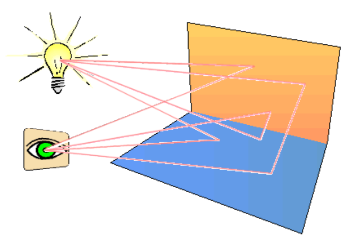
Illumination and Shading - Part 1

- Light Sources
- Empirical Illumination
- Shading
- Transforming Normals
- Project #4

Illumination Models

Computer Graphics Jargon:

- **Illumination** - the transport of luminous flux from light sources between points via direct and indirect paths
- **Lighting** - the process of computing the luminous intensity reflected from a specified 3-D point
- **Shading** - the process of assigning colors to pixels



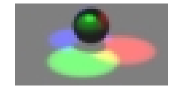
Illumination Models:

- Empirical - simple formulations that approximate observed phenomenon
- Physically-based - models based on the actual physics of light's interactions with matter

Two Components of Illumination

Light Sources

- Emittance Spectrum (color)
- Geometry (position and direction)
- Directional Attenuation



Surface Properties

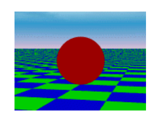
- Reflectance Spectrum (color)
- Geometry (position, orientation, and micro-structure)
- Absorption

Simplifications used by most computer graphics systems:

- Only the direct illumination from the emitters to the reflectors of the scene
- Ignore the geometry of light emitters, and consider only the geometry of reflectors

Ambient Light Source

Even though an object in a scene is not directly lit it will still be visible. This is because light is reflected indirectly from nearby objects. A simple *hack* that is commonly used to model this indirect illumination is to use of an *ambient light source*.



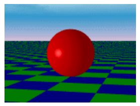
Ambient light has no spatial or directional characteristics. The amount of ambient light incident on each object is a constant for all surfaces in the scene. An ambient light can have a color.

The amount of ambient light that is reflected by an object is independent of the object's position or orientation. Surface properties are used to determine how much ambient light is reflected.

Directional Light Sources

All of the rays from a directional light source have a common direction, and no point of origin. It is as if the light source was infinitely far away from the surface that it is illuminating.

Sunlight is an example of an infinite light source.

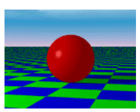


The direction from a surface to a light source is important for computing the light reflected from the surface. With a directional light source this direction is a constant for every surface.

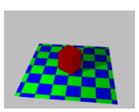
A directional light source can be colored.

Point Light Sources

The rays emitted from a point light radially diverge from the source. A point light source is a fair approximation to a local light source such as a light bulb.



The direction of the light to each point on a surface changes when a point light source is used. Thus, a normalized vector to the light emitter must be computed for each point that is illuminated.

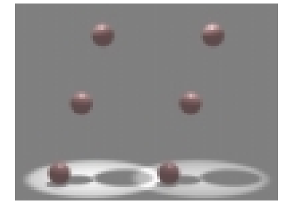


$$\vec{d} = \frac{\dot{p} - \dot{l}}{\left\| \dot{p} - \dot{l} \right\|}$$

Other Light Sources

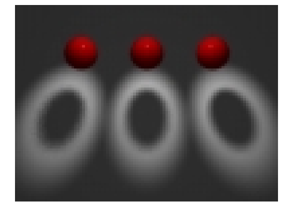
Spotlights

- Point source whose intensity falls off away from a given direction
- Requires a color, a point, a direction, parameters that control the rate of fall off



Area Light Sources

- Light source occupies a 2-D area (usually a polygon or disk)
- Generates *soft* shadows

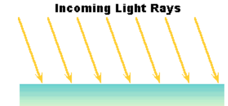


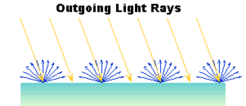
Extended Light Sources

- Spherical Light Source
- Generates *soft* shadows

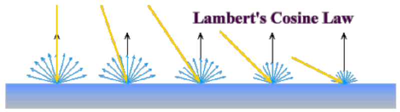
Ideal Diffuse Reflection

First, we will consider a particular type of surface called an *ideal diffuse reflector*. An ideal diffuse surface is, at the microscopic level, a very rough surface. Chalk is a good approximation to an ideal diffuse surface. Because of the microscopic variations in the surface, an incoming ray of light is equally likely to be reflected in any direction over the hemisphere.

