

1988

NEAR EAST UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

GRADUATION PROJECT
(COM 400)

PROJECT TITLE : ILLUMINATION MODELS &
SURFACE RENDERING
METHODS IN COMPUTER GRAPHICS

SUBMITTED TO : MEHRDAD KHALEDI

SUBMITTED BY : LALE BEHNAZ ERSAN

CONTENTS



| | |
|---|----|
| PROFECT TABLE | 1 |
| WHAT 'S AN ILLUMINATION MODEL | 4 |
| - PHOTOREALIZM IN COMPUTER GRAPHICS | |
| - WHAT EFFECTS ILLUMINATION MODEL | |
| LIGHT SOURCES | 5 |
| LIGHT EMITTING AND REFLECTING SOURCES | 6 |
| -DIFFUSE REFLECTION | 7 |
| -SPECULAR REFLECTION | |
| BASIC ILLUMINATION MODELS | 8 |
| -AMBIENT LIGHT | |
| DIFFUSE REFLECTION | |
| SPECULAR REFLECTION AND PHONG MODEL | 12 |
| COMBINED DIFFUSE AND SPECULAR REFLECTIONS | 16 |
| WITH MULTIPLE LIGHT SOURCES | |
| WARN MODEL | 17 |
| FLOPS | 19 |
| SPOTLIGHTS | |
| INTENSITY ATTENUATION | |
| COLOR CONSIDERATIONS | 20 |
| TRANSPARENCY | 22 |
| SHADOWS | 25 |
| DISPLAYING LIGHT INTENSITIES | |
| ASSIGNING INTENSITY LEVELS | |
| GAMMA CORRECTION AND VIDEO LOOK UP TABLES | 28 |
| DISPLAYING CONTINUOS- TONE IMAGES | 31 |
| HALF TONE PATTERNS AND DITHERING TECHNIQUES | |
| HALF TONE APPROXIMATIONS | 32 |
| DITHERING TECHNIQUES | 35 |

CONTENTS

| | |
|---|----|
| POLYGON-RENDERING METHODS | 37 |
| CONSTANT INTENSITY SHADING | |
| GOURAUD SHADING | 38 |
| PHONG SHADING | 39 |
| FAST PHONG SHADING | |
| RAY-TRACING METHODS | 40 |
| BASIC RAY- TRACING ALGORITHM | |
| RAY-SURFACE INTERSECTION CALCULATIONS | |
| REDUCING OBJECT-INTERSECTION CALCULATIONS | 41 |
| SPACE-SUBDIVISION METHODS | 42 |
| ANTIALIASED RAY TRACING | 43 |
| DISTRIBUTED RAY TRACING | |
| RADIOSITY LIGHTING MODEL | 47 |
| BASIC RADIOSITY MODEL | |
| PROGRESSIVE REFINEMENT | |
| RADIOSITY METHOD | |
| ENVIRONMENT MAPPING | 49 |
| ADDING SURFACE DETAIL WITH POLYGONS | 50 |
| TEXTURE MAPPING | 51 |
| PROCEDURAL TEXTURING | |
| METHODS | |
| BUMP MAPPING | 52 |
| FRAME MAPPING | |
| CONCLUSION | 53 |
| REFERENCES | 56 |

WHAT'S AN ILLUMINATION MODEL ?

An *illumination model*, also called a *lighting model* and sometimes referred to as a *shading model*, is used to calculate the intensity of light that we should see at a given point on the surface of an object.

A *surface-rendering algorithm* uses the intensity calculations from an illumination model to determine the light intensity for all projected pixel positions for the various surfaces in a scene.

Surface rendering can be performed by applying the illumination model to every visible surface point, or the rendering can be accomplished by interpolating intensities across the surfaces from a small set of illumination-model calculations. Realistic displays of a scene are obtained by generating perspective projections of objects and by applying natural lighting effects to the visible surfaces.

Scan-line, image-space algorithms typically use interpolation schemes, while ray-tracing algorithms invoke the illumination model at each pixel position. Sometimes, surface-rendering procedures are termed *surface shading models*. To avoid confusion, we will refer to the model for calculating light intensity at a single surface point as an illumination model or a lighting model, and we will use the term surface rendering to mean a procedure for applying a lighting model to obtain pixel intensities for all the projected surface positions in a scene.

Photorealism in computer graphics involves two elements:

- 1- Accurate graphical representations of objects and
- 2-Good physical descriptions of the lighting effects in a scene.

Lighting effects include light reflections, transparency, surface texture, and shadows. Modeling the colors and lighting effects that we see on an object is a complex process involving both physics and psychology. Fundamentally, lighting effects are described with models that consider the interaction of electromagnetic energy with object surfaces. Once light reaches our eyes, it triggers perception processes that determine what we actually "see" in a scene.

What effects on physical illumination models ?

Physical illumination models involve a number of factors, such as object type, object position relative to light sources and other objects, and the light-source conditions that we set for a scene. Objects can be constructed of opaque materials, or they can be more or less transparent. In addition, they can have shiny or dull surfaces, and they can have a variety of surface-texture patterns. Light sources, of varying shapes, colors, and positions, can be used

to provide the illumination effects for a scene. Given the parameters for the optical properties of surfaces, the relative positions of the surfaces in a scene, the color and positions of the light sources, and the position and orientation of the viewing plane, illumination models calculate the intensity projected from a particular surface point in a specified viewing direction.

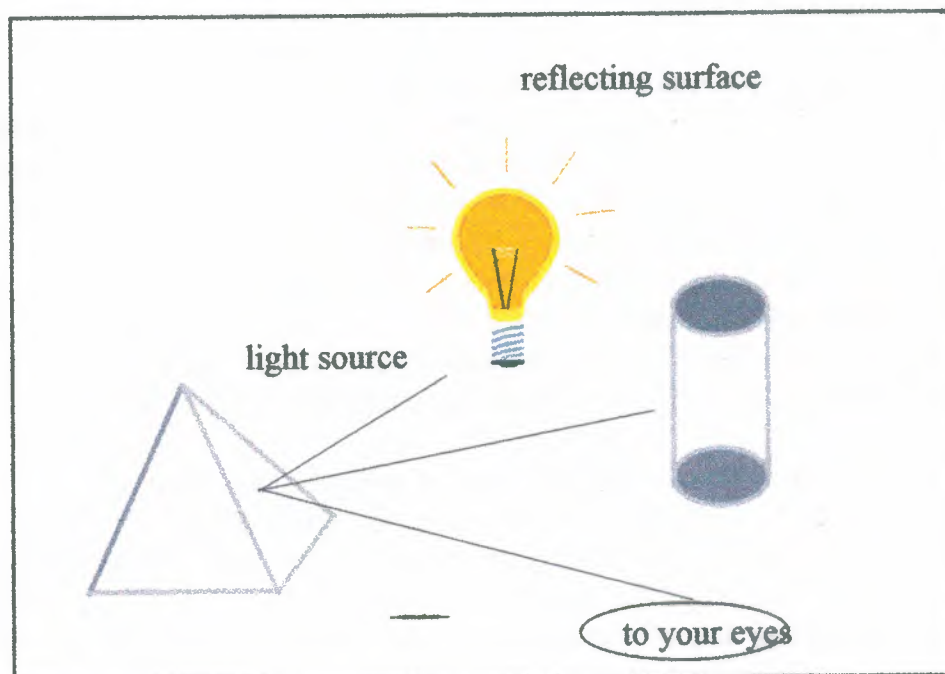
Illumination models in computer graphics are often loosely derived from the physical laws that describe surface light intensities. To minimize intensity calculations most packages use empirical models, such as the radiosity algorithm, calculate light intensities by considering the propagation of radiant energy between the surfaces and light sources in a scene.

In the following sections I will explain the basic illumination models often used in graphic packages and calculating surface intensities methods accurately. Then various surface rendering algorithms for applying the lighting models to obtain the appropriate shading over visible surfaces in a scene will be examined.

LIGHT SOURCES

The total reflected light of an opaque nonluminous object is the sum of the contributions from light sources and other reflecting surfaces in the scene.

(Fig. 1.1) : Light viewed from an opaque nonluminous surface is in a combination of reflected light from a light source and reflections of light reflections from other surfaces.



LIGHT EMITTING AND LIGHT REFLECTING SOURCES

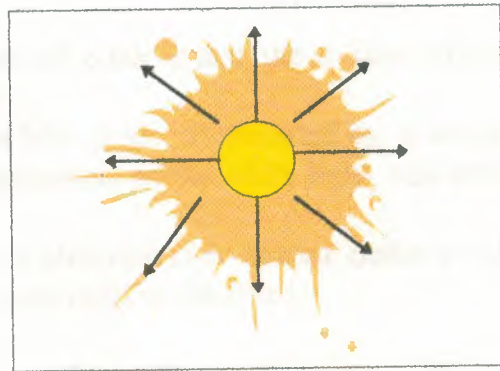
If a surface that is not directly exposed to a light source may still be visible if near by objects are illuminated. Sometimes, light sources are referred to as *light-emitting sources*; and light reflecting surfaces, such as the walls of a room, are termed *light reflecting sources*.

I will use the term *light source* to mean an object that is emitting radiant energy, such as a light bulb or the sun.

A luminous object, in general, can be both a light source and a light reflector. For example, a plastic globe with a light bulb inside both emits and reflects light from the surface of the globe. Emitted light from the globe may then illuminate other objects in the vicinity.

The simplest model for a light emitter is a *point source*. Rays from the source then follow radially diverging paths from the source position, as shown in :

Fig. (1.2). Diverging ray paths from a light source



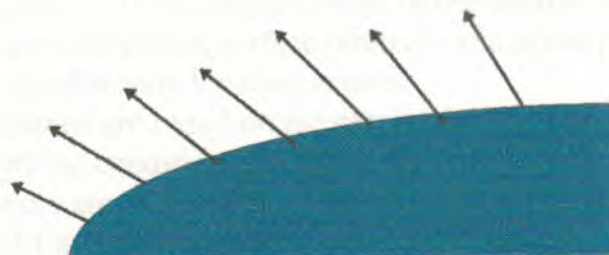
This light-source model is a reasonable approximation for sources whose dimensions are small compared to the size of objects in the scene. Sources, such as the sun, that are sufficiently far from the scene can be accurately modeled as point sources. A nearby source, such as the long fluorescent light in is more accurately modeled as a distributed light source. In this case, the illumination effects can not be approximated realistically with a point source, because the area of the source is not small compared to the surfaces in the scene. An accurate model for the distributed source is one that considers the accumulated illumination effects of the points over the surface of the source.

When light is incident on an opaque surface, part of it is reflected and parts absorbed. The amount of incident light reflected by a surface depends on the type of material. Shiny materials reflect more of the incident light, and dull surfaces absorb more of the incident light. Similarly, for an illuminated transparent surface, some of the light will be reflected and some will be transmitted through the material.

What's a diffuse reflection?

The surfaces that are grainy, rough, tend to scatter the reflected light is called diffuse reflection. A very rough matte surface produces primarily diffuse reflections, so that the surface appears equally bright from all of its surfaces.

Fig 1.4 diffuse reflections from a surface



What we call color is just the diffuse reflection of the incident light from that object :

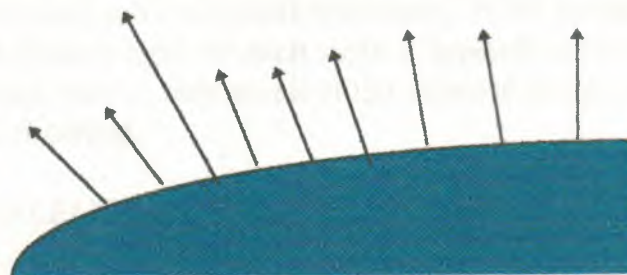
EX: If a blue object illuminated by a white light source, it reflects the blue component of the white light and totally absorbs all the other components.

EX: If the blue object is viewed under a red light, it appears black since all the incident light is absorbed.

What's a specular reflection ?

In addition to diffuse reflection, light sources create highlights, or bright spots, called specular reflection. This highlighting effect is more pronounced on shiny surfaces than on dull surfaces.

