Lecture 7: Rendering Monochrome Shaded Opaque Objects

If a raster-scan device has the capability of generating a specified light intensity at each pixel, then true shaded pictures can be displayed. First we shall discuss the problems associated with determining the required intensities for a single colour only. In the next lecture we shall examine the problems associated with the generation of coloured images and a complete, realistic 3D scene. For the time being we will still confine our discussion to objects made up of planar faces. We may approximate complex objects by using a large number of these planar facets. The analysis that we carry out will be equally appropriate for objects defined mathematically, such as spheres cylinders and higher order smooth surfaces.

The illumination models described below are derived from physical laws, but as will be pointed out later, exact laws sometimes produce very poor, unrealistic images. There are many reasons for this. First, the images are displayed on a given display device, which has its own physical characteristics. Secondly, the results will always be approximate, and the eye is so perceptive and sensitive that the errors made by the approximations are actually accentuated. Thirdly, there is a large psychological factor in "seeing". Looking at the two-dimensional surface of the television screen, we accept a 3D view as real. When we look at the same scene in a newspaper, we accept it as the 2D representation of a 3D scene. Hence, many of the methods developed for rendering are combinations of: (a) physical laws, (b) properties that can be calculated within a reasonable time period, and (c) what "looks good" to the viewer.

Computer Generated Images of Illuminated Objects

An object which does not produce light itself absorbs, reflects, and/or transmits light when it is illuminated by other light sources. This is what makes objects visible. How we see an object depends on the number, position, and properties of the light sources, the light absorbing, transmitting and reflecting properties of the objects, and the respective positions of the light sources, the objects, and the viewer. Generally however reflected light can be approximated by a combination of diffuse and specular components.

Diffuse Reflection

Diffuse reflection is a main characteristic of matt surfaces. The physical explanation for diffuse reflection is that light is absorbed by the surface and only some of the wavelengths are readmitted. The readmitted light occurs equally in all directions; therefore, the light intensity of the reflected light is independent of the location of the viewer. Since the absorbed energy per surface area is a function of the incident light direction, the reflected intensity depends on the angle between the incident light and the normal to the surface. The physical law for diffused reflection from a surface illuminated by a point light source is:

$$I_{reflected}(\lambda) = I_i(\lambda)\ k_d(\lambda)\ \mathrm{Cos}(\theta)$$

where Ii(λ) is the incident light intensity as a function of wavelength (colour), kd(λ) is the diffuse reflection coefficient which is also a function of wavelength, and θ is the angle between the incident light and the normal vector to the surface. It is known as Lambert's cosine law. The position of the viewer does not enter into the equations. If the direction vector between the light source and the object is called ***s***, the direction normal to the surface is ***n***, and we normalise both vectors so that both have unit length, then the same equation may be expressed by the dot product of these two vectors:

$$I_{reflected}(\lambda) = Ii(\lambda)\ kd(\lambda)\ (\boldsymbol{n} \bullet \boldsymbol{s})$$

The law is illustrated by Diagram 7.1 for a single light source. Since the amount of reflected light is assumed to be proportional to the incident light the reflected light from a number of point sources is calculated by the sum of individual reflected intensities. Since in this lecture we are only concerned with monochrome images we can drop the wavelength (λ) from our equations.

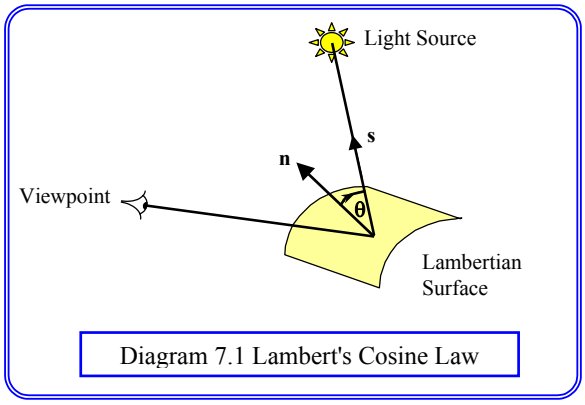


Diagram 7.1 Lambert's Cosine Law

Ambient Light

In scenes which contain many objects, reflected light goes through multiple reflections between the objects. It would be impossible to calculate exactly the effect of these multiple reflections. It is assumed that the multiple reflections from multiple light sources and opaque objects produce the ambient light intensity, a background, constant light intensity which has no direction. The fraction of ambient light reflected from a

surface may be expressed by the constant ka. The total diffusely reflected light intensity from both the ambient light and a point light source is given by:

$$I_{reflected} = Ia\ ka + Ii\ kd\ (\boldsymbol{n} \bullet \boldsymbol{s})$$

In many cases it is sufficient to use a single constant for the product Ia ka, which is chosen empirically. In a later lecture we shall see there is a technique, called radiosity, for modelling ambient light more accurately.