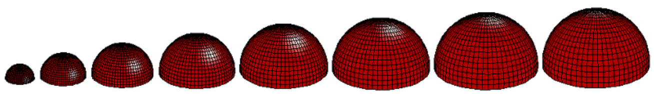




Lambert's Cosine Law



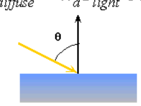
Ideal diffuse reflectors reflect light according to *Lambert's cosine law*, (these are sometimes called Lambertian reflectors). Lambert's law states that the reflected energy from a small surface area in a particular direction is proportional to the cosine of the angle between that direction and the surface normal. Lambert's law determines how much of the *incoming* light energy is reflected. Remember that the amount of energy that is reflected in any one direction is constant in this model. In other words, the reflected intensity is **independent of the viewing direction**. The intensity does, however, depend on the light source's orientation relative to the surface, and it is this property that is governed by Lambert's law.



Computing Diffuse Reflection

The angle between the surface normal and the incoming light ray is called the angle of incidence and we can express a intensity of the light in terms of this angle.

$$I_{diffuse} = k_d I_{light} \cos\theta$$



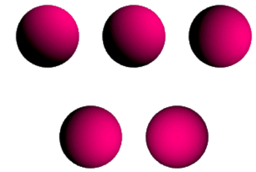
The I_{light} term represents the intensity of the incoming light at the particular wavelength (the wavelength determines the light's color). The k_d term represents the diffuse reflectivity of the surface at that wavelength.

In practice we use vector analysis to compute cosine term indirectly. If both the *normal vector* and the incoming *light vector* are normalized (unit length) then diffuse shading can be computed as follows:

$$I_{diffuse} = k_d I_{light} \left(\overline{n} \cdot \overline{l}\right)$$

Diffuse Lighting Examples

We need only consider angles from 0 to 90 degrees. Greater angles (where the dot product is negative) are blocked by the surface, and the reflected energy is 0. Below are several examples of a spherical diffuse reflector with a varying lighting angles.



Why do you think spheres are used as examples when shading?

Specular Reflection

A second surface type is called a *specular reflector*. When we look at a shiny surface, such as polished metal or a glossy car finish, we see a highlight, or bright spot. Where this bright spot appears on the surface is a function of where the surface is seen from. This type of reflectance is view dependent.

At the microscopic level a specular reflecting surface is very smooth, and usually these microscopic surface elements are oriented in the same direction as the surface itself. Specular reflection is merely the *mirror reflection* of the light source in a surface. Thus it should come as no surprise that it is viewer dependent, since if you stood in front of a mirror and placed your finger over the refelection of a light, you would expect that you could reposition your head to look around your finger and see the light again. An ideal mirror is a purely specular reflector.

In order to model specular reflection we need to understand the physics of reflection.

Snell's Law

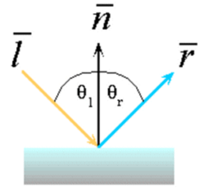
Reflection behaves according to Snell's law which states:

- The incoming ray, the surface normal, and the reflected ray all lie in a common plane.

$$n_l \sin\theta_l = n_r \sin\theta_r$$

- The angle that the reflected ray forms with the surface normal is determined by the angle that the incoming ray forms with the surface normal, and the relative speeds of light of the mediums in which the incident and reflected rays propogate according to the following expression.

  (Note: n_l and n_r are the indices of refraction)

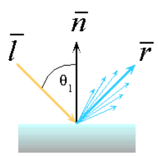


Reflection is a very special case of Snell's Law where the incident light's medium and the reflected rays medium is the same. Thus we can simplify the expression to:

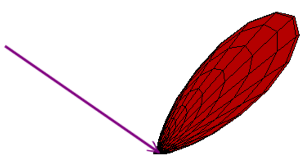
$$\theta_l = \theta_r$$

Non-ideal Reflectors

Snell's law, however, applies only to ideal *mirror* reflectors. Real materials, other than mirrors and chrome tend to deviate significantly from ideal reflectors. *At this point we will introduce an empirical model that is consistent with our experience, at least to a crude approximation.*



In general, we expect most of the reflected light to travel in the direction of the ideal ray. However, because of microscopic surface variations we might expect some of the light to be reflected just slightly offset from the ideal reflected ray. As we move farther and farther, in the angular sense, from the reflected ray we expect to see less light reflected.

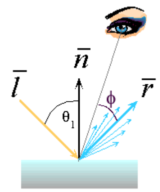


Phong Illumination

One function that approximates this fall off is called the *Phong Illumination* model. This model has no physical basis, yet it is one of the most commonly used illumination models in computer graphics.