Fig(1.3) An illustration of specular reflection is shown :



BASIC ILLUMINATION MODELS

The things that will be discussed is simplified methods for calculating light intensities. The empirical models described in this section provide simple and fast methods for calculating surface intensity at a given point, and they produce reasonably good results for most scenes.

Lighting calculations are based on the optical properties of surfaces, the background lighting conditions, and the light source specifications

Optical parameters are used to the surface properties, such as glossy, matte, opaque, and transparent. This controls the amount of reflection and absorption of incident light. All light sources are considered to the point sources, specified with a coordinate position and an intensity value (color)

Ambient Light

A surface that is not exposed directly to a light source still will be visible if nearby objects are illuminated. In our basic illumination model, we can set a general level of brightness for a scene. This is a simple way to model the combination of light reflections from various surfaces to produce a uniform illumination called the *ambient light*, or *background light*.

Ambient light has no spatial or directional characteristics. The amount of ambient light incident on each object is a constant for all surfaces and over all directions.

We can set the level for the ambient light in a scene with parameter I_a , and each surface is then illuminated with this constant value. The resulting reflected light constant for all surfaces and over all directions independent of the viewing direction and the spatial orientation of the surface. But the intensity of the reflected light for each surface depends on the optical properties of the surface; that is, how much of the incident energy is to be reflected and how much absorbed.

DIFFUSE REFLECTION

Ambient-light reflection is approximation of global diffuse lighting effects. Diffuse reflections are constant over each surface in a scene, independent of the viewing direction. The fractional amount of the incident light that is diffusely reflected can be set for each surface with parameter k_d the *diffused reflection coefficient* or *diffuse reflectivity*. Parameter k is assigned a constant value in the interval 0 to 1, according to the reflecting properties we want the surface to have. If a highly reflective, we set the value of k_d near 1. This produces a bright surface with the intensity of the reflected light near that of the incident light.

To simulate a surface that absorbs most of the incident light, we set the reflectivity to a value near 0. Actually, parameter k_d is a function of surface color, but for the being we will assume k_d is a constant.

If a surface is exposed only to ambient light, we can express the intensity of the diffuse reflection at any point on the surface as

$$I_{\text{ambdiff}} = K_d I_d$$

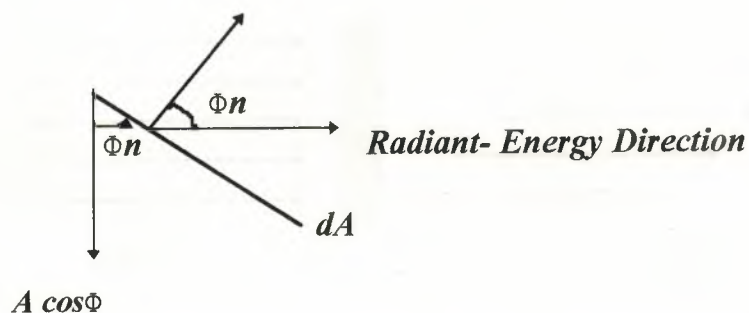
Since ambient light produces a flat-uninteresting shading for each surface scenes are rarely rendered with ambient light alone. At least one light source is included in a scene, often as a point source at the viewing position.

We can model the diffuse reflections of illumination from a point source in a similar way. That is, we assume that the diffuse reflections from the surface are scattered with equal intensity in all directions, independent of the viewing direction. Such surfaces are sometimes referred to as ideal diffuse reflectors.

They are also called Lambertian reflectors since radiated light energy from any point on the surface is governed by Lambert cosine law. This law states that the radiant energy from any small surface area dA in any direction Φ_n relative to the surface normal is proportional to $\cos \Phi_n$.

The light intensity, though, depends on the radiant energy per projected area perpendicular to direction which is $dA \cos \Phi_n$

Result: Lambertian reflection is same over all viewing directions.



fig(1.5) Radiant energy from a surface area dA in direction Φ_n relative to the surface normal direction.

I_i is the intensity of the point light source

$$I_{i,\text{diff}} = k_d I_i \cos \Phi$$

A surface is illuminated by a point source only if the angle of incidence is in the range 0-90 degrees. When $\cos \Phi$ is negative the light source is behind the surface.

If N is the unit normal vector and L is the unit direction vector to the point light source then

$$\cos \phi = N.L$$

And diffuse reflection for single point-source illumination is

$$I_{L,DIFF} = k_d I_i (NL)$$

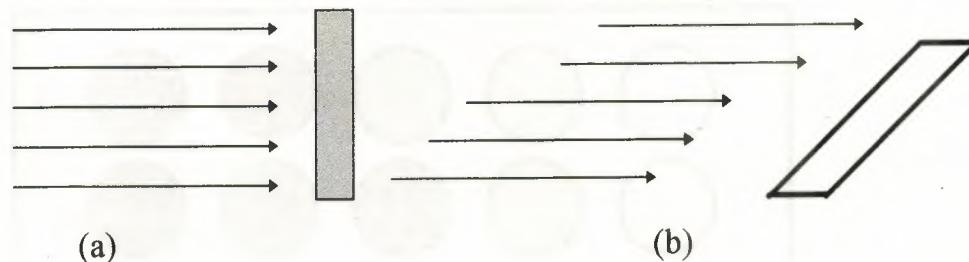
Reflections for point source illumination are calculated in world coordinates or viewing coordinates before shearing and perspective transformations are applied.

Photometry concepts and terms, such as radiant energy will be examined in later sections .

Even though there is equal light scattering in all directions from a perfect diffuse reflector, the brightness of the surface does depend on the orientation of the surface relative to the light source.

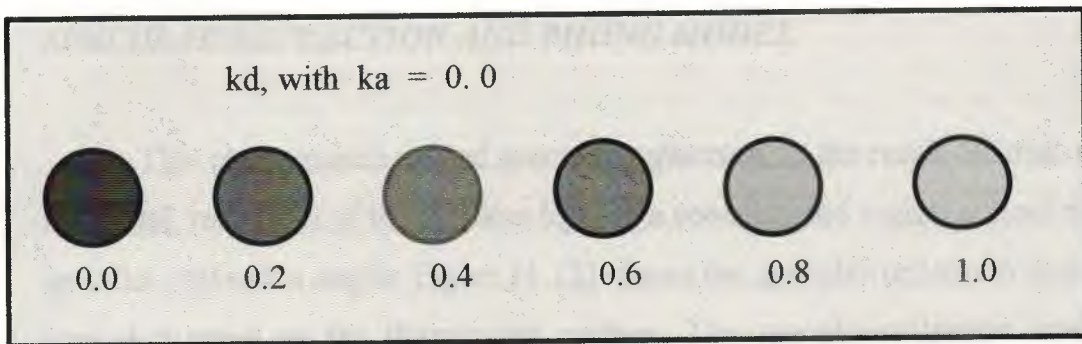
A surface that is oriented perpendicular to the direction of the incident light appears brighter than if the surface were tilted an oblique angle to the direction of the incoming light. This is easily seen by holding a white sheet of paper or smooth cardboard parallel to a nearby window and slowly rotating the sheet away from the window direction.

As the triangle between the surface normal and incoming light direction increases, less of the incident light falls on the surface, as shown in fig. 14-7:



Surfaces perpendicular to the incident light (a) is more illuminated than an equal -sized surface at an oblique angle (b) to the incoming light direction.

This figure shows a beam of light rays incident on two equal-area plane surface .



Diffuse reflections from a spherical surface illuminated by a point light source for values of the diffuse reflectivity coefficient in the interval

$$0 \leq k_d \leq 1$$

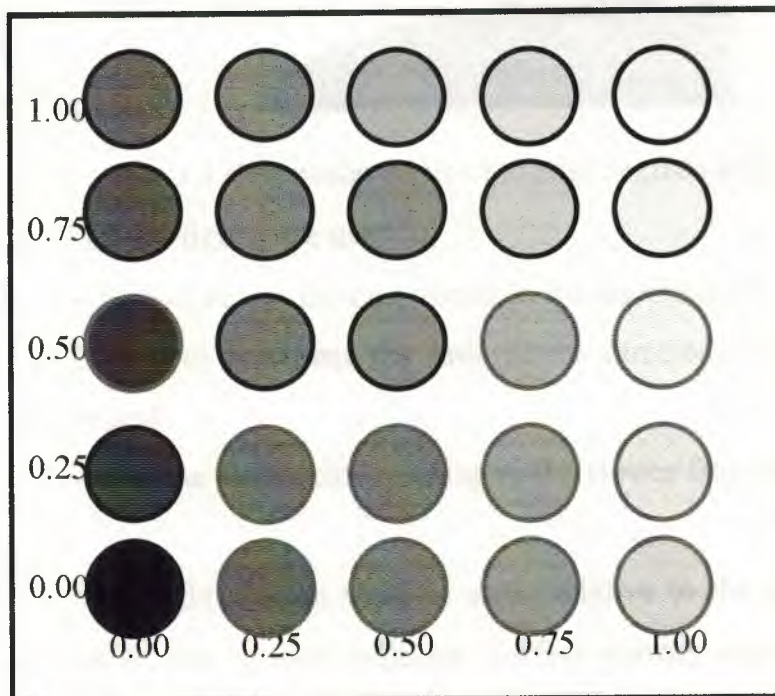
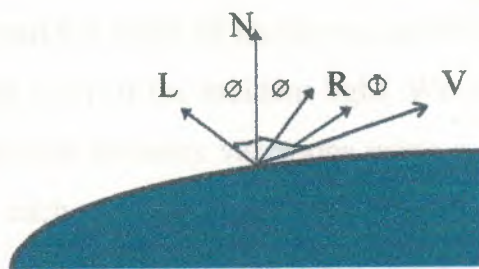


fig (1.6) Diffuse reflections from a spherical surface illuminated with ambient light and a single point source for values of k_a and k_d in the interval $(0,1)$.

SPECULAR REFLECTION AND PHONG MODEL

This phenomenon, called *specular reflection*, is the result of total, or near total, reflection of the incident light in a concentrated region around the specular *reflection angle*. Figure (1.12) shows the specular reflection direction at a point on the illuminated surface. The specular-reflection angle equals the angle of the incident light, with the two angles measured on opposite sides of the unit normal surface vector N .



fig(1.12) Specular reflection angle equals angle of incident θ .

In this figure, we use:

- R to represent the unit vector in the direction of all specular reflection;
- L to represent the unit vector directed toward the point light source; and
- V as the unit vector pointing to the viewer from the surface position.

Angle Φ is the viewing angle relative to the specular-reflection direction R . For an ideal reflector (perfect mirror), incident light is reflected only in the specular-reflection direction. In this case, we would only see reflected light when vectors V and R coincide ($\Phi = 0$).

Objects other than ideal reflectors exhibit specular reflections over a finite range of viewing positions around vector R . Shiny surfaces have a narrow specular-reflection range, and dull surfaces have a wider reflection range.

An empirical model for calculating the specular-reflection range, developed by Phong Bui Tuong and called the *Phong specular-reflection model*, or simply the Phong model, sets the intensity of specular reflection proportional to $\cos^{ns} \Phi$.

