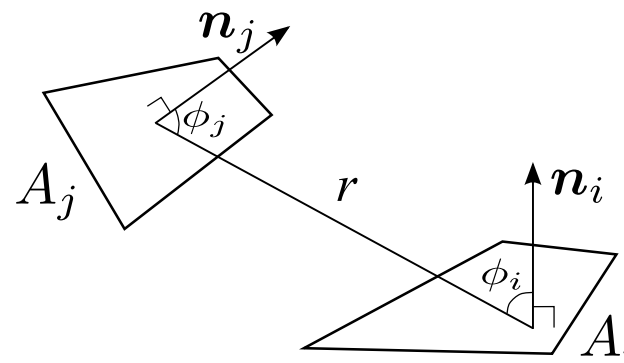


$$F_{ij} = \frac{1}{|A_i|} \int_{A_i} \int_{A_j} \frac{\cos \phi_i \cos \phi_j}{\pi r^2} dA_j dA_i$$

Form Factors - the definition

- The \cos terms compute the projection of each patch in the direction to the other.
- If the patches are in the same plane, facing the same way, there is no coupling. If they directly face each other they are maximally coupled.

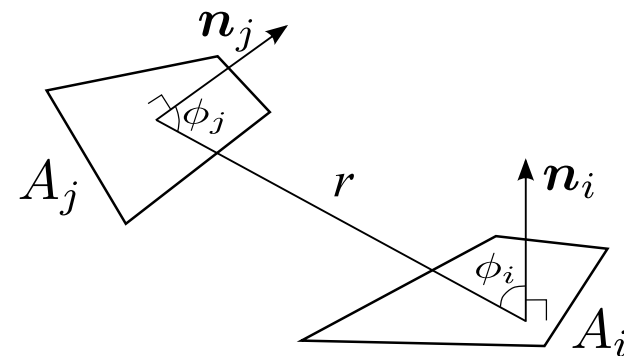
$$F_{ij} = \frac{1}{|A_i|} \int_{A_i} \int_{A_j} \frac{\cos \phi_i \cos \phi_j}{\pi r^2} dA_j dA_i$$



Form Factors - simplifying the computation

- The equation can be simplified if we assume that A_i is small compared with r .
- If this is the case, then we can treat the inner integral as constant over the surface of A_i .

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



Simplifying form factors

- With this assumption the outer integral evaluates to $|A_i|$ (i.e. the area of A_i).
- 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

We assumed that the radius is large compared with patch A_i . Should also be reasonable to assume it is large compared to the size of A_j .

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 A_j

So we get the approximation

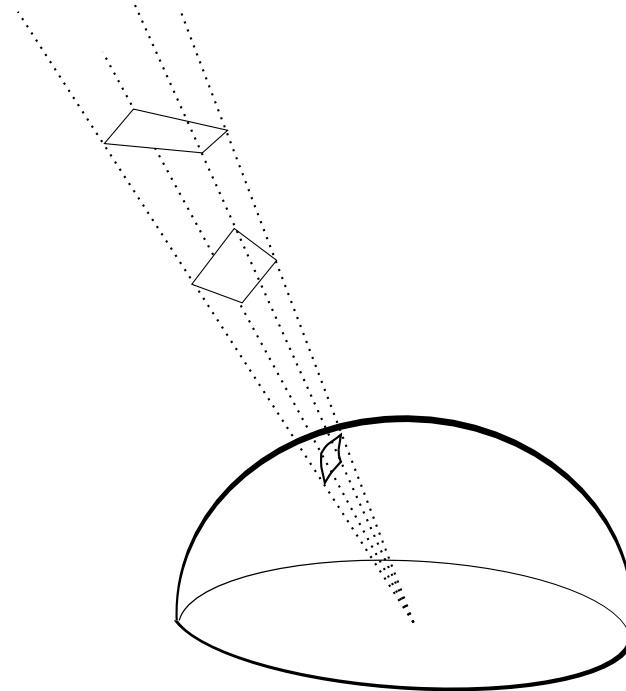
$$F_{ij} = \frac{\cos\phi_i \cos\phi_j |A_j|}{\pi r^2}$$

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

Direct computation of the approximate form factor equation for every pair of patches will be expensive to compute.

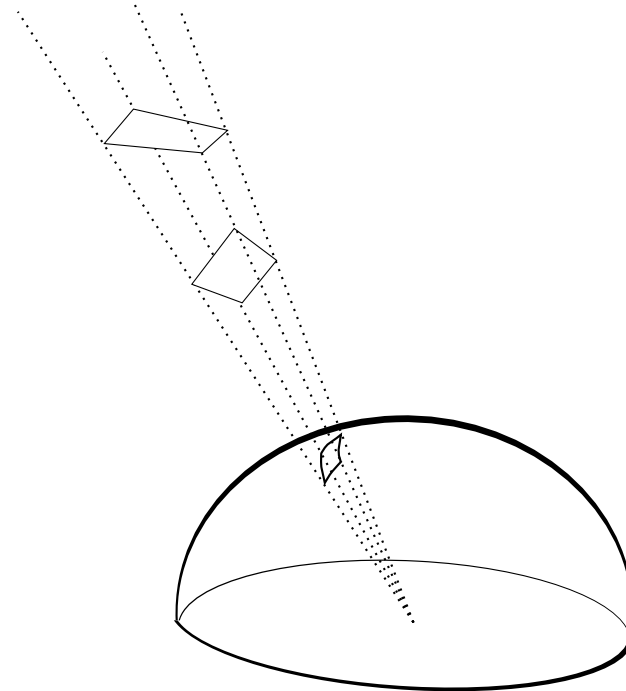
So an approximate computation method based on this observation (the hemi-cube) was developed.



