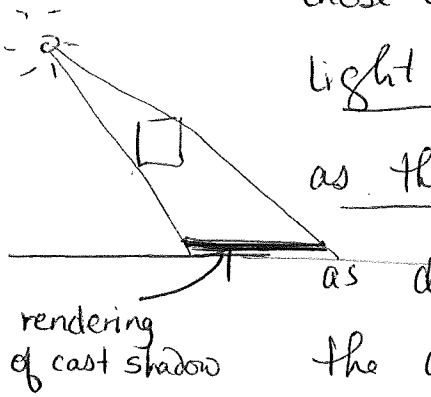
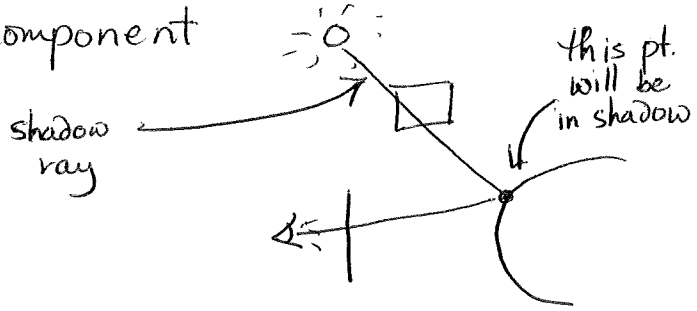
HW #2 Solutions

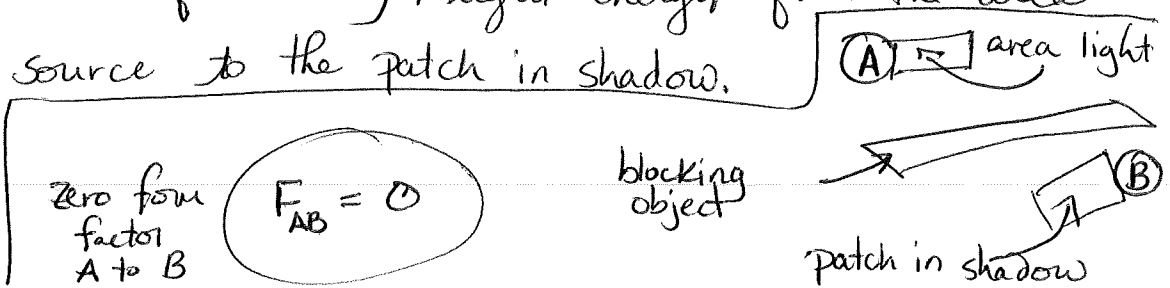
1.) (a) You must create them yourself. For example, to create shadows of objects onto a ground plane, render those objects with the camera positioned at the light source location and use the ground plane as the image plane. Place the rendered shadows as dark objects (triangles) located just above the ground plane so that they appear as shadows.



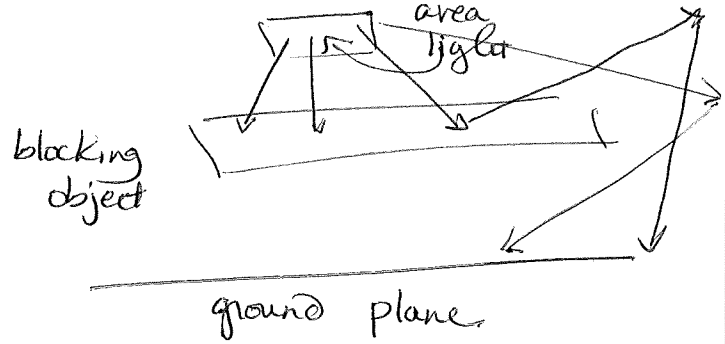
(b) In a ray tracer, cast shadow rays from each point to each light source to determine whether there is a direct illumination component



(c) In radiosity, shadows are captured through the form factors between patches. If a patch representing an area light in not visible to another patch, the form factor between the patches will be zero, and there will be no direct contribution of radiosity / light energy from the area light source to the patch in shadow.



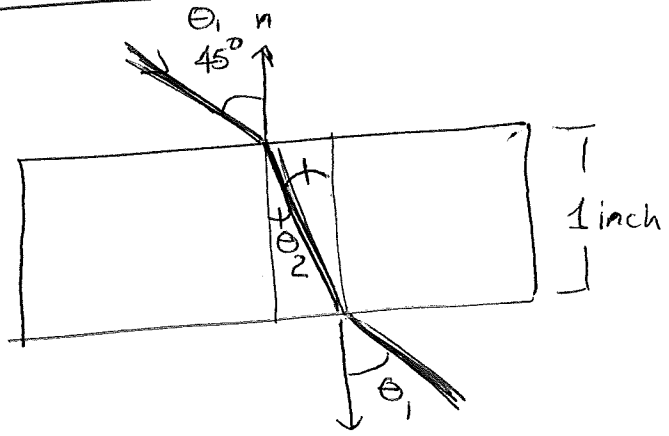
(d) In photon mapping, photons are traced outward from the light sources. If an area is in shadow, photons coming directly from the light source will not reach it, as they will be stopped by the blocking object. Only photons that have survived multiple bounces may be able to reach the shadowed area.



photons are stopped by the blocking object and must reach the ground plane indirectly

---

2.)



