Lecture 10 - Global Illumination (Chapter 21)

I. Global Illumination considers light reflected from other models, vs. local illumination which only considered objects interacting with light only (ch. 17).

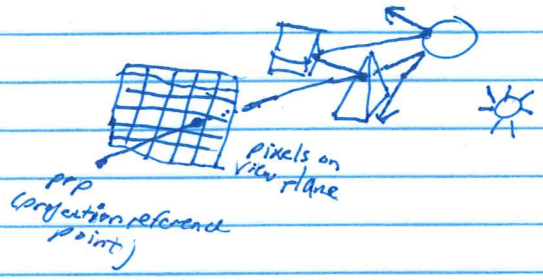
A. Ray Tracing - determine surface shading by following light rays from the camera's eye back into the scene through the pixels of the image plane

B. Radiosity - computes light intensities by propagating light through the scene as radiant energy or the flow of photons.

C. We will look at ray tracing in detail and discuss the idea of radiosity

II. Ray Tracing

A. Basic Ray-tracing Algorithm - detects visible surfaces, shadows, transparency, generates perspective views, and multiple light sources



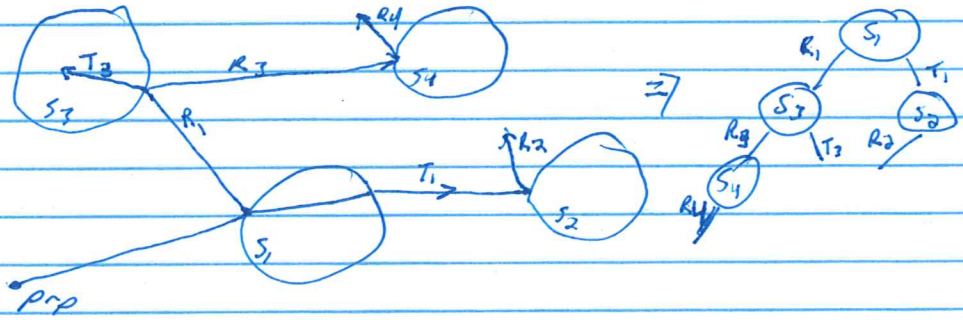
i. Start with view coordinates, so projection is down the z axis and pixels are on the xy view plane. This facilitates generating rays easily. Also keeps objects undistorted (perspective projection causes this)

ii. Propagate rays from the projection point through a pixel and into the scene. Branches happen at intersection (reflection and transmission) sites. Pixel intensities are accumulated along the paths to determine pixel colors.

a. Based on optics. One ray per pixel is like viewing scene from a pinhole camera.

b. We need to compute surface intersections to generate secondary rays with reflection and refraction

c. Generates a ray tracing tree. Left branch represents reflection, and right is refraction. Path continues until: (1) no surface intersected, (2) hits a light source, (3) maximum depth reached.



d. At each surface: invoke basic illumination model
- For rays not intersecting, choose background color
- For rays intersecting light source ($\vec{L}$) choose light intensity
- Send ray from intersection to light, if an intersection is detected only use the ambient light. Called the shadow ray.: $k_a I_a$.
- Otherwise $k_a I_a + k_d(\vec{N}\cdot\vec{L}) + k_s(\vec{H}\cdot\vec{N})^{n_s}$ where $\vec{H}$ is halfway between view ($-\vec{U}$) and light ($\vec{L}$). $\vec{U}$ is ray.

e. Secondary Rays
- Specular reflection: $\vec{R} = \vec{U} - (2\vec{U}\cdot\vec{N})\vec{N}$
- Transmission (refraction): $\vec{T} = \frac{n_i}{n_r}\vec{U} - (\cos\theta_r - \frac{n_i}{n_r}\cos\theta_i)\vec{N}$

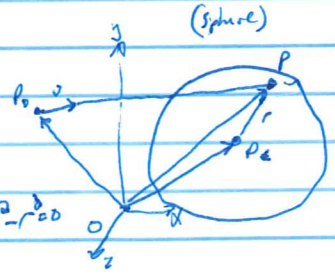
where $\cos\theta_r = \sqrt{1-(\frac{n_i}{n_r})^2(1-\cos^2\theta_i)}$

f. Accumulation - after tree is computed. Post-order traversal (start at leaves) each color of a node is attenuated to the parent node and summed to the intensity of parent surface (can include textures). Pixel is color attenuated from root of the tree. (or background color)

iii. Discussion *Ask class*
a. Ray tracing is highly view dependent - rays are traced from camera. If the view changes (or the scene changes) rays must be retraced.
b. Collision detection can be 95% of ray tracing cost.
c. Highly parallelizable algorithm - processor per pixel (Still not real-time)