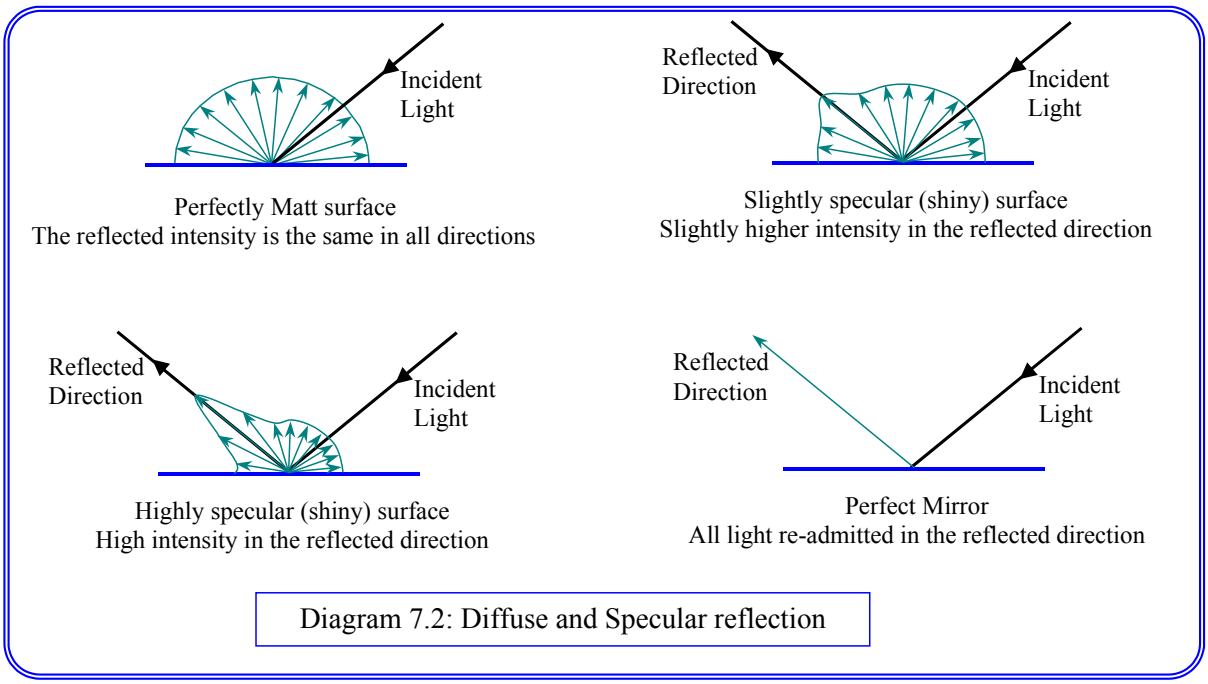


Diagram 7.2: Diffuse and Specular reflection

Specular Reflection

Specular reflection is the characteristic of shiny surfaces. A perfect mirror reflects the incident light from a point source only in one direction. In this case the angle of reflection is equal to the angle of incidence, (θ); furthermore, the incident light ray, the reflected light ray, and the normal vector are all in one plane.

Shiny surfaces reflect light in all directions, but the amount of light reflected is a function of the position of the viewer. The largest amount of light is reflected in the direction of the reflection vector, **r**. Typical patterns for reflected light intensity are shown in diagram 7.2. The reflected light intensity is a function of the angle (α) between the direction of the viewer, **v**, and the vector **r**, as shown in diagram 7.3.

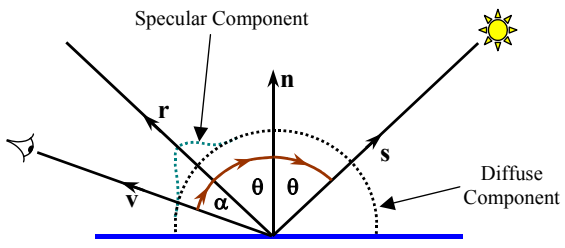


Diagram 7.3
Calculating Specular Reflections

It is possible to make reasonably accurate physical models of specular reflection, but an empirical formula proposed by Bui-Tuong Phong has gained universal acceptance.

$$I_{reflected} = Ii\ f(\theta)\ \cos^{t}(\alpha)$$

where Ii is the incident light intensity, f(θ) is a function that depends on the angle between the incident light ray and the surface normal, and the value of *t* determines the shininess of the surface. The larger the value of t the more shiny the surface. As a first approximation, we may assume that the function f(θ) is a constant. The strength of this formula is that it is easy to calculate, and that it produces reasonably realistic 3D images. For simplicity we take the incident light term Ii to be the same as that for the diffuse equation. Some systems, such as openGL allow you to specify different incident light intensities for each.