The Hemicube method

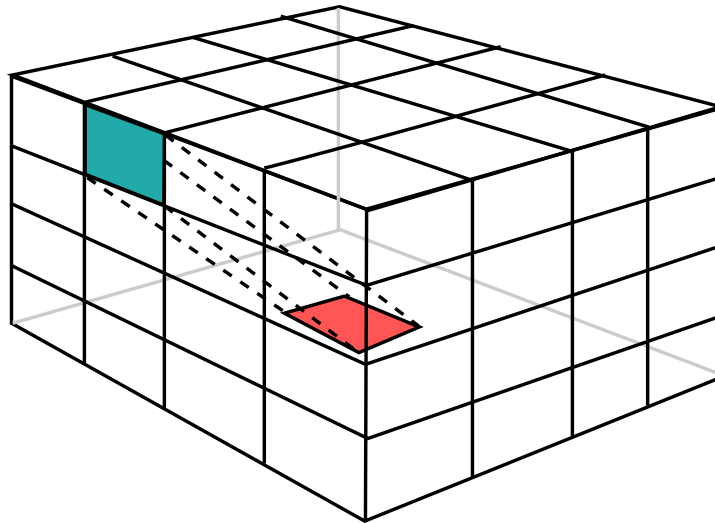
So, all patches that project onto the same area of a surrounding hemicube have approximately the same form factor.

The hemicube is preferred to the hemisphere since computing intersections with planes is computationally less demanding



Delta form factors

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



The resulting form factors can be used for every patch in the scene. We just 'place' the same hemicube over each.

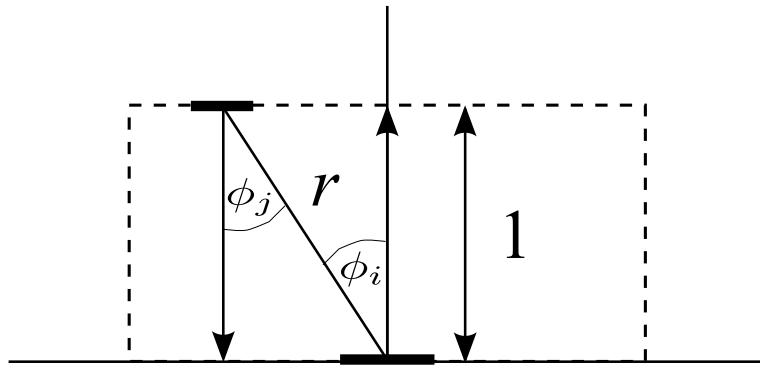
Delta form factors

- If the area of a hemicube pixel is $|A|$, its form factor is:

$$\frac{\cos \phi_i \cos \phi_j |A|}{\pi r^2}$$

- These delta form factors can be computed and stored in a look up table.
- They can then be applied to every patch without the need for further form factor calculations.

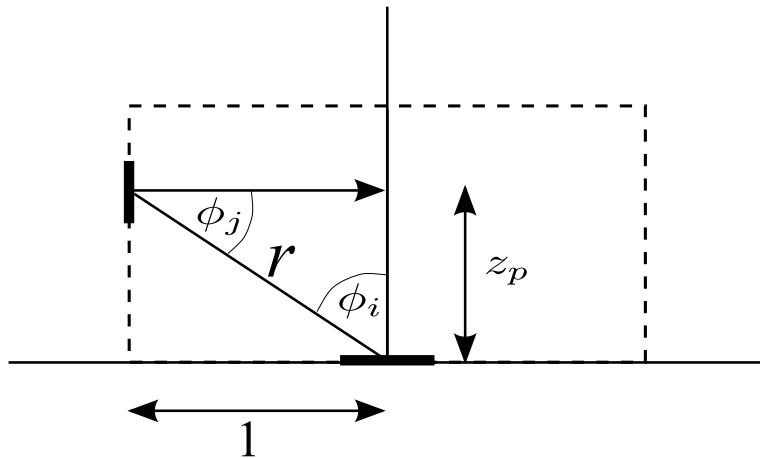
Computing the delta form factors



For a top face we have:

$$\cos \phi_i = \cos \phi_j = \frac{1}{r}$$

so the form factor is $\frac{|A|}{\pi r^4}$



For a side face we have:

$$\cos \phi_i = \frac{1}{r} \quad \cos \phi_j = \frac{z_p}{r}$$

so the form factor is $\frac{z_p |A|}{\pi r^4}$

Problem Break

Given that the hemi-cube 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, $z = 1$:

$$\{x_{min}, y_{min}, x_{max}, y_{max}\} = \{-0.05, -0.05, 0.05, 0.05\}$$

and

$$\{x_{min}, y_{min}, x_{max}, y_{max}\} = \{-0.45, -0.05, -0.55, 0.05\}$$

Solution

For the top face $\cos \phi_i = \cos \phi_j = \frac{1}{r}$ (because $z = 1$ on the top face)