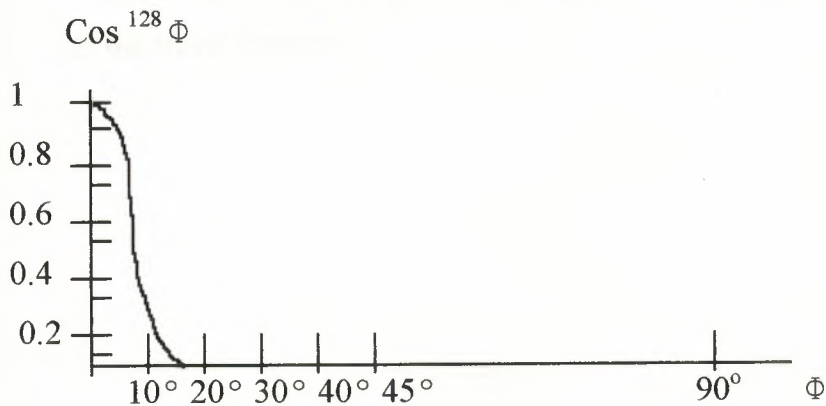
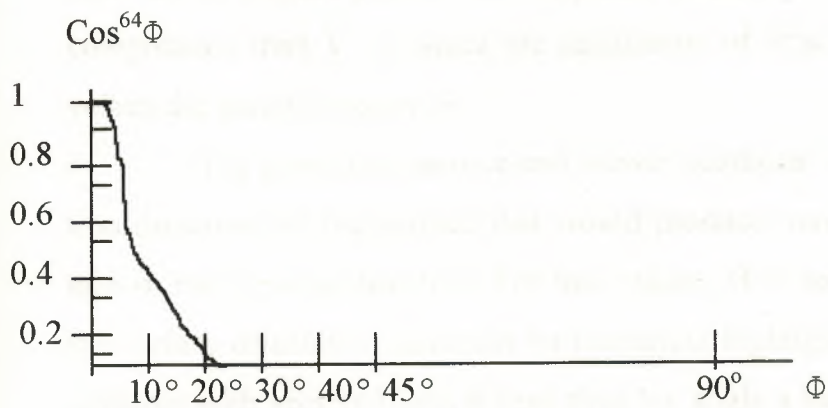
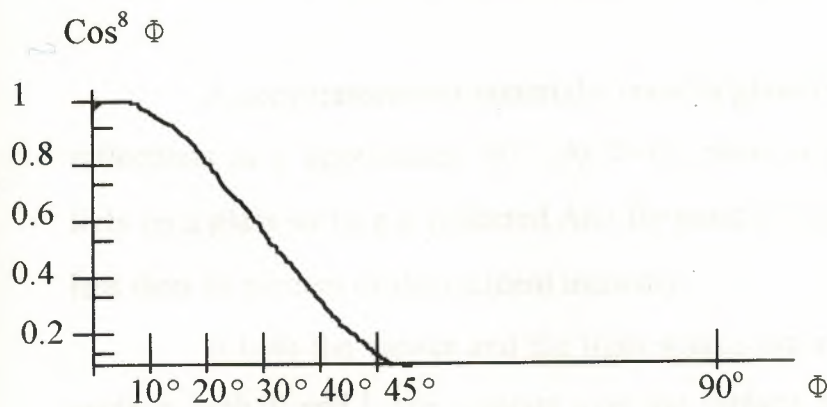
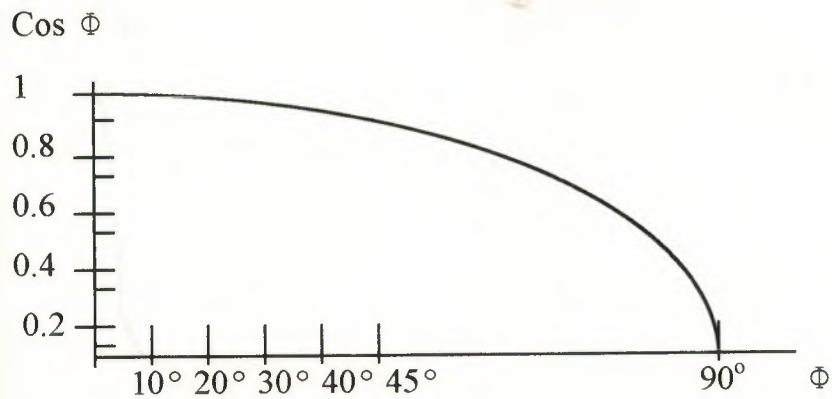
Angle Φ can be assigned values in the range 0° to 90° , so that $\cos \Phi$ varies from 0 to 1. The value assigned to specular-reflection parameter n_s , is determined by the type of surface that we want to display. A very shiny surface is modeled with a large value for (100 or more), and smaller values (down to 1) are used for duller surfaces. For a perfect reflector, n_s is infinite. For a rough surface, such as chalk or cinderblock, n_s would be assigned a value near 1.

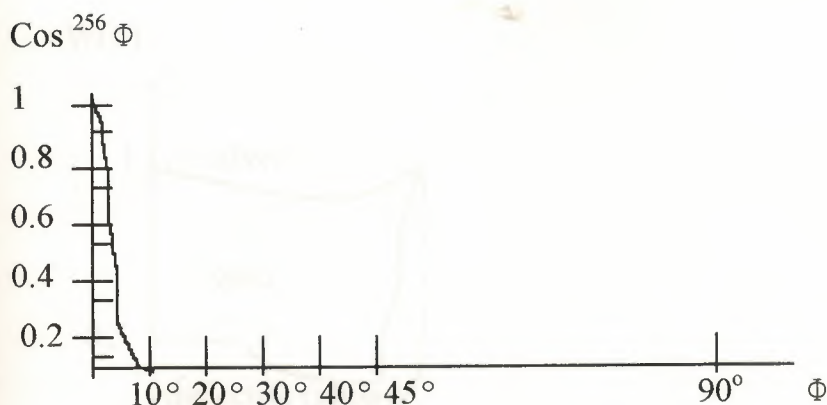
The intensity of specular reflection depends on the material properties of the surface and the angle of incidence, as well as other factors such as the polarization and color of the incident light. We can approximately model monochromatic specular intensity variations using a specular-reflection coefficient, $W(\Phi)$, for each surface. Figure 1.5 shows the general variation of $W(\Phi)$ over the range $\Phi = 0^\circ$ to $\Phi = 90^\circ$ for a few materials. In general, $W(\Phi)$ tends to increase as the angle of incidence increases.

$\Phi = 90^\circ$, $W(\Phi) = 1$ and all of the incident light is reflected. The variation of specular intensity with angle of incidence is described by Fresnel's Laws of Reflection. Using the spectral-reflection function $W(\Phi)$, we can write the Phong specular-reflection model as

$$I_{\text{spec}} = W(\Phi) I, \cos^{ns} \Phi$$

where I , is the intensity of the light source, and Φ is the viewing angle relative to the specular-reflection direction R .





fig(1,22) plots of $\cos^{ns} \Phi$ for several values of specular parameter n_s .

A seen transparent materials such as glass only exhibit appreciable reflections as Φ approaches 90° . At $\Phi=0^\circ$ about 4 percent of the incident light on a glass surface is reflected. And for most of the range of Φ intensity is less than 10 percent of the incident intensity.

If both the viewer and the light source are sufficiently far from the surface, both V and L are constant over the surface, and thus H is also constant for all surface points. For nonplanar surfaces, $N \cdot H$ then requires less computation than $V \cdot R$ since the calculation of R at each surface point involves the variable vector N .

For given light-source and viewer positions, vector H is the orientation direction for the surface that would produce maximum specular reflection in the viewing direction. For this reason, H is sometimes referred to as the surface orientation direction for maximum highlights. Also, if vector V is coplanar with vectors L and R (and thus N), angle α has the value $\Phi/2$. When V , L , and N are not coplanar, $\alpha > \Phi/2$, depending on the spatial relationship of the three vectors.

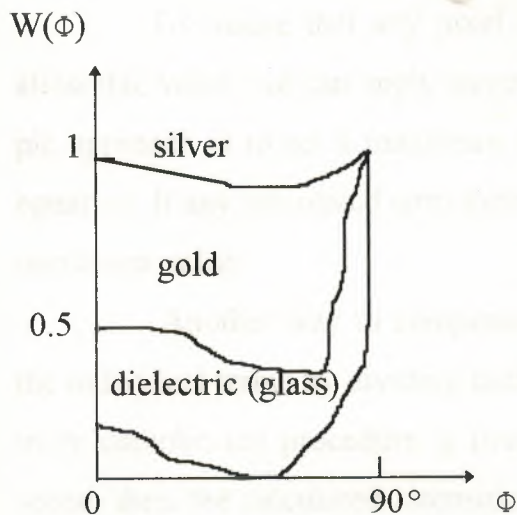


Figure (1.23) Approximate variation of the specular -reflection coefficient as a function of angle of incidence for different materials.

Combined Diffuse and Specular Reflections with Multiple Light Sources

On a single point light source, we can model the combined diffuse and specular reflections from a point on an illuminated surface as

$$\begin{aligned}
 I &= I_{\text{diff}} + I_{\text{spec}} & (1.9) \\
 &= k_a I_a \Phi + k_d I_i (\mathbf{N} \cdot \mathbf{L}) + k_s I_i (\mathbf{N} \cdot \mathbf{H})^{n/s}
 \end{aligned}$$

illustrates surface lighting effects produced by the various terms in Eq.(1.9). If we place more than one point source in a scene, we obtain the light reflection at any surface point by summing the contributions from the individual sources:

To ensure that any pixel intensity does not exceed the maximum allowable value, we can apply some type of normalization procedure. A simple approach is to set a maximum magnitude for each term in the intensity equation. If any calculated term exceeds the maximum, we simply set it to the maximum value.

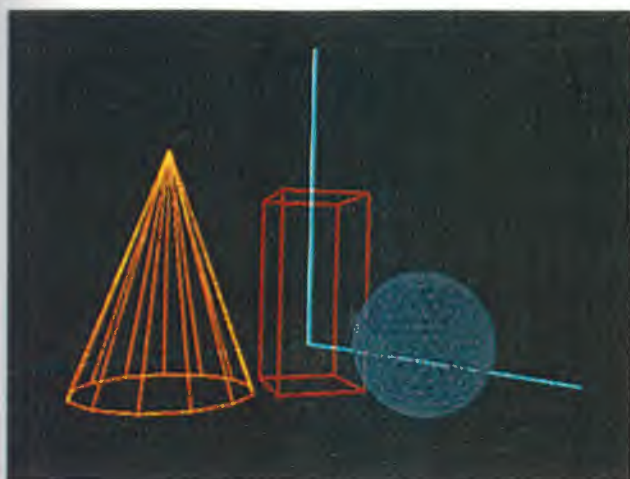
Another way to compensate for intensity overflow is to normalize the individual terms by dividing each by the magnitude of the largest term. A more complicated procedure is first to calculate all pixel intensities for the scene, then the calculated intensities are scaled onto the allowable intensity range.

Warn Model

So far we have considered only point light sources. The Warn model provides a method for simulating studio lighting effects by controlling light intensity in different directions.

Light sources are modeled as points on a reflecting surface, using the Phong model for the surface points. Then the intensity in different directions is controlled by selecting values for the Phong exponent. In addition, light controls, such as "barn doors" and spotlighting, used by studio photographers can be simulated in the Warn model.

This figure shows a wireframe scene (a) is displayed only with ambient lighting in (b) and the surface of each object is assigned a different color. Using ambient light and diffuse reflections due to a single source with $k_s = 0$ for all surfaces, we obtain the light and both diffuse and specular reflections due to a single light source, we obtain the lighting effects shown in (d)



(a)



(b)



(c)



(d)

Flaps are used to control the amount of light emitted by a source in various directions. Two flaps are provided for each of the x, y, and z directions.

Spotlights are used to control the amount of light emitted within a cone with apex at a point-source position. The Warn model is implemented in

Intensity attenuation

The radiant energy from a point light source travels through space its amplitude is assigned by the factor $1/d^2$ where d is the distance that the light has traveled. This factor produces too much intensity variations when d is small and it produces very little variation when d is large.

Graphic packages have compensated these problems by using inverse linear or quadratic functions of d to attenuate intensities. A general inverse quadratic *attenuation function* as:

$$f(d) = 1 / (a_0 + a_1 d + a_2 d^2)$$

A user can construct the coefficients a_0 a_1 a_2 to obtain various lighting effects of a scene. The value for constant term a_0 can be adjusted to prevent $f(d)$ from becoming too large when d is very small.

With a given set of attenuation coefficients, we can limit the magnitude of the attenuation function to 1 with the calculation.

$$f(d) = \min (1, 1 / (a_0 + a_1 d + a_2 d^2))$$

Using this function we can then write our basic illumination model as:

$$I = k_a I_a + \sum f(d_i) I_{ii} [k_d (N.L) + k_s (N.H_i)^{ms}]$$

where d_i is the distance light traveled from light source i .

