CSC418
Lecture Notes

Yuchen Wang

May 29, 2020

Contents

1 Ray Tracing

1.1 Shading

Notation 1.1. The important variables in light reflection are unit vectors
Light direction $\mathbf{l}$: a unit vector pointing toward the light source;
View direction $\mathbf{v}$: a unit vector pointing toward the eye or camera;
Surface normal $\mathbf{n}$: a unit vector perpendicular to the surface at the point where reflection is taking place.

1.1.1 Lambertian Shading

An observation by Lambert in the 18th century: the amount of energy from a light source that falls on an area of surface depends on the angle of the surface to the light.

Definition 1.1 (Lambertian shading model)**.** The vector $\mathbf{l}$ is computed by subtracting the intersection point of the ray and the surface from the light source position.
The pixel color

$$L = k_d I \max(0, \mathbf{n} \cdot \mathbf{l})$$

where k_d is the *diffuse coefficient*, or the surface color; and I is the intensity of the light source.

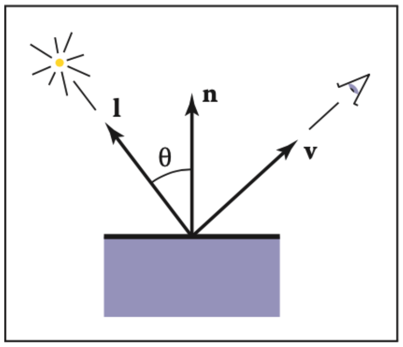


Figure 1: Geometry for Lambertian shading

Remark 1.1. Because $\mathbf{n}$ and $\mathbf{l}$ are unit vectors, we can use $\mathbf{n} \cdot \mathbf{l}$ as a convenient shorthand for $\cos\theta$. This equation applies separately to the three color channels.

Remark 1.2. Lambertian shading is *view independent:* the color of a surface does not depend on the direction from which you look. Therefore it does not produce any highlights and leads to a very matte, chalky appearance.

1.1.2 Blinn-Phong Shading

A very simple and widely used model for specular highlights by Phong (1975) and J.F.Blinn (1976).

Idea Produce reflection that is at its brightest when $\mathbf{v}$ and $\mathbf{l}$ are symmetrically positioned across the surface normal, which is when mirror reflection would occur; reflection then decreases smoothly as the vectors move away from a mirror configuration.
Compare the half vector $\mathbf{h}$ with $\mathbf{n}$: if $\mathbf{h}$ is near the surface normal, the specular component should be bright and vice versa.

Definition 1.2 (Blinn-Phong shading model)**.**

$$\mathbf{h} = \frac{\mathbf{v}+\mathbf{l}}{||\mathbf{v}+\mathbf{l}||}$$
$$L = k_d I \max(0, \mathbf{n}\cdot\mathbf{l}) + k_s I \max(0, \mathbf{n}\cdot\mathbf{h})^p$$

where k_s is the *specular coefficient*, or the specular color of the surface, and $p > 1$.

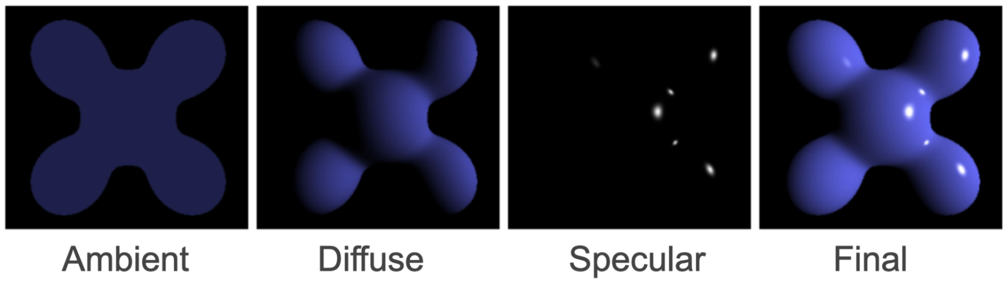
1.1.3 Ambient Shading

A heuristic to avoid black shadows is to add a constant component to the shading model, one whose contribution to the pixel color depends only on the object hit, with no dependence on the surface geometry at all, as if surfaces were illuminated by ambient light that comes equally from everywhere.