So far, we have used Ii for the incident light intensity, indicating that it is a constant. For point sources very far away, like the sun, this is a reasonable approximation. The incident light intensity from a point source decreases as $1/d^2$, where d is the distance between the object and the point source. When researchers used this

physical law to generate pictures they found that the inverse square law produced too much contrast. The intensity range of the display could not express such large intensity variations when the object was either too close or too far from the light source. Another empirical law was found to be useful. It was found that the function 1/(K+d) produced a much more acceptable picture than the physical inverse square law. Again, K is an arbitrary constant that is determined experimentally after the scene is generated. Hence, a good starting point for a simple illumination model may be the following equation:

$$I_{reflected} = Ia\ ka + Ii\ [kd\ (\boldsymbol{n} \bullet \boldsymbol{s}) + ks\ (\boldsymbol{v} \bullet \boldsymbol{r})^t]/(K+d)$$

where the dot products represent the cosine of the angles. This model still ignores the fact that physical light sources are not strictly point sources. The incorporation of realistic light sources into the above equation makes the incident light intensity also a function of the respective positions and angle of incidence between the light source and the surface. Another modification of the above equation may be the variation of ambient light intensity as a function of the distance between the scene and the source of illumination. Also, the above equation is given for one light source only. If there are several light sources, then the sum of the reflected light intensities must be used.

Flat Shading

The simplest way to apply shading to graphical scenes is to take the above equations and use them to determine the brightness at each point on each visible facet. This is referred to as flat shading. If one were to use close light sources and specular reflections then this would be an expensive process. Usually it is carried out with only diffuse and ambient light, and the assumption that each light source is sufficiently far from the face for the reflected light intensity to be constant everywhere on it. Thus it is only necessary, for quick computation, to compute the shade value at one point and apply it to the whole polygon.

This gives us a fast way of shading objects, however, we still have a problem in representing smooth surfaces. The effect of discontinuities in shading between the facets is accentuated by the eye, and becomes very pronounced. This is a well known property of our eye, which is called the Mach band effect. To get round this problem we use interpolation shading. There are two standard methods known as Gouraud shading and Phong shading.

Gouraud Shading

Interpolation shading is done by calculating the appropriate shades at each polygon vertex, and then for each face using linear interpolation to determine the shade values at each pixel. We will consider the case of objects built from triangular faces. This does not loose us any generality since planar polygons can always be divided into triangles. At each vertex we compute the intensity in three dimensions using the above formulae. The facet is projected, and in 2D we have the intensities given at the projected 2D vertices. For each triangle there are three projected vertices, **P1** to **P3** and for each a reflected light intensity, I1 to I3. For any given point inside the triangle the intensity is evaluated by linear interpolation. Any point in the triangle can be defined using:

$$\mathbf{P} = \mathbf{P1} + \mu1(\mathbf{P2} - \mathbf{P1}) + \mu2(\mathbf{P3}-\mathbf{P1})$$

and the intensity is given by

$$I = I1 + \mu1(I2 - I1) + \mu2(I3-I1)$$

Where $\mu1$ and $\mu2$ are scalar paramters. Inside the triangle we have that $0<=\mu1 + \mu2<1$, $0<=\mu1<1$, $0<=\mu2<1$. It would be possible, given a point in the triangle, to solve for $\mu1$ and $\mu2$ and then calculate I, however in practice it is far faster to use a scan line algorithm as illustrated in diagram 7.4.

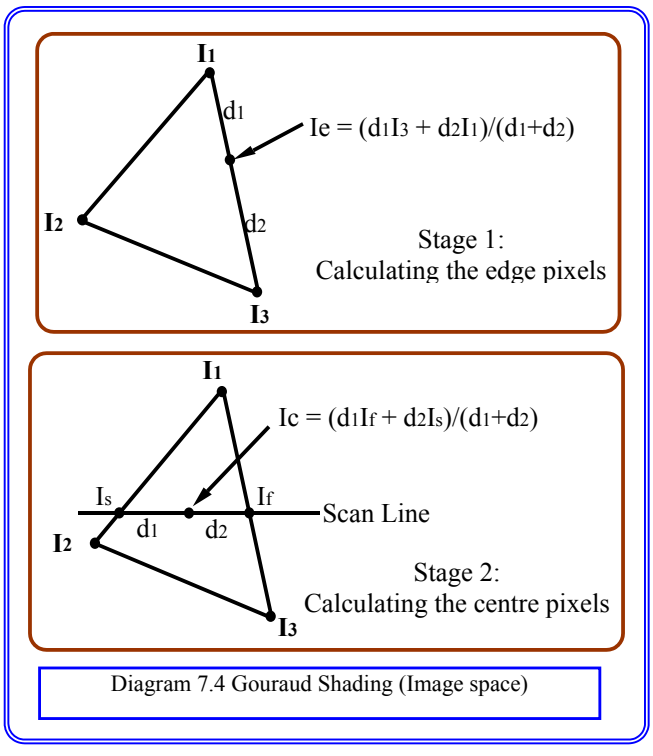


Diagram 7.4 Gouraud Shading (Image space)

Interpolating shades over many polygons.