If the surface is close to the light source (small d) receives a higher incident intensity from the source then a distant surface (large d).

To produce realistic lightning effects our illumination model should take this intensity attenuation into account. If we are illuminating all surfaces with the same intensity, no matter how far they might be from the light source. If two parallel surfaces with the same optical parameters overlap, they would be displayed as one surface.

Color Considerations

Most graphics displays of realistic scenes are in color. But the illumination model we have described so far considers only monochromatic lighting effects. To incorporate color, we need to write the intensity equation as a function of the color properties of the light sources and object surfaces.

For an RGB description, each color in a scene is expressed in terms of red, green, and blue components. We then specify the RGB components of lightsource intensities and surface colors, and the illumination model calculates the RGB components of the reflected light. One way to set surface colors is by specifying the reflectivity coefficients as three-element vectors.

The diffuse reflection coefficient vector, for example, would then have RGB components (k_{dR} , k_{dG} , k_{dB}). If we want an object to have a blue surface, we select a nonzero value in the range from 0 to 1 for the blue reflectivity component, k_{dB} , while the red and green reflectivity components are set to zero ($k_{dR} = k_{dG} = 0$). Any nonzero red or green components in the incident light are absorbed, and only the blue component is reflected. The intensity calculation for this example reduces to the single expression.

$$\text{Eq 1.14 } I_B = k_{aB} I_{aB} + \sum f_i(d) I_{iB} [k_{dB} (N.L) + k_{sB} (N.H_i)^{ns}]$$



Fig (1.23) Light reflections from trombones with reflectance parameters set to simulate shiny brass surfaces

Surfaces typically are illuminated with white light sources, and in general we can set surface color so that the reflected light has nonzero values for all three RGB components. Calculated intensity levels for each color component can be used to adjust the corresponding electron gun in an RGB monitor.

In original specular-reflection model, Phong set parameter k_s to a constant value independent of the surface color. This produces specular reflections that are the same color as the incident light (usually white), which gives the surface a plastic appearance. For a nonplastic material, the color of the specular reflection is a function of the surface properties and may be different from both the color of the incident light and the color of the diffuse reflections.

We can approximate specular effects on such surfaces by making the specular reflection coefficient color dependent. Other color representations besides RGB can be used to describe colors in a scene. And sometimes it is convenient to use a color model with more than three components for a color specification.

Fig (1.24) Light reflection from a teapot with reflectance parameters set to simulate brushed aluminum surfaces and rendered using Monte Carlo ray-tracing methods

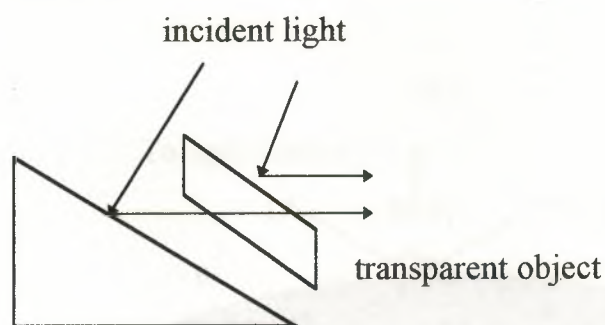


We can approximate specular effects such surfaces by making the specular-reflection coefficient color-dependent, as in Eq. 1.14.

TRANSPARENCY

A transparent surface in general produces both reflected and transmitted light.

Relative contribution of the transmitted light depends on the degree of transparency of the surface and whether any light sources or illuminated surfaces are behind the transparent surface. Figure 1.25 illustrates the intensity contributions to the surface lighting for a transparent object.



fig(1.25) Light emission from transparent surface is in general a combination of reflected and transmitted light.

When a transparent surface is to be modeled, the intensity equations must be modified to include contributions from light passing through the surface. In most cases, the transmitted light is generated from reflecting objects in back of the surface.

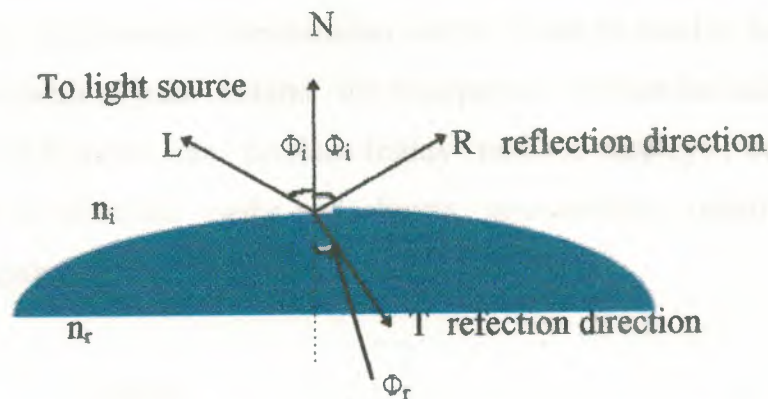
Reflected light from these objects passes through the transparent surface and contributes to the total surface intensity.

Both diffuse and specular transmission can take place at the surfaces of a transparent object. Diffuse effects are important when a partially transparent surface, such as frosted glass, is to be modeled. Light passing

through such materials is scattered so that a blurred image of background objects is obtained.

Diffuse refraction's can be generated by decreasing the intensity of the refracted light and spreading intensity contributions at each point on the refracting surface onto a finite area. These manipulations are time-consuming, and most lighting models employ only specular effects.

Realistic transparency effects are modeled by considering light refraction. When light is incident upon a transparent surface, part of it is reflected and part is *refracted* (Fig.1.7). Because the speed of light is different in different materials, the path of the refracted light is different from that of the incident light. The direction of the refracted light, specified by the angle of refraction, is a function of the index of refraction of each material and the direction of the incident light.



Fig(1.7) Refraction direction R and refraction direction T for a ray of light incident upon a surface with index of refraction n_r .

Index refraction for a material is defined as the ratio of the speed of light in a vacuum to the speed of light in the material. Angle of refraction Φ_r is calculated from the angle of incidence Φ_i , the index of refraction n_i ; of the "incident" material (usually air), and the index of refraction n_r of the refracting material according to Snell's law:

$$\sin \Phi_r = \sin \Phi_i (n_i / n_r)$$

The index of refraction of a material is a function of the wavelength of the incident light, so that the different color components of a light ray will be refracted at different angles.

For most applications we can use an average index of refraction for the different materials they are modeled in a scene. The index of refraction of air is 1 and that of crown glass is about 1.5. Figure (1.28) illustrates the changes in the path direction for a light ray refracted through a glass object.

We can obtain the unit transmission vector T in the refraction direction Φ_r as:

$$T = ((n_i / n_r) (\cos \Phi_i) - \cos \Phi_r) N - (n_i / n_r) L$$

Where N is unit surface normal, and L is the unit vector in the direction of the light source. Transmission vector T can be used to locate intersection path with objects behind the transparent surface. Including refraction effects in a scene can produce highly realistic displays, but the determination of refraction paths and objects intersections requires considerably computation.

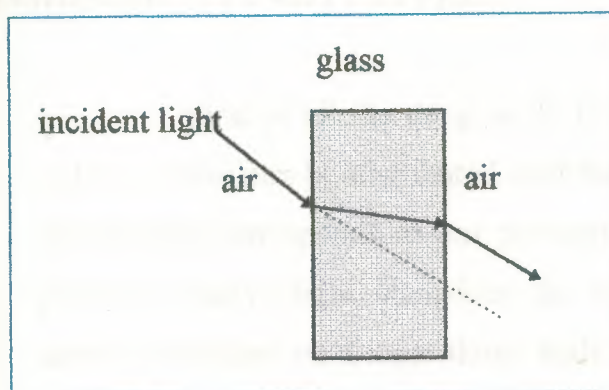


Figure (1.28) refraction of light through a glass object. The emerging refracted ray travels along a path that is parallel to the incident light path (dashed line).

SHADOWS

Hidden-surface methods can be used to locate areas where light sources produce shadows. By applying a hidden-surface method with a light source at the view position we can determine at which surface sections cannot be 'seen' from the light source called *shadows*.

We can display shadow areas with ambient light intensity only, or we can combine the ambient light with specified surface textures.

DISPLAYING LIGHT INTENSITIES

Intensity values calculated by an illumination model must be converted to one of the allowable intensity levels for the particular graphics system in use.

Systems are capable of displaying several intensity levels, while others are capable of only two levels for each pixel (on or off). In the first case, we convert intensities from the lighting model into one of the available levels for storage in the frame buffer. For bilevel systems, we can convert intensities into halftone patterns, as discussed in the next section.

ASSIGNING INTENSITY LEVELS

First of all the thing to do is consider how grayscale values on a video monitor can be distributed over the range between 0 and 1 so that the distribution corresponds to our perception of equal intensity intervals. We perceive relative light intensities the same way that we perceive relative sound intensities: on a logarithmic scale. This means that if the ratio of two intensities is the same as the ratio of two other intensities, we perceive the difference between each pair of intensities to be the same.

As an example, we perceive the difference between intensities 0.20 and 0.22 to be the same as the difference between 0.80 and 0.88. Therefore, to display $n + 1$ successive intensity levels with equal perceived brightness,

the intensity levels on the monitor should be spaced so that the ratio of successive intensities is constant:

$$r = I_n / I_{n-1}$$

Here, we denote the lowest level that can be displayed on the monitor as I_0 and the highest as I_n . Any intermediate intensity can then be expressed in terms of I_0 as

Eq a: $I_k = r^k I_0$

We can calculate the value of r , given the values of I_0 and n for a particular system, by substituting $k = n$ in the preceding expression.

Since $I_n = 1$, we have

Eq b: $r = (1/I_0)^{1/n}$

Thus, the calculation for I_k in Eq. 14-21 can be rewritten as

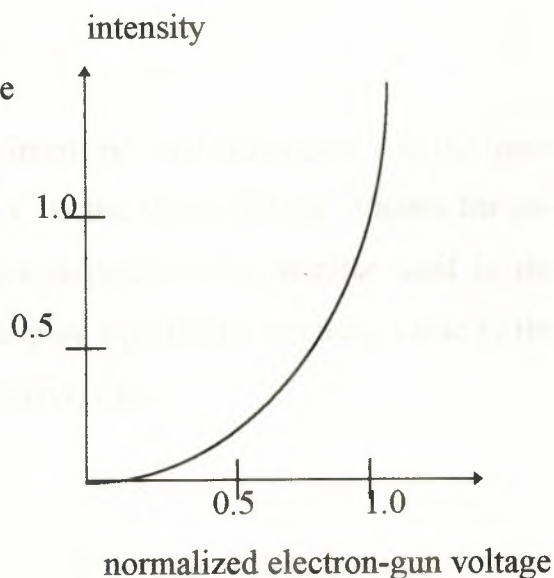
Eq c: $I_k = I_0^{(n-k)/n}$

As an example, if $I_0 = 1/8$ for a system with $n = 3$, we have $r = 2$, and the four intensity values are $1/8$, $1/4$, $1/2$, and 1 .

The lowest intensity value I_0 depends on the characteristics of the monitor and is typically in the range from 0.005 to around 0.025. As in a "black" region displayed on a monitor will always have some intensity value above 0 due to reflected light from the screen phosphors. For a black-and-white monitor with 8 bits per pixel ($n = 255$) and $I_0 = 0.01$, the ratio of successive intensities is approximately $r = 1.0182$. The approximate values for the 256 intensities on this system are 0.0100, .0102, 0.0104, 0.0106, 0.0107, 0.0109, . . . , 0.9821, and 1.0000.

With a color system, we set up intensity levels for each component of the color model. Using the RGB model, for example, we can relate the blue component of intensity at level k to the lowest attainable blue value as in Eq.a.

fig(1.31) A typically monitor response curve, showing the displayed screen intensity as a function of normalized electron-gun voltage.



GAMMA CORRECTION AND VIDEO LOOKUP TABLES

Another problem associated with the display of calculated intensities is the nonlinearity of display devices. Illumination models produce a linear range of intensities. The RGB color (0.25, 0.25, 0.25) obtained from a lighting model represents one half the intensity of the color (0.5, 0.5, 0.5).