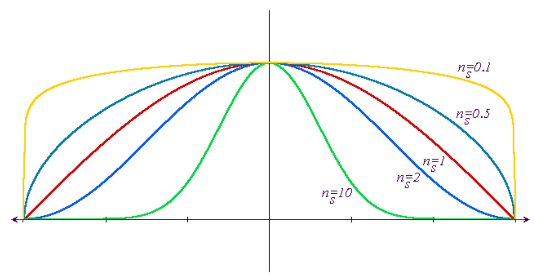
$$I_{specular} = I_{light}(\cos\phi)^{n_{shiney}}$$

The *cos* ϕ term is maximum when the surface is viewed from the mirror direction and falls off to 0 when viewed at 90 degrees away from it. The n_{shiny} term controls the rate of this fall off.



Effect of n_{shiney}

The diagram below shows the how the Phong reflectance drops off based on the viewer's angle from the reflected ray for various values of n_{shiny}.



Computing Phong Illumination

The *cos* term of Phong's specular illumination using the following relationship.

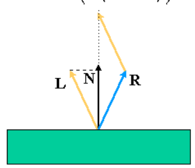
$$I_{specular} = k_s I_{light} \left(\hat{V} \cdot \hat{R}\right)^{n_{shiny}}$$

The V vector is the unit vector in the direction of the viewer and the R vector is the mirror reflectance direction. The vector R can be computed from the incoming light direction and the surface normal as shown below.

$$\hat{R} = \left(2\left(\hat{N} \cdot \hat{L}\right)\right)\hat{N} - \hat{L}$$

The following figure illustrates this relationship.

$$\hat{R} + \hat{L} = \left(2\left(\hat{N} \cdot \hat{L}\right)\right)\hat{N}$$



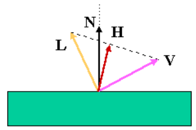
Blinn & Torrance Variation

Jim Blinn introduced another approach for computing Phong-like illumination based on the work of Ken Torrance. His illumination function uses the following equation:

$$I_{specular} = k_s I_{light} \left(\hat{N} \cdot \hat{H}\right)^{n_{shiney}}$$

In this equation the angle of specular dispersion is computed by how far the surface's normal is from a vector bisecting the incoming light direction and the viewing direction.

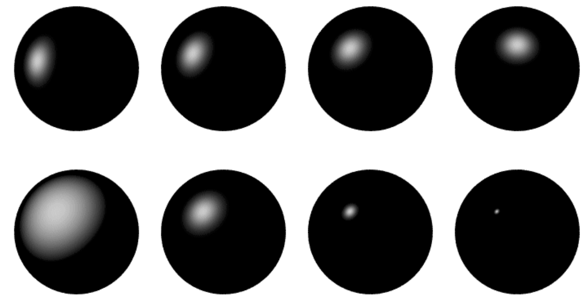
$$\hat{H} = \frac{\hat{L} + \hat{V}}{\left|\hat{L} + \hat{V}\right|}$$



On your own you should consider how this approach and the previous one differ.

Phong Examples

The following spheres illustrate specular reflections as the direction of the light source and the coefficient of shineyness is varied.



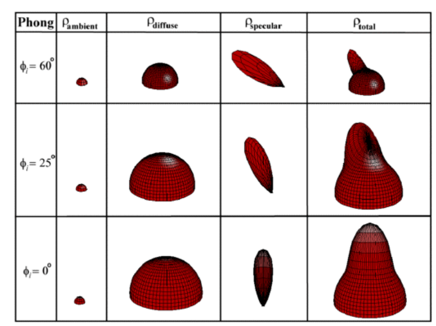
Putting it all together

Our final empirical illumination model is:

$$I_{total} = k_a I_{ambient} + \sum_{i=1}^{lights} I_i \left(k_d\left(\hat{N} \cdot \hat{L}\right) + k_s\left(\hat{V} \cdot \hat{R}\right)^{n_{shiney}}\right)$$

Notes:

- The *Phong Lighting* model
- Once per light
- Once per color component
- Reflectance coefficients, k_a, k_d, and k_s may or may not vary with the color component
  If they do, you need to be careful.



Where do we Illuminate?

To this point we have discussed how to compute an illumination model at a point on a surface. But, at which points on the surface is the illumination model applied? Where and how often it is applied has a noticable effect on the result.

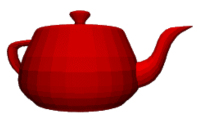
Illuminating can be a costly process involving the computation of and normalizing of vectors to multiple light sources and the viewer.

