RADIOSITY: DAY 1 NOTES

Computer graphics is the process of starting from a scene, which contains information like:

- Which surfaces (objects) are in the scene
- Light sources
- Material properties (reflectivity, color, etc.)
- Location of the viewer

And rendering a realistic depiction of that scene to an image. One common format for scenes is an OBJ file, which describes:

- A list of vertices, with their 3D coordinates (and possibly colors).
- A list of faces (usually triangles), in terms of which vertices they contain.

OBJ files specify color per-vertex, but we usually think of it as per-face.

1. How to render such a scene?

A scene contains a bunch of triangles (3D coordinates for each vertex), which need to be projected onto the 2D screen. Mathematically, the viewer can be described with a location and a direction of sight, and the screen is a plane perpendicular to that line of sight some distance away from the viewer. For each vertex in the scene, draw a line from it to the viewer. Its projection is where that line intersects the screen plane.

Once vertices are projected, the lines between them can be filled in by drawing lines between the 2D screen coordinates. But how to fill in the triangles?

We could fill in each triangle with the color specified by the scene. But then every triangle with the same color would appear exactly the same, and a gray ball would look flat. So what do we do?

2. Lighting!

Consider the wall of a room. It's probably all one uniform color, but the physical wall doesn't look uniform. Rather, the parts nearer light sources look brighter than the parts further away. An accurate rendering of that wall shouldn't be uniform either.

3. Standard Graphics Techniques

Colors can be represented by the amount of red, green, and blue in them. One approach to assign colors to triangles is to give each a triple (K^r, K^g, K^b) for the reflectance in each color. In this interpretation, a triangle with $K^g = 0.5$ would reflect half the green light that hits it, while one with $K^b = 0$ would reflect none of the blue light.

A scene could be filled with ambient light, which affects all triangles equally. A light source can also be assigned red, green, and blue components, called (E_a^r, E_a^g, E_a^b). Under ambient light, a triangle's rendered color would be:

$$(K^r E_a^r, K^g E_a^g, K^b E_a^b)$$

This sort of lighting is just the uniform material strategy, but with the option to have different light colors. We need more.

Another layer that can be added on top is diffuse lighting. This type can be emitted by a point source at a particular position, with color (E_d^r, E_d^g, E_d^b). This sort of light should be brighter on surfaces facing the light source and darker (or entirely removed) on surfaces facing away. We usually represent the direction a given triangle faces by an outward-pointing normal vector $\hat{n}$, which is the unit vector perpendicular to the surface. Let $\vec{v}$ be the vector from the center of the triangle to the light source, and let θ be the angle between $\hat{n}$ and $\vec{v}$. When these vectors are perfectly aligned, we want maximal diffuse light, and when they're misaligned we want less. A convenient function that does what we want is $\cos(\theta)$, and we can add diffuse light to our lighting formula as:

$$\left(K^r \left(E_a^r + E_d^r \cos(\theta)\right), K^g \left(E_a^g + E_d^g \cos(\theta)\right), K^b \left(E_a^b + E_d^b \cos(\theta)\right)\right)$$

If there are multiple diffuse light sources, we can add more terms onto this equation.

One drawback of diffuse light is that it makes objects look dull. One option to add shininess is specular lighting, which mimics the effect of a mirror. If the viewer's position lines up just right with a light source's location, the object will look much brighter because the light reflects directly at the viewer rather than being absorbed and re-emitted from the surface. Given the vectors $\hat{n}$ and $\vec{v}$, we can compute the direction $\hat{r}$ that a perfect reflection off that triangle would go. We can also compute the vector $\vec{e}$ from the center of the face to the viewer's location. If the angle ϕ between $\hat{r}$ and $\vec{e}$ is small, the surface should appear much brighter. If the scene is composed of enough small triangles, it won't be noticeable that the bright spots are made up of triangles. We can use $\cos(\phi)^\alpha$ as a function which is close to 1 when ϕ is close to 0 and falls off quickly if α is large. Adding this term to the red component of the color gives:

$$K^r \left(E_a^r + E_d^r \cos(\theta) + E_s^r \cos(\phi)^\alpha\right)$$

There are many tricks that can make these effects give better results. For instance, these colors can be computed on each vertex (and then interpolated to the rest of each triangle), which gives smoother results (Gouraud shading). These effects aren't necessarily physically realistic, but they produce good-looking results.

One more realistic option is raytracing. For each pixel on the screen, a ray can be drawn from the viewer through the screen at that location. That ray can be extended until it hits an object, where it may reflect onto another object, and so on. The color of each object along the way can be computed in a physically realistic way, and is good for scenes with lots of shiny objects. However, it takes a lot of computing.

4. Radiosity

The approach we'll look at in this course is radiosity, which is mostly used when most surfaces are diffuse (e.g. architectural rendering). It's good at accurately producing effects that other methods struggle to produce, like physically correct shadows and color bleeding (a brightly illuminated colored object tints nearby objects). Our goal is to take a scene and some light sources, and use the radiosity method to figure out exactly which color each triangle should appear.

So how do we do it?

5. Radiative Transport Theory

We'll end up solving an equation to figure out what color each triangle should be. We'll derive that equation through radiative transport theory, which describes how particles flow through a medium. That'll give us the Rendering Equation, which we can solve numerically (by a computer, approximately).

We'll use $X \subset \mathbb{R}^3$ as the *host medium*. In rendering, X might be a room with some furniture in it. The boundary of X (denoted ∂X) is the points on the edge of X, which includes the edge of the room and its furniture. We'll investigate the flow of particles through a general X.

Each particle (for rendering, a photon) can be represented with a location $\vec{x}$ and a direction $\vec{\omega}$. Both aspects are important for rendering; ultimately, only the ones that are heading towards the viewer will affect the final scene. No other aspects (e.g. polarization, interference) will be considered.

Denote the unit sphere ($x^2 + y^2 + z^2 = 1$) by S^2. This set describes all possible directions. Let $\Omega \subseteq S^2$, and let $\omega \in \Omega$. We may also consider a subset $V \subseteq X$. Then the state of a given photon in V and traveling in a direction in Ω is in $V \times \Omega$.

6. Steradians (solid angles)

Given $V \subseteq X$, its volume is straightforward to compute and think about. But what about $\Omega \subseteq S^2$? Rather than thinking about a sphere, consider a contiguous subset of a circle. We usually measure the size of these subsets in radians, which are the swept-out arc length divided by the radius of the circle. Moving the analogy up to S^2, we can define the *steradian* measure of Ω as the surface area of Ω on the sphere divided by the square of the radius of the sphere. The largest possible steradian measure is then 4π: the ratio of surface area to radius for a sphere.

7. Back to Radiative Transport Theory

We will assume:

- All particles are completely characterized by position $\in X$ and direction $\in S^2$.
- All particles travel at the same speed.
- Particles do not interact with each other.
- Particles may interact with the host medium, by being absorbed or scattered.
- The distribution of particles is in equilibrium at all times; that is, on average, the particles leaving any region are balanced by particles entering.