Usually, these calculated intensities are then stored in an image file as integer values, with one byte for each of the three RGB components. This intensity file is also linear, so that a pixel with the value (64, 64, 64) has one half the intensity of a pixel with the value (128, 128, 128). A video monitor is a nonlinear device.

If we say the voltages for the electron gun proportional to the linear pixel values, the displayed intensities will be shifted according to the *monitor response curve* as fig(1.31).

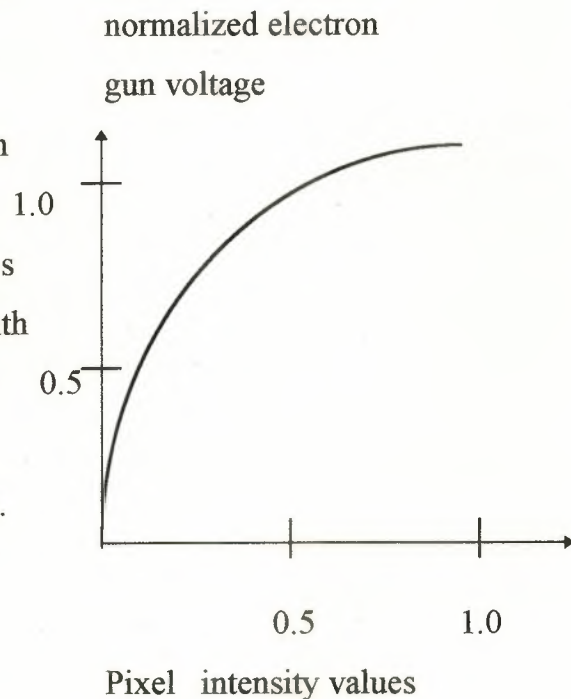
To correct for monitor nonlinearities graphics systems use a *video lookup table* that adjusts the linear pixel values. The monitor response curve is described by the exponential function.

$$\text{Eq 1.a} \quad I = a V^{\gamma}$$

Parameter I is the displayed intensity, and parameter V is the input voltage. Values for parameters a and γ is the input voltage. Values for parameters a and γ depend on the characteristics of the monitor used in the graphics system. Thus, if we want to display a particular intensity value I , the correct voltage value to produce this intensity is :

$$\text{Eq 1.b} \quad V = (I^{1/\gamma})/a$$

Fig 1.3 A video look up correction curve for mapping pixel intensities to electron-gun voltages using gamma correction with $\gamma = 2.22$. Values for both pixel intensity and monitor voltages are normalized on the interval 0-1.



Above calculation is referred to as *gamma correction* of intensity . Monitor gamma values are typically between 2.0 and 3.0 .The national television system committee (NTSC) signal standard is $\gamma = 2.2$ Fig 1.3 shows a gamma correction curve using the NTSC gamma value.Eq 1.b is used to set up the video look up table that converts integer pixel values in the image files to values that control the electron gun voltages.

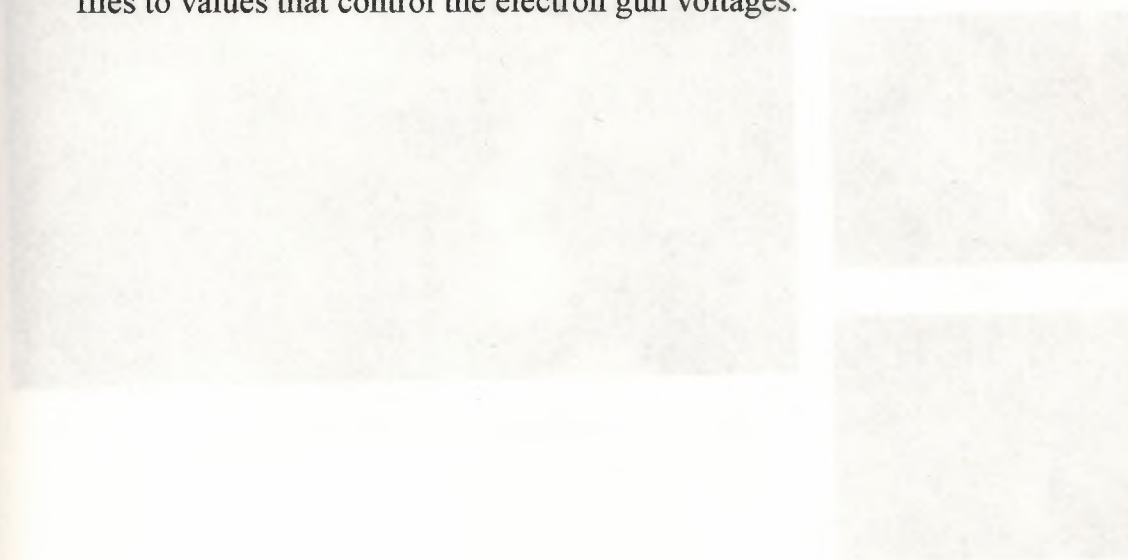
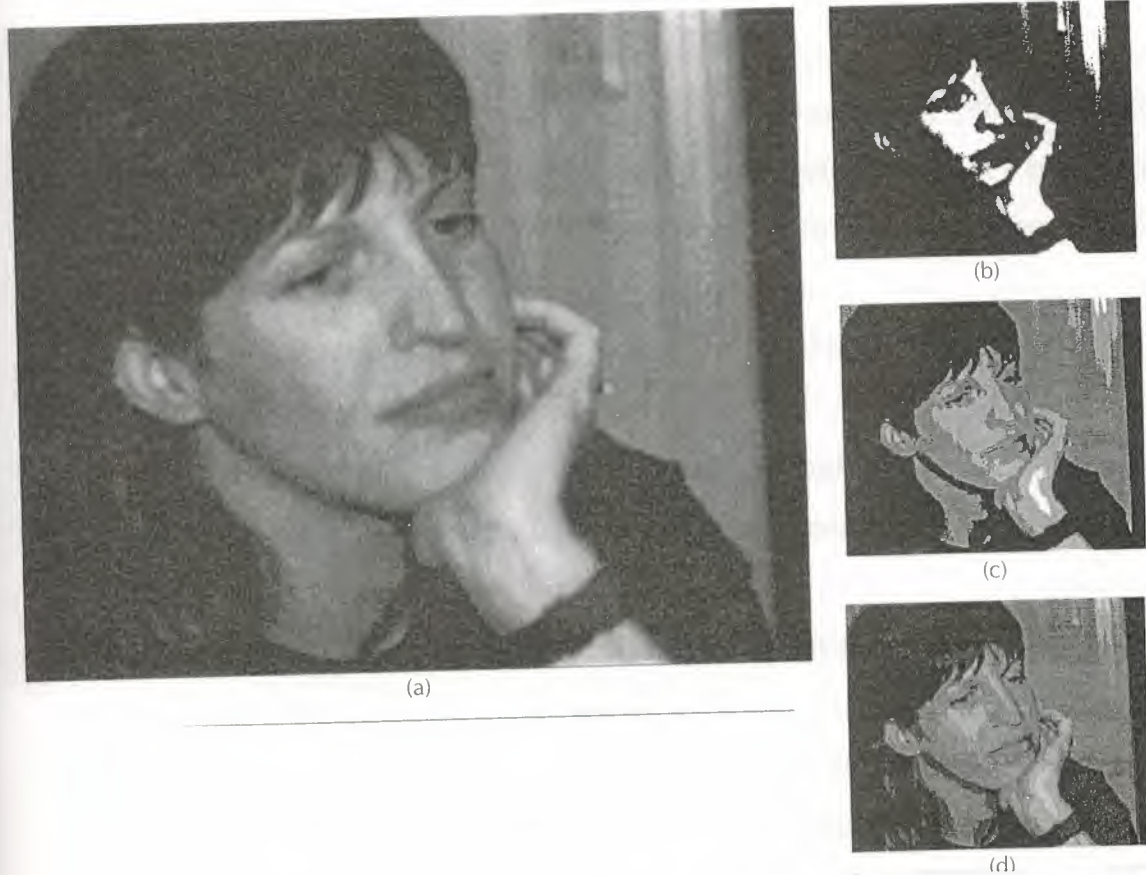


Fig 1.3 | A gamma correction curve (a) gamma = 1.0 (b) gamma = 2.2

gamma correction can be combined with a gamma correction map to produce a look up table that contains both corrections. If I is an input intensity value from an image, then the gamma correction is increased by γ and a gamma value is used with equation 1.c.

If the video resolution of a monitor is 640x480, then the frame buffer must be 640x480x3. The gamma correction map is a 640x480x3 array of values. The gamma correction map is used to correct the gamma of the image. The gamma correction map is used to correct the gamma of the image. The gamma correction map is used to correct the gamma of the image.



Fig(1.33) A continuous tone photograph (a) printed with (b) two intensity levels , (c) four intensity levels , and (d) eight intensity levels.

Gamma correction can be combined with logarithmic intensity mapping to produce a look up table that contains both conversions. If I is an input intensity value from an illumination model , first nearest intensity is located I_k from a table of values created with equations Eq c.

If the video amplifiers of a monitor are designed to convert the linear input pixel values to electron gun voltages we can not combine the two intensity - conversion processes. In this case , gamma correction is build into the hardware and the logarithmic values I_k must be precomputed and stored in the frame buffer or the color table.

DISPLAYING CONTINUOUS TONE IMAGES

High quality computer graphics systems generally provide 256 intensity levels for each color component, but acceptable displays can be obtained for many applications with fewer levels. A four level system provides minimum shading capability for continuous tone images, while photorealistic images can be generated on systems that are capable of from 32 to 256 intensity levels per pixel.

Fig 1.33 shows a continuous tone photograph displayed with various intensity levels. When a small number of intensity levels are used to reproduce a continuous tone image, the borders between the different intensity regions (called *contours*) are clearly visible. In the two level reproduction, the features of the photograph are just barely identifiable. Using four intensity levels, we begin to identify the original shading patterns, but the contouring effects are glaring. With eight intensity levels, contouring effects are still obvious, but we begin to have a better indication of the original shading. At 16 or more intensity levels, contouring effects diminish and the reproductions are very close to the original. Reproductions of continuous tone images using more than 32 intensity levels show only very subtle differences from the original.

HALFTONE PATTERNS AND DITHERING TECHNIQUES

When an output device has a limited intensity range, we can create an apparent increase in the number of available intensities by incorporating multiple pixel positions into the display of each intensity value. When we view a small region consisting of several pixel positions, our eyes tend to integrate or average the fine detail into an overall intensity. Bilevel monitors and printers, in particular, can take advantage of this visual effect to produce pictures that appear to be displayed with multiple intensity values.

Continuous-tone photographs are reproduced for publication in newspapers, magazines, and books with a printing process called halftoning, and the reproduced pictures are called halftones. For a black-and-white photograph, each intensity area is reproduced as a series of black circles on a white background. The diameter of each circle is proportional to the darkness required for that intensity region. Darker regions are printed with larger circles, and lighter regions are printed with smaller circles (more white area). An enlarged section of a photograph reproduced with a halftoning method, showing how tones are represented with varying size dots as gray-scale halftone reproduction.

Color halftones are printed using dots of various sizes and colors. Book and magazine halftones are printed on high-quality paper using approximately 60 to 80 circles of varying diameter per centimeter. Newspapers use lower-quality paper and lower resolution (about 25 to 30 dots per centimeter).

Halftone Approximations

In computer graphics, halftone reproductions are approximated using rectangular pixel regions, called halftone patterns or pixel patterns.

The number of intensity levels that we can display with this method depends on how many pixels we include in the rectangular grids and how many levels a system can display.

With n by n pixel for each grid on a bilevel system can display $n^2 + 1$ intensity levels.

