1 Calculating normals to faces

Look at your hand or a flat object. Depending on how it is oriented it looks bigger or smaller. If you look at it "on edge", you hardly see any area at all.

An object looks largest when its perpendicular vector (call it n) points right at you. In other words, it points in the same direction as v, the vector to the viewer.

What vector is perpendicular to a triangle abc?

Answer: $w = (b - a) \times (c - a)$.

What is the length $|w|$ of w? It is twice the area of the triangle.

Suppose $a = (0, 0, 0), b = (b_x, 0, 0), c = (c_x, c_y, 0)$. Then $w = (0, 0, b_x c_y)$. If you draw the triangle, that's the base times the height, which is twice the area. This still holds true if you rotate it.

Suppose w points right at you, in the direction of v. Suppose $|v| = 1$. What is $v \cdot w$ (the dot product)? It's $|v||w| \cos\theta$ where θ is the angle, which is zero. So it's $|w|$.

Now suppose w is perpendicular to v, so we are looking at the triangle edge on. Then $w \cdot v = 0$. So if w is pointing straight at us or w is perpendicular to v, $w \cdot v$ is twice the area that we see. It turns out that $w \cdot v$ is always twice the area that we see. If it's negative, we are looking at the "back" of the triangle.

This is the cosine rule. The area you see is $\cos(\theta)$ times the actual area, where θ is the angle between the perpendicular n and the direction to the viewer v.

Now suppose you are looking at a polygon p_0, p_1, p_2, p_4, p_5. These points may not be exactly in a plane. That's the reason a table wobbles: the ends of the legs are not in a plane. What is the perpendicular? It is reasonable to define the perpendicular to be the direction to the viewer who sees the biggest area.

What is the area that the viewer in direction v sees? Break the polygon into triangles p_0, p_1, p_2, p_0, p_2, p_3, p_0, p_3, p_4, p_0, p_4, p_5. Call the cross products for these w_{012}, w_{023}, w_{034}, and w_{045}. The total area you see (times two) is $v \cdot w_{012} + v \cdot w_{023} + v \cdot w_{034} + v \cdot w_{045}$. But the dot product distributes, so that $v \cdot (w_{012} + w_{023} + w_{034} + w_{045})$. When is that largest? When v points in the same direction as $n = w_{012} + w_{023} + w_{034} + w_{045}$, which is to say

$v = (w_{012} + w_{023} + w_{034} + w_{045})/|w_{012} + w_{023} + w_{034} + w_{045}|$

2 Phong Reflectance Model

So if an object has perpendicular n and you are in direction v, the fraction of the area you see is $v \cdot n$. Suppose you were a light source. The area that receives light is multiplied times $v \cdot n$. If v and n point in the same direction, $v \cdot n = 1$ and it receives maximum light. If v and n are perpendicular, $v \cdot n = 0$ and it receives no light because the light "sees" it edge on.

So far we are talking about how much light a surfaces RECEIVES. How much does it EMIT

and in what direction?

2.1 Diffuse Reflection

A LAMBERTIAN surface is rough, not shiny. It takes all the light it receives and sends it back out in all directions equally. So the color of a Lambertian surface is

$$k_d I_d (n \cdot l)$$

where l is the unit vector pointing at the light source, n is the unit vector perpendicular to the surface, k_d is the reflectance of the surface, and I_d is the color of the light source. The d stands for DIFFUSE reflection.

You can think of k_d and I_d as vectors. $k_d = (1, 0, 0)$ is a red surface. $I_d = (0, 0, 1)$ is a blue light. Multiplying is done component by component. (The times method of PV does that.) So even if $n \cdot l = 1$, $k_d I_d n \cdot l$ will be $(0, 0, 0)$ in this case. But that's right. A red object in a blue light looks black.

NOTE: the formula only works "during the day". If the light is behind the surface, it doesn't light it at all. How can you tell?

2.2 Specular Reflection

Another kind of surface is shiny and polished, with a mirror being the ultimate. In that case, the light bounces off in just one direction. What direction? We know that angle of reflection equals the angle of incidence. I will draw a diagram that shows that r must be $-l + sn$ for some s. What s? Well, the angle rule tells us that the cosine will be the same too, so $l \cdot n = r \cdot n$

$l \cdot n = (-l + sn) \cdot n$

$l \cdot n = -l \cdot n + sn \cdot n$

$l \cdot n = -l \cdot n + s$

$s = 2(l \cdot n)$

So $r = 2(l \cdot n)n - l$.

Most surfaces are somewhere between perfectly rough (Lambertian) and perfectly shiny. So there is a bright spot if v is close to r. v is close to r if $v \cdot r$ is close to 1. We want the spot to be dark if $v \cdot r$ is not so near to 1. The way to do this is use an exponent:

$k_s I_s (v \cdot r)^\alpha$

where k_s is the surfaces's specular reflectance, I_s is the light's specular color (which usual is the same as I_d), and the shininess α depends on the material. The book tells you what value look plastic, not so shiny, or metal, much more shiny.