For models defined by collections of polygonal facets or triangles:

- Each facet has a common surface normal
- If the light is directional then the diffuse contribution is constant across the facet
- If the eye is infinitely far away and the light is directional then the specular contribution is constant across the facet

Flat Shading

The simplest shading method applies only one illumination calculation for each primitive. This technique is called *constant* or *flat shading*. It is often used on polygonal primitives.



Issues:

- For point light sources, the direction to the light source varies over the facet
- For specular reflections, the direction to the eye varies over the facet

Nonetheless, often illumination is computed for only a single point on the facet. Which one? Usually the centroid. For a convex facet the centriod is given as follows:

$$centroid = \frac{1}{vertices} \sum_{i=1}^{vertices} \overline{p}_i$$

Facet Shading

Even when the illumination equation is applied at each point of the faceted nature of the polygonal nature is still apparent.

To overcome this limitation normals are introduced at each vertex.

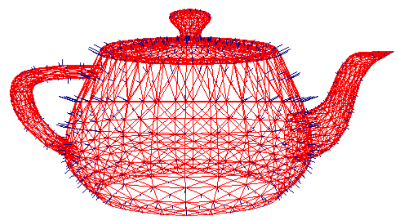
- Usually different than the polygon normal
- Used only for shading (not backface culling or other geometric computations)
- Better approximates the "real" surface
  - Assumes the polygons were a piecewise approximation of the *real* surface (C^0)
  - Normals provide information about the tangent plane at each point (C^1)

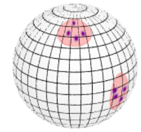
Vertex Normals

If vertex normals are not provided they can often be approximated by averaging the normals of the facets which share the vertex.

$$\overline{n}_v = \sum_{i=1}^{k} \frac{\overline{n}_i}{|\overline{n}_i|}$$

This only works if the polygons *reasonably* approximate the underlying surface.





A better approximation can be found using a clustering analysis of the normals on the unit sphere.

Gouraud Shading

The *Gouraud Shading* shading method applies the illumination model on a subset of surface points and interpolates the intensity of the remaining points on the surface. In the case of a polygonal mesh the illumination model is usually applied at each vertex and the colors in the triangles interior are linearly interpolateded from these vertex values.

Hit Ctrl-Shift-S to rotate teapot

The linear interpolation can be accomplished using the plane equation method discussed in the lecture on rasterizing polygons.

Notice that facet artifacts are still visible.

Phong Shading

In Phong shading (not to be confused with Phong's illumination model), the surface normal is linearly interpolated across polygonal facets, and the Illumination model is applied at every point.

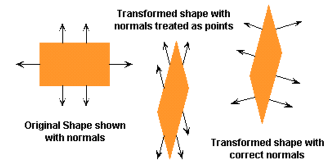
A Phong shader assumes the same input as a Gouraud shader, which means that it expects a normal for every vertex. The illumination model is applied at every point on the surface being rendered, where the normal at each point is the result of linearly interpolating the vertex normals defined at each vertex of the triangle.

Phong shading will usually result in a very smooth appearance, however, evidence of the polygonal model can usually be seen along silhouettes.

Transforming Surface Normals

By now you realize that surface normals are the most important geometric surface characteristic used in computing illumination models. They are used in computing both the diffuse and specular components of reflection.

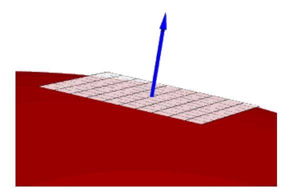
However, the vertices of a model do not transform in the same way that surface normals do. A naive implementer might consider transforming normals by treating them as points offset a unit length from the surface. But even this approach will not work. Consider the following two dimensional example.



The problem with transforming normals occurs when objects undergo an *anisotropic* scaling (scaling that is not uniform in all directions). Such scaling results only from affine modeling transforms (Why not Euclidean transforms?).

Normals Represent Tangent Spaces

The fundamental problem with transforming normals is largely a product of our mental model of what a normal really is. A normal is not a geometric property relating to points of the surface, like a quill on a porcupine. Instead normals represent geometric properties on the surface. They are an implicit representation of the tangent space of the surface at a point.



In three dimensions the tangent space at a point is a plane. A plane can be represented by either two basis vectors, but such a representation is not unique. The set of vectors orthogonal to such a plane is, however unique and this vector is what we use to represent the tangent space, and we call it a normal.

Triangle Normals

Now that we understand the geometric implications of a normal it is easy to figure out how to transform them.