Definition 1.3 (simple shading model / Blinn-Phong model with ambient shading)**.**

$$L = k_a I_a + k_d I \max(0, \mathbf{n}\cdot\mathbf{l}) + k_s I \max(0, \mathbf{n}\cdot\mathbf{h})^p$$

where k_a is the surface's ambient coefficient or "ambient color", and I_a is the ambient light intensity.



1.1.4 Multiple Point Lights

Property 1.1 (superposition)**.** The effect by more than one light source is simply the sum of the effects of the light sources individually.

Definition 1.4 (extended simple shading model)**.**

$$L = k_a I_a + \sum_{i=1}^{N} [k_d I_i \max(0, \mathbf{n}\cdot\mathbf{l}_i) + k_s I_i \max(0, \mathbf{n}\cdot\mathbf{h}_i)^p]$$

where $I_i, \mathbf{l}_i$ and $\mathbf{h}_i$ are the intensity, direction, and half vector of the i-th light source.

1.2 A Ray-Tracing Program

```
for each pixel do
    compute viewing ray
    if (ray hits an object with $t \in [0, \infty)$) then
        Compute $\mathbf{n}$
        Evaluate shading model and set pixel to that color
    else
        set pixel color to background color
```

1.3 Shadows

Recall from 1.1 that light comes from direction $\mathbf{l}$. If we imagine ourselves at a point $\mathbf{p}$ on a surface being shaded, the point is in shadow if we "look" in direction $\mathbf{l}$ and see an object. If there are no objects, then the light is not blocked.

```
function raycolor( ray $\mathbf{e} + t\mathbf{d}$, real $t_0$, real $t_1$ )
hit-record rec, srec
if (scene→hit($\mathbf{e} + t\mathbf{d}$, $t_0$, $t_1$, rec)) then
    $\mathbf{p} = \mathbf{e} + (\text{rec}.t)\,\mathbf{d}$
    color $c = \text{rec}.k_a\ I_a$
    if (not scene→hit($\mathbf{p} + s\mathbf{l}$, $\epsilon$, $\infty$, srec)) then
        vector3 $\mathbf{h}$ = normalized(normalized($\mathbf{l}$) + normalized($-\mathbf{d}$))
        $c = c + \text{rec}.k_d\, I \max(0, \text{rec}.\mathbf{n} \cdot \mathbf{l}) + (\text{rec}.k_s)\, I\, (\text{rec}.\mathbf{n} \cdot \mathbf{h})^{\text{rec}.p}$
    return $c$
else
    return background-color
```

Remark 1.3. The usual adjustment to avoid the problem of intersecting $\mathbf{p}$ with the surface that generates it is to make the shadow ray check for $t \in [\epsilon, \infty)$ where ϵ is some small positive constant.

Remark 1.4. The code above assumes that $\mathbf{d}$ and $\mathbf{l}$ are not necessarily unit vectors.

1.4 Ideal Specular Reflection

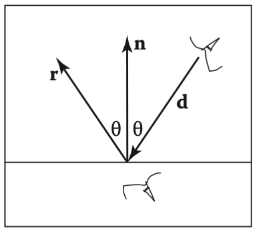


Figure 2: When looking into a perfect mirror, the viewer looking in direction **d** will see whatever the viewer "below" the surface would see in direction **r**

$$\mathbf{r} = \mathbf{d} - 2(\mathbf{d} \cdot \mathbf{n})\mathbf{n}$$

In the real world, a surface may reflect some colors more efficiently than others, so it shifts the colors of the objects it reflects (i.e. some energy is lost when the light reflects from the surface).
We implement the reflection by a recursive call of *raycolor*:

```
color c = c + km * raycolor(p + sr, epsilon, max_t);
```

where k_m is the specular RGB color, p is the intersection of the viewing ray and the surface, $s \in [\epsilon, max\_t)$ for the same reason as we did with shadow rays: we don't want the reflection ray to hit the object that generates it.
To make sure that the recursive call will terminate, we need to add a maximum recursion depth.