2.3 Ambient Reflection

Finally, most of the light misses the surface and bounces off the walls and other objects of the room. That light gets to the surface from all directions and is colored by the room. This is called the AMBIENT light. It is simply

$k_a I_a$

because it does not depend on l or v.

So the total color of a point on a surface is

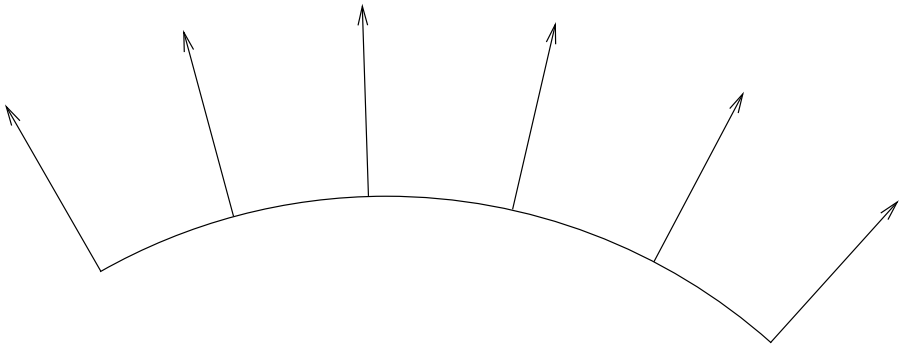
$k_d I_d (n \cdot l) + k_s I_s (v \cdot r)^\alpha + k_a I_a$

2.4 Choosing the surface reflectances and colors of light

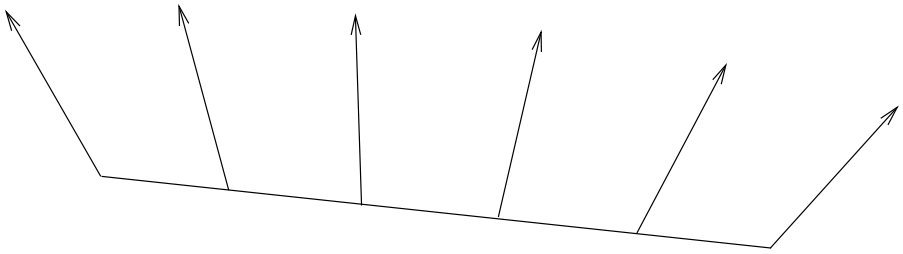
What values for the k's and I's? k_d is the color of the object. When you wax and polish something, the wax fills the roughness and has a smooth surface. So specular reflection doesn't "pick up" the color and k_s should be white or a shade of grey. Ambient light comes from all directions so very little is reflected specularly, so $k_a = k_d$. $I_d = I_s$ is the light source. Ambient light I_a is the light source times an average of all the k_d in the room.

3 Phong *Shading*

I could not find a web page that explained the genius behind Phong *shading*. Let's think in 2D for a while. The curve shown here has varying normals along its length. The vectors are pendicular to the curve at their base points. We use the Phong *reflectance* formulas above to generate the colors. The amazing thing is that our brains can figure out the shape from the shading. So if the color is generated using the formulas above, we see the shape as it is.



Now suppose we move the base points so they lie on a line segment as shown below, but still keep the "normal vectors" the same and generate the colors using the formulas. What do we see? Maybe the colors are shifted in position a tiny bit, but they are determined by the vectors, so we see the same colors in almost the same positions. So naturally, we see the *same* shape as in the first figure. A flat shape cannot have varying normals. So the combination of "normals" and positions in the second figures are *impossible*. You can't expect our brains to say that that slight distortion means that its a flat shape with impossible normals. We see the "correct" curved shape in the first figure with a little distortion that we ignore.



Now suppose we send just the two endpoint base points and their two vectors to the vertex shader using attribute variables vPosition and vNormal and the vertex shader sends these to the fragement shader using varying variables fNormal and fPosition. These are automatically interpolated. The graphics cards generates the rest of the positions and normals by interpolation when they get to the fragment shader. So what the fragment shader gets is the vectors and their base points as shown in the second figure. There is a little more distortion because the curve is not a line. Again our eyes disregard it.

So instead of sending a point and vector for maybe 100 pixels along this line segment, we send only two of each. But in 3D, the line segment is a triangle, so we have to send *one more* point and vector, but the number of internal points and vectors we avoid sending is more like $100^2 = 10000$. This is a huge saving in time to transfer data.

SUMMARY: the graphics card plus our brains convert the last figure into the first figure. And the benefit is *squared* in 3D.