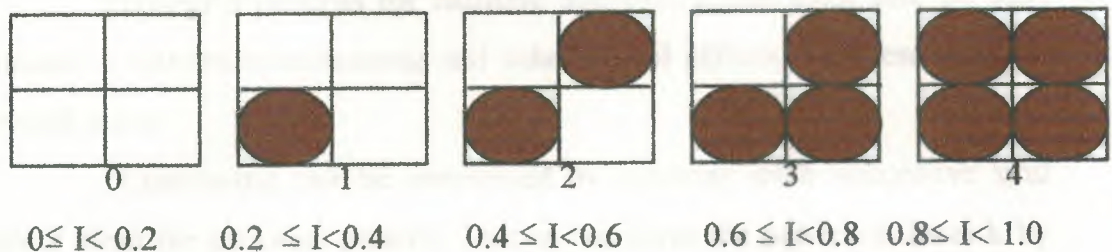


Figure (1.36) A 2 by 2 pixel grid used to display five intensity levels on a bilevel system. The intensity values that could be mapped to each grid are listed below each pixel pattern.

This figure shows one way to set up pixel patterns to represent five intensity levels that could be used with a bilevel system.

To display a particular intensity with level number k , we turn on each pixel whose position number is less than or equal to k .

Although the use of n by n pixel patterns increases the number of intensities that can be displayed, they reduce the resolution of the displayed picture by a factor of $1/n$ along each of the x and y axes. A 512 by 512 screen area, for instance, is reduced to an area containing 256 by 256 intensity points with 2 by 2 grid patterns. And with 3 by 3 patterns, we would reduce the 512 by 512 area to 128 intensity positions along each side.

Another problem with pixel grids is that subgrid patterns become apparent as the grid size increases. The grid size that can be used without distorting the intensity variations depends on the size of a displayed pixel. Therefore, for systems with lower resolution (fewer pixels per centimeter), we must be satisfied with fewer intensity levels. On the other hand, high-quality displays require at least 64 intensity levels. This means that we need 8 by 8 pixel grids. And to achieve a resolution equivalent to that of halftones in books and magazines, we must display 60 dots per centimeter. Thus, we need to be able to display $60 \times 8 = 480$ dots per centimeter. Some devices, for example high-quality film recorders, are able to display this resolution.

Pixel-grid patterns for halftone approximations must also be constructed to minimize contouring and other visual effects not present in the original scene.

Contouring can be minimized by evolving each successive grid pattern from the previous pattern. That is, we form the pattern at level k by adding an "on" position to the grid pattern at level $k-1$. Thus, if a pixel position is on for one grid level, it is on for all higher levels. We can minimize the introduction of other visual effects by avoiding symmetrical patterns.

With a 3 by 3 pixel grid, for instance, the third intensity level above zero would be better represented by the pattern in Fig. 1-38(a) than by any of the symmetrical arrangements in Fig. 1-38(b).

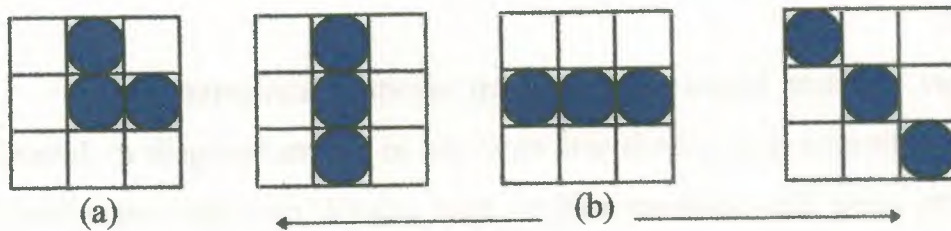
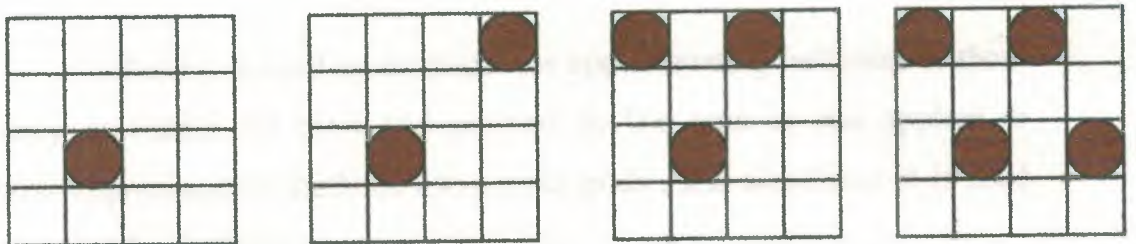
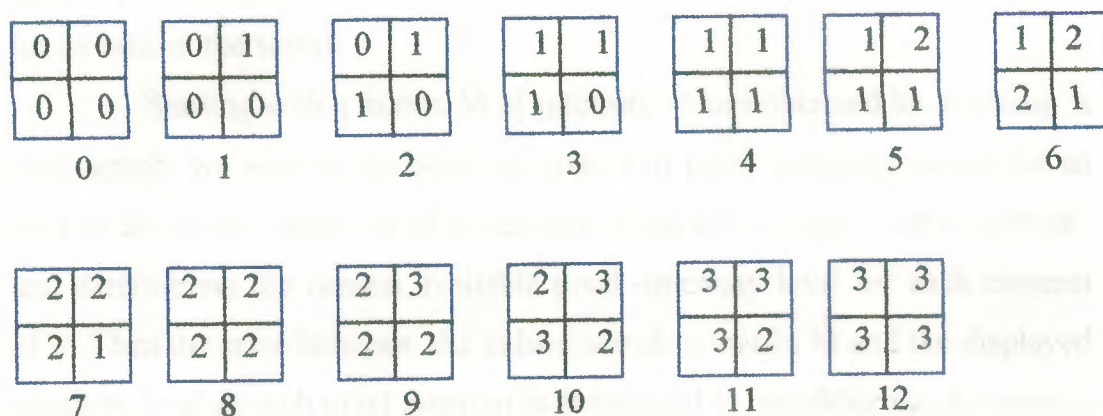


Fig (1-38) For a 3 by 3 pixel grid, pattern (a) is to be preferred to the patterns in (b) for representing the third intensity level above 0.



Fig(1.39) Halftone grid patterns with isolated pixels that can not be effectively reproduced on some hard-copy devices.



Fig(1.40) Intensity levels 0 through 12 obtained with halftone approximations using 2 by 2 pixel grids on four-level system.

The symmetrical patterns in this figure would produce vertical, horizontal, or diagonal streaks in any large area shaded with intensity level 3. For hard-copy output on devices such as film recorders and some printers, isolated pixels are not effectively reproduced. There are, a grid pattern with a single "on" pixel or one with isolated "on" pixels, as in Fig. (1-39) should be avoided.

DITHERING TECHNIQUES

Dithering is used in technical for approximating halftones without reducing resolution, as pixel-grid patterns do. The term is also applied to halftone-approximation methods using pixel grids, and sometimes it is used to refer to color halftone approximations only.

Random values applied to pixel intensities to break up contours are often referred to as *dither noise*. The effect is to add noise over an entire picture, which tends to soften intensity boundaries.

Another method for mapping a picture with m and n points to a display area m by n pixel is *error diffusion*. Here, the error between an input intensity value and the displayed pixel intensity level at a given position is

dispersed, or diffused, to pixel positions to the right and below the current pixel position. Starting with a matrix M of intensity values obtained by scanning a photograph, we want to construct an array I of pixel intensity values for an area of the screen.

Starting with a matrix M of intensity values obtained by scanning a photograph, we want to construct an array I of pixel intensity values for an area of the screen. Rows of M is scanned from left to right, top to bottom, and determining the nearest available pixel-intensity level for each element of M . Then the error between the values stored in matrix M and the displayed intensity level at each pixel position is distributed to neighboring elements in M by an algorithm:

```

for i := 1 to m do
  for i := 1 to n do
    begin
      { determine the available intensity level  $I_k$  }
      { that is closest to the value  $M_{i,j}$ . }
       $I_{i,j} := I_k$ ;
       $err := M_{i,j} - I_{i,j}$ ;
       $err := M_{i,j} + \alpha \cdot err$ ;
       $M_{i,j+1} := M_{i,j+1} + \beta \cdot err$ ;
       $M_{i+1,j} := M_{i+1,j} + \gamma \cdot err$ ;
       $M_{i+1,j+1} := M_{i+1,j+1} + \delta \cdot err$ ;
    end
  
```

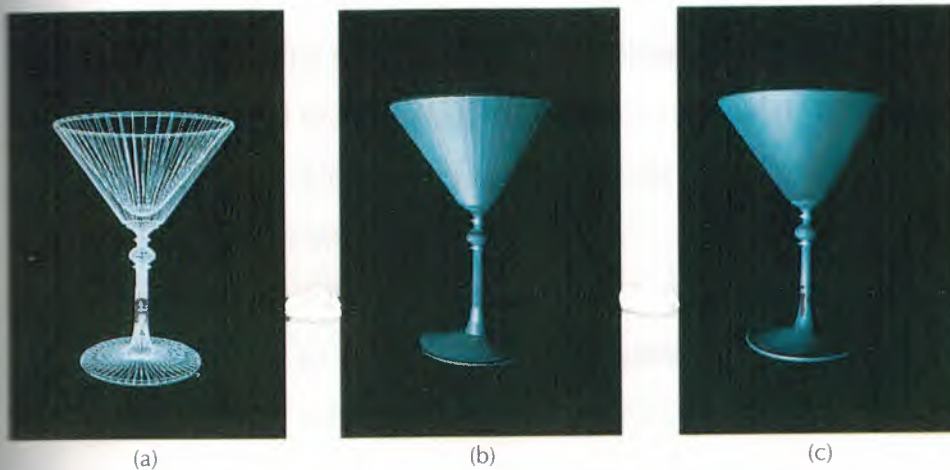


Fig (1.47) A polygon mesh approximation of an object (a) is rendered with flat shading (b) and with Gouraud shading (c)

POLYGON RENDERING METHODS

In this method the application of an illumination model to the rendering of standard graphics objects formed with polygon surfaces. The objects are usually polygon surfaces the objects are usually polygon-mesh approximations of curved-surface objects, but they may also be polyhedral that are not curved-surface approximations. scan line algorithms typically apply a lightning model to obtain polygon surface rendering in one of two ways.

Each polygon can be rendered with an single intensity, or the intensity can be obtained at each point of the surface using an interpolation scheme.

CONSTANT-INTENSITY SHADING:

A simple method for rendering an object with surfaces is *CONSTANT-INTENSITY SHADING*, also called flat shading. In this method, a single intensity is calculated for each polygon. All points over the surface of the polygon are

then displayed with the same intensity value. Constant shading can be useful for displaying the general appearance of a curved surface.

Flat shading of polygon facets provides an accurate rendering for an object if all the following assumptions are valid:

- * The object is a polyhedron and is not an approximation of an object with a curved surface.

- * All light sources illuminating the object are sufficiently far from the surface so that N.L and the attenuation function are constant over the surface.

- * The viewing position is sufficiently far from the surface so that V.R is constant over the surface.

Even if all of these conditions are not true, we can still reasonably approximate surface -lighting effects using small polygon facets with flat shading and calculate the intensity for each facet at the center of the polygon.

GOURAUD SHADING

The intensity-interpolation scheme developed by *Gouraud* and referred to as Gouraud shading, renders a polygon surface by linearly interpolating intensity values across the surface. Intensity values for each polygon are matched with the values of adjacent polygons along the common edges, thus eliminating the intensity discontinuities that can occur in flat shading.

Each polygon surface is rendered with this method by performing the following calculations:

- * Determine the average unit normal vector at each polygon vertex.
- * Apply an illumination model to each vertex to calculate the vertex intensity.
- * Linearly interpolate the vertex intensities over the surface of the polygon.

At each polygon vertex, we obtain a normal vector by averaging the surface normals of all polygons sharing that vertex.

Fig (1.35) A polygon mesh approximation of an object (a) is rendered with fat shading (b) and with Gouraud shading (c).

Calculations lie above section are used to obtain intensities at successive horizontal pixel positions along each scan line.

When surfaces are to be rendered in color., the intensity of each color component is calculated at the vertices. Gouraud shading can be combined with a hidden-surface algorithm to fill in the visible polygons along each scan line. An example of an object shaded with this method appears in fig (1.35).