Use Snell's Law

$$\frac{\sin\theta_1}{\sin\theta_2} = \frac{n_2}{n_1}$$

n_1 = refractive index of air = 1.00

n_2 = refractive index of glass = 1.52 (crown glass)

$$\sin\theta_2 = \sin\theta_1 \left(n_1/n_2\right) = \sin\left(\pi/4\right)\left(\frac{1.00}{1.52}\right) = 0.46$$

$$\theta_2 = \sin^{-1}(0.46) = 0.48 \text{ radians} \approx 28^\circ$$

3.) Traverse tree front to back ⇒ call traverseNode(root, ray)

traverseNode(n, ray)

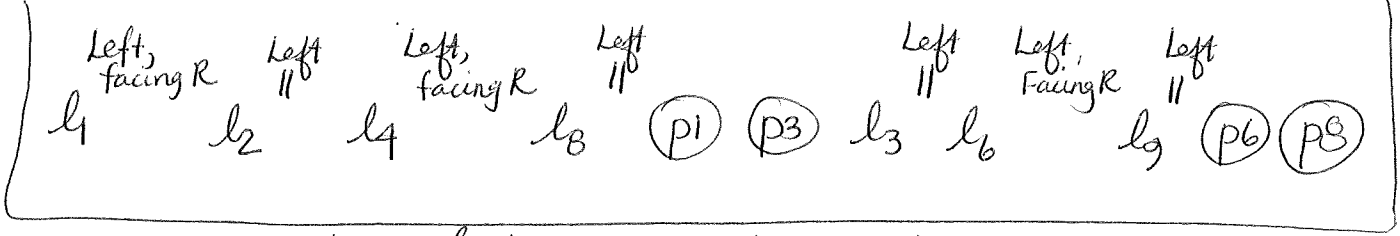
(if) n is a leaf
- (if) isIntersection(ray, n)
  - return intersection

(if) viewpoint is left of n
- intersection = traverseNode(n.leftChild, ray)
- (if) intersection return intersection
- (if) ray is towards n
  - intersection = traverseNode(n.rightChild, ray)
  - if intersection return intersection

(else)
- intersection = traverseNode(n.rightChild, ray)
- (if) intersection return intersection
- (if) ray is towards n
  - intersection = traverseNode(n.leftChild, ray)
  - if intersection return intersection

return false

(b)



we can skip the right branches of l_2, l_8, l_3, and l_9

4) Here is the system of equations we solve:

$$B_i = E_i + \rho_i \sum_j F_{i \to j} B_j$$

Radiosity for patch i is equal to radiosity <u>emitted</u> from patch i (E_i) plus radiosity <u>reflected</u> from patch i due to incident radiosity from all other patches j.

The equation represents an equilibrium in light energy bouncing around a scene from patch to patch to patch.

---

5) $B_1 = E_1 = 1$

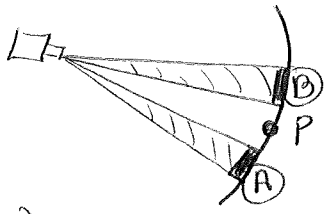
$$B_2 = \tfrac{1}{2} \sum_j F_{i \to j} B_j = \tfrac{1}{2}\left(\tfrac{1}{2} B_1 + \tfrac{1}{2} B_3\right) = \tfrac{1}{4} + \tfrac{1}{4} B_3$$

$$B_3 = \tfrac{1}{2} \sum_j F_{i \to j} B_j = \tfrac{1}{2}\left(\tfrac{1}{2} B_1 + \tfrac{1}{2} B_2\right) = \tfrac{1}{4} + \tfrac{1}{4} B_2$$

Solving this system, we find:

$$B_1 = 1$$
$$B_2 = B_3 = \tfrac{1}{3}$$

6) Separation of components requires projecting a pattern onto all surfaces and observing the illumination that results.



For example here, light at point p would include effects of subsurface scattering from illuminated regions (A) and (B).

This technique fails if the pattern of light projected is too coarse compared to distance traveled due to subsurface scattering (eg, if the distance is too great for scattered light from A & B to reach point P.)

7) My opinion —

Aligned with / resulting from physical effects:

- squash & stretch (to some extent)
- timing (somewhat - eg, result of impact)
- follow through & over lapping action (eg. efficient throwing of a baseball)
- slow in / slow out (limitations of our muscles)
- arcs (energy efficient motion at our joints)
- secondary action (passive physics)

Nothing to do with physics

- Anticipation (this is acting)
- Exaggeration (storytelling)
- Appeal (creative genius!)