B. Intersection Computation
i. Rays have a position $\vec{P_0}$ and unit direction vector $\vec{U}$
any point on ray is $\vec{P} = \vec{P_0} + s\vec{U}$ for a value s.
a. P_0 chosen as pixel position $\vec{P_{pix}}$ initially with $\vec{U} = \frac{\vec{P_{pix}} - \vec{P_{eye}}}{|\vec{P_{pix}} - \vec{P_{eye}}|}$
b. goal is to solve for $\vec{P}$ (and s)
c. Secondary rays generated from $\vec{P}$ with $\vec{U}$ as $\vec{R}$ or $\vec{T}$

ii. Ray-Sphere computations
a. Surface of sphere is all points $\vec{P}$: $|\vec{P}-\vec{P_c}|^2 - r^2 = 0$
b. Substitute ray eq. for $\vec{P}$: $|\vec{P_0}+s\vec{U}-\vec{P_c}|^2 - r^2 = 0$
c. $\Delta P = P_c - P_0$ and expand dot product: $s^2 - 2(\vec{U}\cdot\vec{\Delta P})s + |\Delta P|^2 - r^2 = 0$
Solve: $s = \vec{U}\cdot\vec{\Delta P} \pm \sqrt{(\vec{U}\cdot\vec{\Delta P})^2 - |\Delta P|^2 + r^2}$



if discriminant is negative either no intersection or sphere is behind ray.
d. Choose smaller s and solve for $\vec{P}$

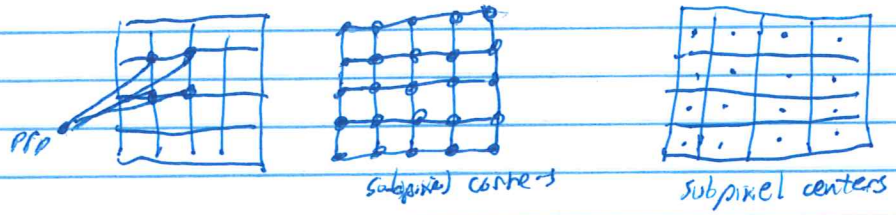
iii. Ray-Polyhedron - more complex. Use Barycentric coordinates for triangles. Can exploit bounding volumes + hierarchies + orientations of faces, etc.

Discuss

C. Anti-aliasing in Ray-Tracing

i. Supersampling – use multiple evenly spaced rays over each pixel area either through subpixel corners or subpixel centers. Average pixel intensities for pixel color.

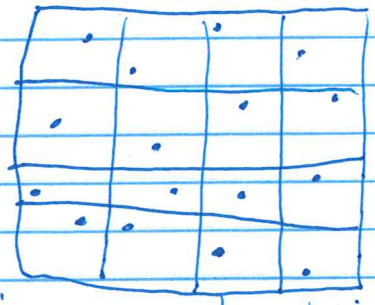
ii. Adaptive sampling – use unevenly spaced rays, cluster more toward object boundaries or add more rays in supersampling if a small object is missed.



iii. Cast cones into scene instead of rays. Possibility

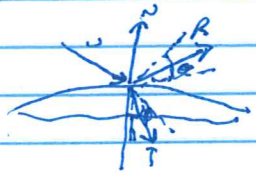
D. Distributed (Distribution) Ray tracing – stochastically sampled rays create low-level noise that replaces aliasing effects.

i. Randomly distribute rays over pixel area or one random in each subpixel (jittering)



intensities are averaged to obtain result.

ii. During reflection and refraction, randomly distributed rays are used as well.



iii. Shadow ray computations – distribute as well. When some rays hit the light and others do not a penumbra (partially illuminated) region is created. Only for extended light sources, not point sources.

iv. Can be used for camera focus + motion blur.

III. Radiosity Lighting Model.

A. So far lighting has mainly ignored interactions between surfaces. Ray tracing was able to incorporate specular lighting between surfaces, but not diffuse interactions. Radiosity solves this problem for more realistic effects, eg. soft shadows. *show pictures*

B. Idea is that light scatters from hitting an object (diffuse) and then goes to other objects, i.e., propagates. Each surface has a form factor with any other surface.

C. ~~Model as a linear equation~~
Break scene into small surface patches and model radiosity (diffuse interactions) as a linear system of equations. The solution gives the diffuse color of each surface patch.

D. Usually solved through iterative refinement or progressive refinement (propagate a few patches of light so first pass is light after one diffuse "bounce", etc. *show pictures*

E. View independent *Ask class why?*
but not all encompassing (specular missing)