This method removes the intensity discontinuities associated with the constant-shading model, but it has some other deficiencies. Highlights on the surface are sometimes displayed with anomalous shapes, and the linear intensity interpolation can cause bright or dark intensity streaks, called MACH bands, to appear on the surface. These effects can be reduced by dividing the surface into a greater number of polygon faces or by using other methods such as Phong shading that require more calculations.

PHONG SHADING:

A more accurate method for rendering a polygon surface is to interpolate the normal vectors and then apply the illumination model to each surface point. This method developed by Phong Bui Tuong is called Phong shading, or normal vector interpolation shading. It displays more realistic highlights on a surface and reduces the Mach-band effect. A polygon surface is rendered by the following steps:

- * Determine the average unit normal vector at each polygon vertex.
- * Linearly interpolate the vertex normals over the surface of the polygon.
- * Apply an illumination model along each scan line to calculate projected pixel intensities for the surface points.

Finally, we can express the denominator in Eq.14-46 as a Taylor-series expansion and retain terms up to second degree in x and y . This yields

$$I_{\text{DIFF}(x,y)} = T_5 x^2 + T_4 xy + T_3 y^2 + T_2 x + T_1 y + T_0$$

where each T_k is a function of parameters a , b , c , and so forth.

Using forward differences, we can evaluate Eq.14-48 with only two additions for each pixel position (x, y) once the initial forward-difference parameters have been evaluated. Although fast Phong shading reduces the Phong-shading calculations, it still takes approximately twice as long to render a surface with fast Phong shading as it does with Gouraud shading. Normal Phong shading using forward differences takes about six to seven times longer than Gouraud shading.

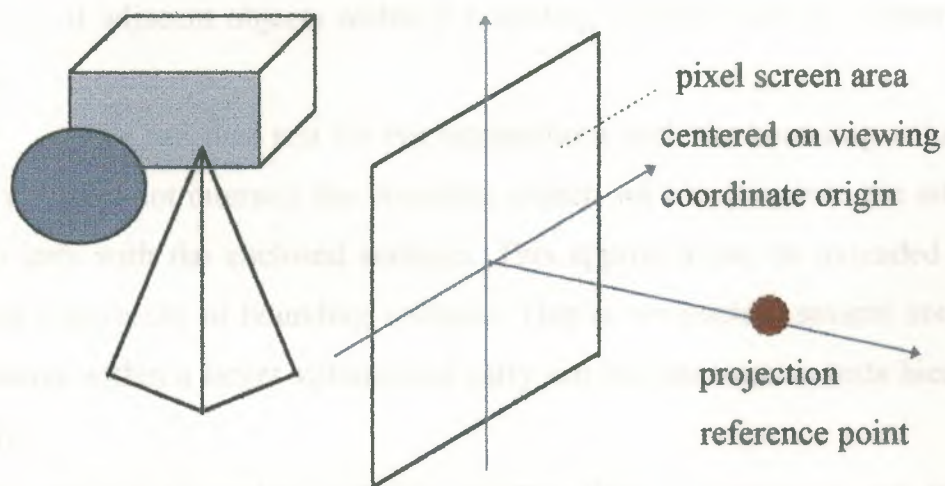
Fast Phong shading for diffuse reflection can be extended to include specular reflections. Calculations similar to those for diffuse reflections are used to evaluate specular terms such as $(N \cdot H)^n$'s in the basic illumination model. In addition, we can generalize the algorithm to include polygons other than triangles and finite viewing positions.

RAY-TRACING METHODS

Up to here we introduced the notion of ray casting, where a ray is sent out from each pixel position to locate surface intersections for object modeling using constructive solid geometry methods. We also discussed the use of ray casting as a method for determining visible surfaces in a scene .

Ray tracing is an extension of this basic idea. Instead of merely looking for the visible surface for each pixel, we continue to bounce the ray around the scene .

As illustrated in Fig. 14-49, collecting intensity contributions.



This provides a simple and powerful rendering technique for obtaining global reflection and transmission effects. The basic ray-tracing algorithm also provides for visible-surface detection, shadow effects, transparency, and multiple light-source illumination. Many extensions to the basic algorithm have been developed to produce photorealistic displays. Ray-traced displays can be highly realistic, particularly for shiny objects, but they require considerable computation time to generate.

Reducing Object-Intersection Calculations

Ray-surface intersection calculations can account for as much as 95 percent of the processing time in a ray tracer. For a scene with many objects, most of the processing time for each ray is spent checking objects that are not visible along the ray path.

Therefore, several methods have been developed for reducing the processing time spent on these intersection calculations.

One method for reducing the intersection calculations is to enclose groups of adjacent objects within a bounding volume, such as a sphere or a box

We can then test for ray intersections with the bounding volume. If the ray does not intersect the bounding object, we can eliminate the intersection tests with the enclosed surfaces. This approach can be extended to include a hierarchy of bounding volumes. That is, we enclose several bounding volumes within a larger volume and carry out the intersection tests hierarchically.

First, we test the outer bounding volume; then, if necessary, we test the smaller inner bounding volumes; and so on.

Space-Subdivision Methods

Another way to reduce intersection calculations, is to use space-subdivision methods. We can enclose a scene within a cube, then we successively subdivide the cube until each subregion (cell) contains no more than a preset maximum number of surfaces. For example, we could require that each cell contain no more than one surface. If parallel - and vector-processing capabilities are available, the maximum number of surfaces per cell can be determined by the size of the vector.

ANTI-ALIASED RAY TRACING

The two basic technical for antialiasing in ray tracing algorithms are *supersampling* and *adaptive sampling*. Sampling in ray tracing is an extension of the sampling methods. In supersampling and adaptive sampling the pixel is treated as a finite square area instead of a single point. Supersampling uses multiple, evenly spaced rays (samples) over each pixel area. Adaptive sampling uses unevenly spaced rays in some regions of the pixel area. For example, more rays can be used near object edges to obtain a better estimate of the pixel intensities. Another method for sampling is to randomly distribute the rays over the pixel area.



Fig.(1.66) This ray traced scene took 24 seconds to render on a Kendall Square research KSR1 Parallel computer with 32 processors. Rodin Thinker was modeled with 3036 primitives. Two light sources and one primary ray per pixel were used to obtain the global illumination effects from the 1,675,776 rays processed.

DISTRIBUTED RAY TRACING

It is a stochastic sampling method which randomly distributes rays according to the various parameters in an illumination model.

Illumination parameters include pixel area, reflection and refraction directions, camera lens area, and time. Aliasing effects are thus replaced with low-level "noise", which improves picture quality and allows more accurate modeling of surface gloss and translucency, finite camera apertures, finite light sources, and motion-blur displays of moving objects.

modeling of surface gloss and translucency, finite camera apertures, finite light sources, and motion-blur displays of moving objects.

Distributed ray tracing (also referred to as distribution ray tracing) essentially provides a Monte Carlo evaluation of the multiple integrals that occur in an accurate description of surface lighting.

Pixel sampling is accomplished by randomly distributing a number of rays over the pixel surface. Choosing ray positions completely at random, however, can result in the rays clustering together in a small region of the pixel area, and leaving other parts of the pixel unassembled. A better approximation of the light distribution over a pixel area is obtained by using a technique called jittering on a regular subpixel grid. This is usually done by initially dividing the pixel area (a unit square) into the 16 subareas shown in Fig (1-73) and generating a random jitter position in each subarea.

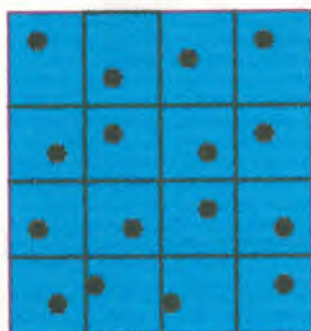


fig (1.73) pixel sampling using 16 subpixel areas and a jittered ray position from the center coordinates for each subarea.

The random ray positions are obtained by jittering the center coordinates of each subarea by small amounts, δ_x and δ_y , where both δ_x and δ_y are assigned values in the interval $(-0.5, 0.5)$. We then choose the ray position in a cell with center coordinates (x, y) as the jitter position $(x + \delta_x, y + \delta_y)$.

Integer codes 1 through 16 are randomly assigned to each of the 16 rays, and a table lookup is used to obtain values for the other parameters

the overall pixel intensity. If the subpixel intensities vary too much, the pixel is further subdivided.

To model camera-lens effects, we set a lens of assigned focal length f in front of the projection plane and distribute the subpixel rays over the lens area. Assuming we have 16 rays per pixel, we can subdivide the lens area into 16 zones. Each ray is then sent to the zone corresponding to its assigned code. The ray position within the zone is set to a jittered position from the zone center. Then the ray is projected into the scene from the jittered zone position through the focal point of the lens.

We locate the focal point for a ray at a distance f from the lens along the line from the center of the subpixel through the lens center, as shown in Fig.14-74.

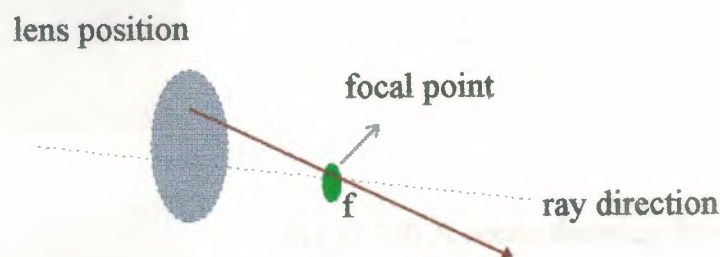


Fig (1-74) Distributing subpixel rays over a camera lens of focal length f .

Objects near the focal plane are projected as sharp images. Objects in front or in back of the focal plane are blurred. To obtain better displays of out-of-focus objects, we increase the number of subpixel rays.

Ray reflections at surface-intersection points are distributed about the specular reflection direction R according to the assigned ray codes. The maximum spread about R is divided into sixteen angular zones, and each ray is reflected in a jittered position from the zone center corresponding to its integer code. We can use the Phong Model, $\cos^{ns} \Phi$, to determine the maximum reflection spread. If the material is transparent, refracted rays are distributed about the transmission direction T in a similar manner.

reflection spread. If the material is transparent, refracted rays are distributed about the transmission direction T in a similar manner.

Extended light sources are handled by distributing a number of shadow rays over the area of the light source, as demonstrated. The light source is divided into zones, and the shadow rays are assigned jitter directions to the various zones.



Fig (1.38) A room scene rendering with distributed ray tracing methods



Fig (1.39) A scene showing the focusing antialiasing, and illumination effects possible with a combination of ray tracing and radiosity methods. Realistic physical models of light illumination were used to generate the refraction effects, including the caustic in the shadow of the glass.

Additionally zones can be weighted according to the intensity of the light source within that zone and the size of the projected zone area on to the object surface. More shadow rays are then sent to zones with higher weights. If light source, penumbra is generated at that surface point.

Additional rays are used for highly blurred objects to reduce calculations we can use bounding boxes or spheres for initial ray intersection tests. That is, we move the bounding object according to the motion require-

Additional rays are used for highly blurred objects to reduce calculations we can use bounding boxes or spheres for initial ray intersection tests. That is, we move the bounding object according to the motion requirements and test for intersection. If the ray does not intersect the bounding object, we do not need to process the individual surfaces within the bounding volume.

RADIOSITY LIGHTING MODEL

Diffuse reflections can be modeled from a surface by considering the radiant energy transfers between surfaces, subject to conservation of energy laws. This method for describing diffuse reflections is generally referred to as the *radiosity model*

BASIC RADIOSITY MODEL

The radiant energy interactions are considered between all surfaces in a scene. Differential amount of radiant energy dB determined by leaving each surface point in the scene and summing the energy contributions over all surfaces to obtain the amount of energy transfer between surfaces.

Intensity I or luminance of the diffuse radiation in direction (θ, ϕ) is the radiant energy per unit time per unit projected area per unit solid angle with units watts/ (meter² * steradians):

$$I = dB / dw \cos \phi$$



Figure 14-88

Image of a constructivist museum rendered with a progressive-refinement radiosity method.

(Courtesy of Shenchang Eric Chen, Stuart I. Feldman, and Julie Dorsey, Program of Computer Graphics, Cornell University. © 1988, Cornell University, Program of Computer Graphics.)