On a faceted planar surface, vectors in the tangent plane can be computed using surface points as follows.

$$\bar{t}_1 = \bar{p}_1 - \bar{p}_0 \quad \bar{t}_2 = \bar{p}_2 - \bar{p}_0$$

Normals are always orthogonal to the tangent space at a point. Thus, given two tangent vectors we can compute the normal as follows:

$$\bar{n} = \bar{t}_1 \times \bar{t}_2$$

This normal is perpendicular to both of these tangent vectors.

$$\bar{n} \cdot \bar{t}_1 = \bar{n} \cdot \bar{t}_2 = 0$$

Transforming Tangents

The following expression shows an affine transformation of a tangent vector.

$$\mathbf{A} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\bar{t}_1' = \mathbf{A}\bar{t}_1 = \mathbf{A}(\bar{p}_1 - \bar{p}_0)$$

If we multiply this out we get

$$\bar{t}_1' = \mathbf{A}'\bar{t}_1$$

$$\mathbf{A}' = \begin{bmatrix} a & b & c \\ e & f & g \\ i & j & k \end{bmatrix}$$

Transforming Normals

This transformed tangent, *t'*, must be perpendicular to the transformed normal, *n'*. Let's solve for the transformation, **Q** that maintains this relationship.

$$\bar{n}' \cdot \bar{t}' = (\mathbf{Q}\bar{n})^T \mathbf{A}'\bar{t} = \bar{n}^T \mathbf{Q}^T \mathbf{A}'\bar{t} = 0$$

All that we need to do is find a value for the **Q** matrix so that the transformed normal and the transformed tangent are still orthogonal. This can be accomplished by letting

$$\mathbf{Q}^T = \mathbf{A}'^{-1}$$

which gives

$$\bar{n}' \cdot \bar{t}' = \bar{n}^T \mathbf{A}'^{-1} \mathbf{A}'\bar{t} = \bar{n}^T \bar{t} = \bar{n} \cdot \bar{t} = 0$$

Thus the transform that must be applied to normals so that they remain perpendicular to the tangent space of transformed points is:

$$\bar{n}' = \mathbf{Q}\bar{n} = \left(\mathbf{A}'^{-1}\right)^T \bar{n}$$

Is there a class of matrices where $(\mathbf{A}'^{-1})^T = \mathbf{A}$?

Normals of Nonplanar Surfaces

Not all surfaces are given as planar facets. A common example of such a surface is called a parametric surface. For a parametric surface the three-space coordinates are determined by functions of two parameters, *u* and *v* in our case.

$$S = \begin{bmatrix} X(u,v) \\ Y(u,v) \\ Z(u,v) \end{bmatrix}$$

For parametric surfaces two vectors in the tangent plane can be found by computing partial derivitives as follows.

$$\bar{t}_1 = \left[\frac{\partial X(u,v)}{\partial u}, \frac{\partial Y(u,v)}{\partial u}, \frac{\partial Z(u,v)}{\partial u}\right]$$

$$\bar{t}_2 = \left[\frac{\partial X(u,v)}{\partial v}, \frac{\partial Y(u,v)}{\partial v}, \frac{\partial Z(u,v)}{\partial v}\right]$$

And the normal is computed as before:

$$\bar{n} = \bar{t}_1 \times \bar{t}_2$$

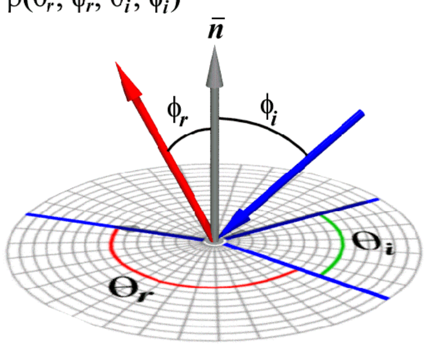
Discuss Project #4

Rule Changes:

- Project is due midnight Monday (11/20/00)
- You are required only to implement an ambient and a directional light source
- Parser for input files:
    - **Comments** - # the rest of the line after a pound sign is ignored
    - **Eye Position** - eye *x y z*
    - **Look-at Position** - look *x y z*
    - **World-space up vector** - up *x y z*
    - **Horizontal field-of-view** - fov *angle_in_degrees*
    - **Ambient Light Source** - la *r g b* # color intensities range from 0 to 1
    - **Directional Light Source** - ld *r g b x y z* # x y z is a vector from the light to the surface
    - **Polygon Vertex** - v *x y z*
    - **Polygon Facet** - f i_1 i_2 i_3 i_4 ... i_n # index to vertex 0-based
    - **Surface Parameters** - surf r g b k_a k_d k_s n_{shiney}

Next Time

$\rho(\theta_r, \phi_r, \theta_i, \phi_i)$



Physically-Based Illumination Models