Thus

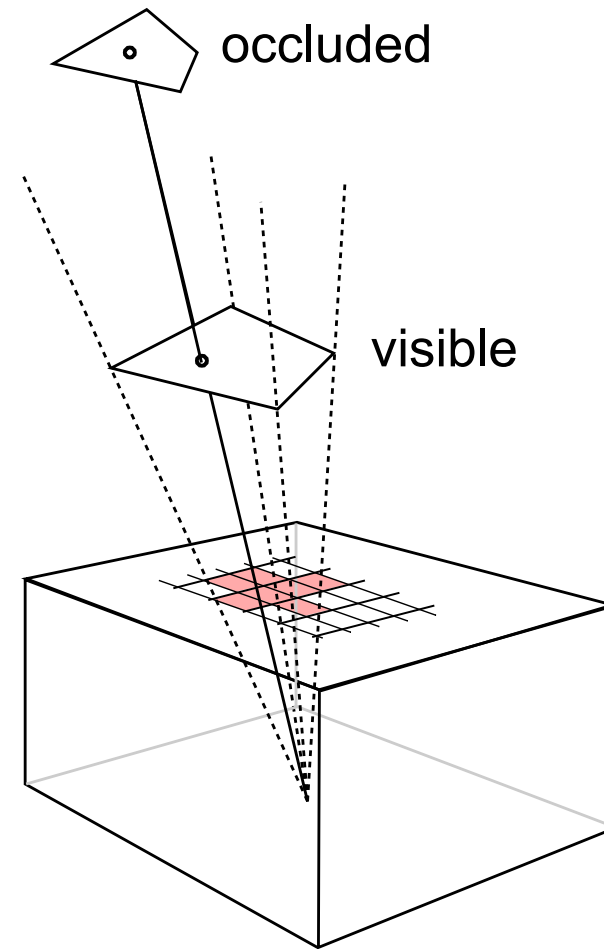
$$\frac{\cos \phi_i \cos \phi_j |A|}{\pi r^2} = \frac{|A|}{\pi r^4}$$

Case 1: $r = 1$ so form factor is $\frac{0.01}{\pi} = 0.00318$

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

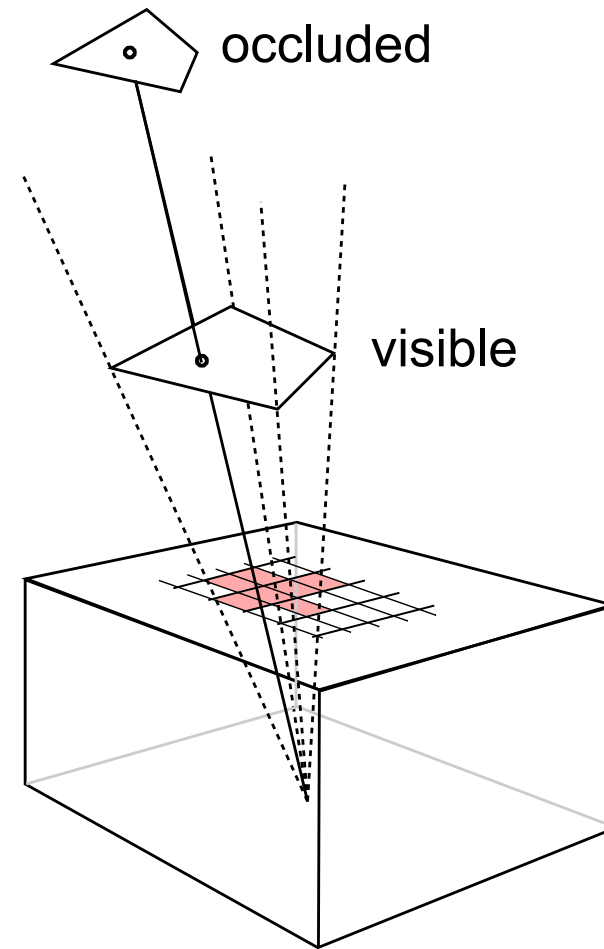
Projection of patches onto the hemicube

- We now need to know which patch is visible from each hemicube pixel.
- This could be done by ray tracing (casting), or projection.
- Ray tracing neatly solves the occlusion problem.
- Projection would require z-buffering.



Sum the pixels per patch

- Notice that all we need to determine is the nearest visible patch 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.



In summary

- Rendering with radiosity:
 1. Divide the graphics world into discrete patches
 2. Compute form factors by the hemi-cube method
 3. Solve the matrix equation for the radiosity of each patch.
 4. Average the radiosity values at the corners of each patch
 5. Compute a texture map of each point inside the patch
 6. Project to the viewing window and render with interpolation shading.