Figure 14-89

Simulation of the stair tower of the Engineering Theory Center Building at Cornell University rendered with a progressive-refinement radiosity method.

(Courtesy of Keith Howie and Ben Trumbore, Program of Computer Graphics, Cornell University. © 1990, Cornell University, Program of Computer Graphics.)



(a)



(b)

Figure 14-90


Simulation of two lighting schemes for the Parisian garret from the Metropolitan Opera's production of *La Bohème*: (a) day view and (b) night view. (Courtesy of Julie Dorsey and Mark Shepard, Program of Computer Graphics, Cornell University. © 1991, Cornell University, Program of Computer Graphics.)

ENVIRONMENT MAPPING

An alternate procedure for modeling global reflections is to define an array of intensity values that describes the environment around a single object or a set of objects. Instead of interobject ray tracing or radiosity calculations to pick up the global specular and diffuse illumination effects, we simply map the *environment array* onto an object in relationship to the viewing direction. This procedure is referred to as environment mapping, also called *reflection mapping* although transparency effects could also be modeled with the environment map.

Environment mapping is sometimes referred to as the "poor person's ray-tracing" method, since it is a fast approximation of the more accurate global-illumination rendering techniques we discussed in the previous two sections.

The environment map is defined over the surface of an enclosing universe. Information in the environment map includes intensity values for light sources, the sky, and other background objects. The enclosing universe is shown as a sphere, but a cube or a cylinder is often used as the enclosing universe.

To render the surface of an object, we project pixel areas onto the surface and then reflect the projected pixel area onto the environment map to pick up the surface-shading attributes for each pixel. If the object is transparent, we can also refract the projected pixel area to the environment map. The environment-mapping process for reflection of a projected pixel area is illustrated in . Pixel intensity is determined by averaging the intensity values within the intersected region of the environment map.

ADDING SURFACE DETAIL

Most objects do not have smooth, even surfaces so we need surface texture to model accurately such objects as brick walls, gravel roads, and shag carpets. In addition, some surfaces contain patterns that must be taken into account in the rendering procedures. The surface of a vase could contain a painted design, a water glass might have the family crested engraved into the surface, a tennis court contains markings for the alleys, service areas, and base line and four lane highway has dividing lines and other markings, such as oil spills and tire skids.

MODEL SURFACE DETAIL WITH POLYGONS

A simple method for adding surface detail is to model structure and patterns with polygon faces. For large scale detail, polygon modeling can give good results. Some examples of large scale detail are squares on a checkboard, dividing lines on a highway, tile patterns on a linoleum floor, floral designs in a smooth low pile rug, panels on a door, and lettering on the side of a panel truck.

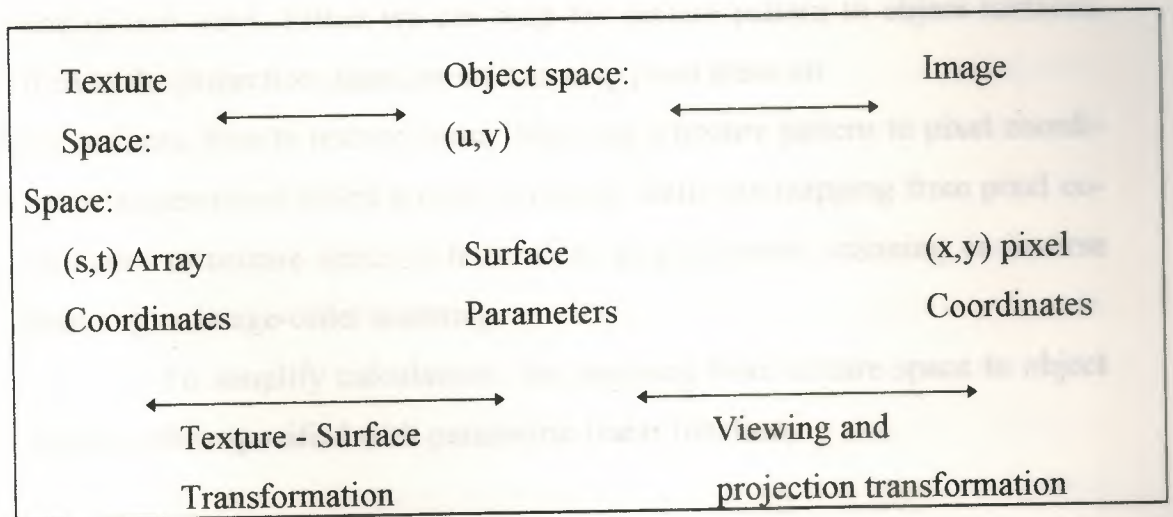


fig (1.40) Coordinate reference systems for texture space, object space, and image space.

Surface-pattern polygons are generally overlaid on a larger surface polygon and are processed with the parent surface. Only the parent polygon is processed by the visible-surface algorithms, but the illumination parameters for the surface detail polygons take precedence over the parent polygon. When intricate or fine surface detail is to be modeled, polygon methods are not practical. For example, it would be difficult to accurately model the surface structure of a raisin with polygon facets.

Texture Mapping

A common method for adding surface detail is to map texture patterns onto the surfaces of objects. The texture pattern may either be defined in a rectangular array or as a procedure that modifies surface intensity values. This approach is referred to as texture mapping or pattern mapping.

Usually, the texture pattern is defined with a rectangular grid of intensity values in a texture space referenced with (s, t) coordinate values, as shown in Fig. 14-94. Surface positions in the scene are referenced with uv object-space coordinates, and pixel positions on the projection plane are referenced in xy Cartesian coordinates. Texture mapping can be accomplished in one of two ways. Either we can map the texture pattern to object surfaces, then to the projection plane; or we can map pixel areas on object surfaces, then to texture space. Mapping a texture pattern to pixel coordinates is sometimes called texture scanning, while the mapping from pixel coordinates to texture space is referred to as pixel-order scanning or inverse scanning or image-order scanning.

To simplify calculations, the mapping from texture space to object space is often specified with parametric linear functions

$$u = f_u(s,t) = a_u s + b_u t + c_u$$

$$v = f_v(s,t) = a_v s + b_v t + c_v$$

The object-to-image space mapping is accomplished with the concatenation of the viewing and projection transformations. A disadvantage of mapping from texture space to pixel space is that a selected texture patch usually does not match up with the pixel boundaries, thus requiring calculation of the fractional area of pixel coverage. Therefore, mapping from pixel space to texture space is the most commonly used texture-mapping method. This avoids pixel subdivision calculations, and allows antialiasing (filtering) procedures to

BUMP MAPPING

It is not a good method to apply texture mapping method to add fine surfaces detail if we want to model the surface roughness that appears on objects such as oranges, strawberries and raisins. The illumination texture pattern usually does not correspond to the illumination direction in the scene A.

Better method for creating surface bumpiness is to apply a perturbation function to the surface normal and then use perturbed normal in the illumination model calculations

FRAME MAPPING

This technique is an extension of bump mapping. In this mapping both the surface normal N and the local coordinate system is attached to N . The local coordinates are defined with a surface-tangent vector T and a binormal vector $B = T \times N$.

Frame mapping is used to model anisotropy surfaces. We orient T along the "grain" of the surface and apply directional perturbations, in addition to bump perturbations in the direction of N . In this way, we can model wood-grain patterns, cross-thread patterns in cloth, and streaks in marble or similar materials. Both bump and directional perturbations can be obtained with table lookups.

CONCLUSION

In general, an object is illuminated with radiant energy from light-emitting sources and from the reflective surfaces of other objects in the scene. Light sources can be modeled as point sources or as distributed (extended) sources. Objects can be either opaque or transparent. And lighting effects can be described in terms of diffuse and specular components for both reflections and refraction's.

An empirical, point light-source, illumination model can be used to describe diffuse reflections with Lambert's cosine law and to describe specular reflections with the *Phong model*. General background (ambient) lighting can be modeled with a fixed intensity level and a coefficient of reflection for each surface. In this basic model, we can approximate transparency effects by combining surface intensities using a transparency coefficient. Accurate geometric modeling of light paths through transparent materials also is obtained by calculating refraction angles using *Snell's law*. Color is incorporated into the model by assigning a triple of RGB values to intensities and surface reflection coefficients. We can also extend the basic model to incorporate distributed light sources, studio lighting effects, and intensity attenuation.

Intensity values calculated with an illumination model must be mapped to the intensity levels available on the display system in use.

A logarithmic intensity scale is used to provide a set of intensity levels with equal perceived brightness. In addition, gamma correction is applied to intensity values to correct for the nonlinearity of display devices. With bilevel monitors, we can use halftone patterns and dithering techniques to simulate a range of intensity values.

Halftone approximations can also be used to increase the number of intensity options on systems that are capable of displaying more than two intensities per pixel. Ordered-dither, error-diffusion, and dot-diffusion methods are used to simulate a range of intensities when the number of points to be plotted in a scene is equal to the number of pixels on the display device.

Surface rendering can be accomplished by applying a basic illumination model to the objects in a scene. We apply an illumination model using either constant-intensity shading, Gouraud shading, or Phong shading.

Constant shading is accurate for polyhedrons or for curved-surface polygon meshes when the viewing and light-source positions are far from the objects in a scene.

Gouraud shading approximates light reflections from curved surfaces by calculating intensity values at polygon vertices and interpolating these intensity values across the polygon facets.

A more accurate, but slower, surface-rendering procedure is *Phong shading*, which interpolates the average normal vectors for polygon vertices over the polygon facets. Then, surface intensities are calculated using the interpolated normal vectors.

Fast Phong shading can be used to speed up the calculations using Taylor series approximations.

Ray tracing provides an accurate method for obtaining global, specular reflection and transmission effects. Pixel rays are traced through a scene, bouncing from object to object while accumulating intensity contributions. A ray-tracing tree is constructed for each pixel, and intensity values are combined from the terminal nodes of the tree back up to the root. Object-intersection calculations in ray tracing can be reduced with space-subdivision methods that test for ray-object intersections only within subregions of the total space.

Distributed (or distribution) ray tracing traces multiple rays per pixel and distributes the rays randomly over the various ray parameters, such as direction and time. This provides an accurate method for modeling surface gloss and translucency, finite camera apertures, distributed light sources, shadow effects, and motion blur.

Radiosity methods provide accurate modeling of diffuse-reflection effects by calculating radiant energy transfer between the various surface patches in a scene. Progressive refinement is used to speed up the radiosity calculations by considering energy transfer from one surface patch at a time.

Highly photorealistic scenes are generated using a combination of ray tracing and radiosity.

A fast method for approximating global illumination effects is *environment mapping*. An environment array is used to store background intensity information for a scene. This array is then mapped to the objects in a scene based on the specified viewing direction.

Surface detail can be added to objects using polygon facets, texture mapping, bump mapping, or frame mapping. Small polygon facets can be overlaid on larger surfaces to provide various kinds of designs. Alternatively, texture patterns can be defined in a two-dimensional array and mapped to object surfaces.

Bump mapping is a means for modeling surface irregularities by applying a bump function to perturb surface normals.

Frame mapping is an extension of bump mapping that allows for horizontal surface variations, as well as vertical variations.

REFERENCES

Computer Graphics (Donald Hearn - M. Pauline Baker). A general discussion of energy propagation, transfer equations, rendering processes, and our perception of light and color is given in Glassner (1994). Algorithms for various surface rendering techniques are presented in Glassner (1990), Arvo (1991), and Kirk (1992). For further discussion of ordered dither, error diffusion, and dot diffusion see Knuth (1987). Additional information on ray-tracing methods can be found in Quek and Hearn (1988), Glassner (1989), Shirley (1990), and Koh and Hearn (1992). Radiosity methods are discussed in Goral et al. (1984), Cohen and Greenberg (1985), Cohen et al. (1988), Wallace, Elmquist, and Haines (1989), Chen et al. (1991), Dorsey, Sillion, and Greenberg (1991), He et al. (1992), Sillion et al. (1991), Schoeneman et al. (1993), and Lischinski, Tampieri, and Greenberg (1993).