In many cases it is desirable to represent a smooth surface by many polygons. We can remove the visual effect of the polygon boundaries by interpolating over them. We do this by calculating an average normal vector in 3D at each vertex as shown in diagram 7.5. This is the average of all the normal vectors of the triangles meeting at that vertex that belong to the same surface. Where adjacent triangles do not belong to the same surface they must be treated separately to preserve the discontinuity between different surfaces. Once the normal vectors are known, the reflected light intensities are calculated for each vertex, and the 2D calculations can proceed as before.

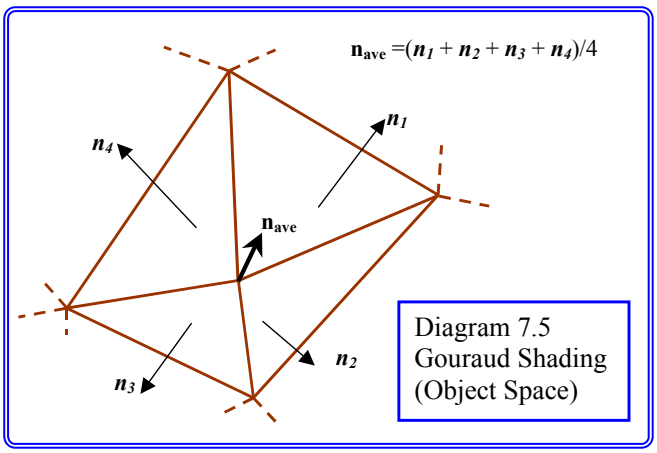


Diagram 7.5
Gouraud Shading
(Object Space)

Phong Shading

Phong shading works by interpolating the surface normals across a polygon. Hence it can be used to produce effects such as specular highlights. However, it has to be calculated in the three dimensional object space. Again, taking triangular facets, consider one with vertices, **V1** to **V3** and unit normal vectors ***n1*** to ***n3*** at the vertices. These normals are calculated by the same method outlined for Gouraud shading. We may express any point for this facet in parametric form:

$$\mathbf{P} = \mathbf{V1} + \mu 1(\mathbf{V2} - \mathbf{V1}) + \mu 2(\mathbf{V3} - \mathbf{V1})$$

The average normal vector at the same point may be calculated as the vector **a**:

$$\mathbf{a} = \boldsymbol{n1} + \mu 1(\boldsymbol{n2} - \boldsymbol{n1}) + \mu 2(\boldsymbol{n3} - \boldsymbol{n1})$$

and then

$$\mathbf{n}_{average} = \mathbf{a} / |\mathbf{a}|$$

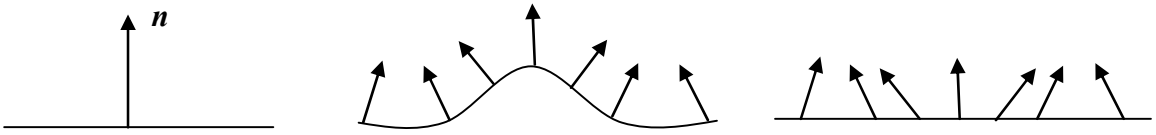
since the normal vector is assumed to have unit length in the above equations.

In practice, the normal interpolation calculations are carried out in two dimensions, even though this is not quite accurate. Using a scan line algorithm, as for Gouraud shading, it takes three times as long to calculate the surface normal vector at each point (since it has three components). If specularities are to be shown, it is then necessary to revert to three dimensions to calculate the reflected vector. Although computationally expensive, Phong shading gives much better results and minimises the Mach band effect.

There are more complex illumination models and methods for generating realistic scenes of 3D objects than the ones presented here. These are beyond the scope of this course.

Bump Mapping

Complex detail can be added to the rendering of a smooth surface by perturbing the normals before applying the shading algorithm. The idea is similar to texture mapping, except instead of specifying the intensities that create a visual pattern we specify the normals. When the surface is rendered the eye is fooled by the shading pattern into perceiving surface irregularities.



Planar Facet

Bump Map

Normals used for rendering