duced by choosing values for the error-diffusion parameters that sum to a value less than 1 and by rescaling the matrix values after the dispersion of errors. One way to rescale is to multiply all elements of **M** by 0.8 and then add 0.1. Another method for improving picture quality is to alternate the scanning of matrix rows from right to left and left to right.

A variation on the error-diffusion method is *dot diffusion*. In this method, the m by n array of intensity values is divided into 64 classes numbered from 0 to 63, as shown in Fig. 14-43. The error between a matrix value and the displayed intensity is then distributed only to those neighboring matrix elements that have a larger class number. Distribution of the 64 class numbers is based on minimizing the number of elements that are completely surrounded by elements with a lower class number, since this would tend to direct all errors of the surrounding elements to that one position.

14-5 POLYGON-RENDERING METHODS

In this section, we consider the application of an illumination model to the rendering of standard graphics objects: those formed with polygon surfaces. The objects are usually polygon-mesh approximations of curved-surface objects, but they may also be polyhedra that are not curved-surface approximations. Scan-line algorithms typically apply a lighting model to obtain polygon surface rendering in one of two ways. Each polygon can be rendered with a single intensity, or the intensity can be obtained at each point of the surface using an interpolation scheme.

Constant-Intensity Shading

A fast and simple method for rendering an object with polygon 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 quickly displaying the general appearance of a curved surface, as in Fig. 14-47.

In general, flat shading of polygon facets provides an accurate rendering for an object if all of the following assumptions are valid:

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

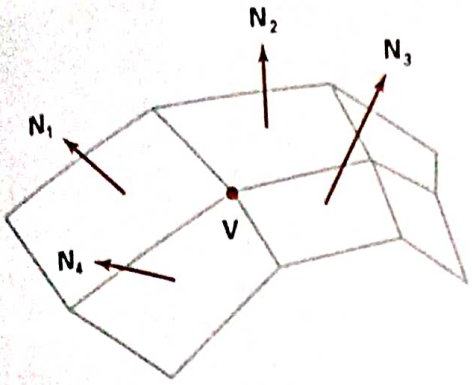


Figure 14-44
The normal vector at vertex **V** is calculated as the average of the surface normals for each polygon sharing that vertex.

- All light sources illuminating the object are sufficiently far from the surface so that $\mathbf{N} \cdot \mathbf{L}$ and the attenuation function are constant over the surface.
- The viewing position is sufficiently far from the surface so that $\mathbf{V} \cdot \mathbf{R}$ 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, say, at the center of the polygon.

Gouraud Shading

This **intensity-interpolation** scheme, developed by Gouraud and generally 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 Gouraud shading 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, as illustrated in Fig. 14-44. Thus, for any vertex position **V**, we obtain the unit vertex normal with the calculation

$$\mathbf{N}_V = \frac{\sum_{k=1}^{n} \mathbf{N}_k}{\left| \sum_{k=1}^{n} \mathbf{N}_k \right|} \tag{14-37}$$

Once we have the vertex normals, we can determine the intensity at the vertices from a lighting model.

Figure 14-45 demonstrates the next step: interpolating intensities along the polygon edges. For each scan line, the intensity at the intersection of the scan line with a polygon edge is linearly interpolated from the intensities at the edge endpoints. For the example in Fig. 14-45, the polygon edge with endpoint vertices at positions 1 and 2 is intersected by the scan line at point 4. A fast method for obtaining the intensity at point 4 is to interpolate between intensities I_1 and I_2 using only the vertical displacement of the scan line:

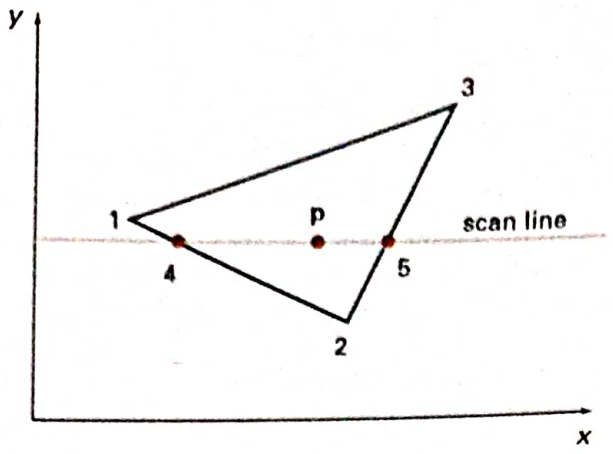


Figure 14-45
For Gouraud shading, the intensity at point 4 is linearly interpolated from the intensities at vertices 1 and 2. The intensity at point 5 is linearly interpolated from intensities at vertices 2 and 3. An interior point **p** is then assigned an intensity value that is linearly interpolated from intensities at positions 4 and 5.

$$I_4 = \frac{y_4 - y_2}{y_1 - y_2} I_1 + \frac{y_1 - y_4}{y_1 - y_2} I_2 \tag{14-38}$$

Similarly, intensity at the right intersection of this scan line (point 5) is interpolated from intensity values at vertices 2 and 3. Once these bounding intensities are established for a scan line, an interior point (such as point **p** in Fig. 14-45) is interpolated from the bounding intensities at points 4 and 5 as

$$I_p = \frac{x_5 - x_p}{x_5 - x_4} I_4 + \frac{x_p - x_4}{x_5 - x_4} I_5 \tag{14-39}$$

Incremental calculations are used to obtain successive edge intensity values between scan lines and to obtain successive intensities along a scan line. As shown in Fig. 14-46, if the intensity at edge position (x, y) is interpolated as

$$I = \frac{y - y_2}{y_1 - y_2} I_1 + \frac{y_1 - y}{y_1 - y_2} I_2 \tag{14-40}$$

then we can obtain the intensity along this edge for the next scan line, $y - 1$, as

$$I' = I + \frac{I_2 - I_1}{y_1 - y_2} \tag{14-41}$$

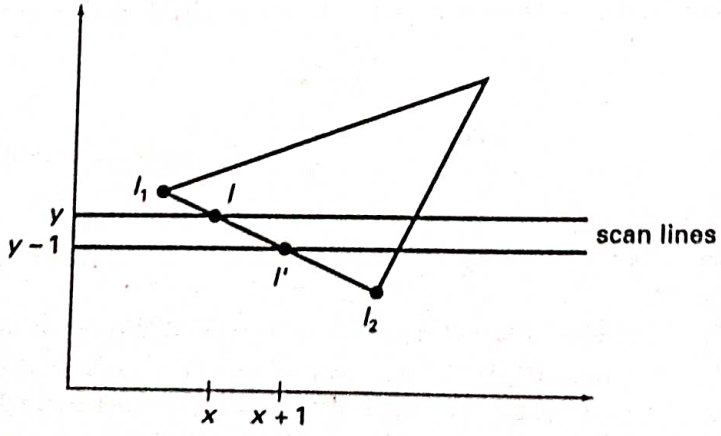


Figure 14-46
Incremental interpolation of intensity values along a polygon edge for successive scan lines.

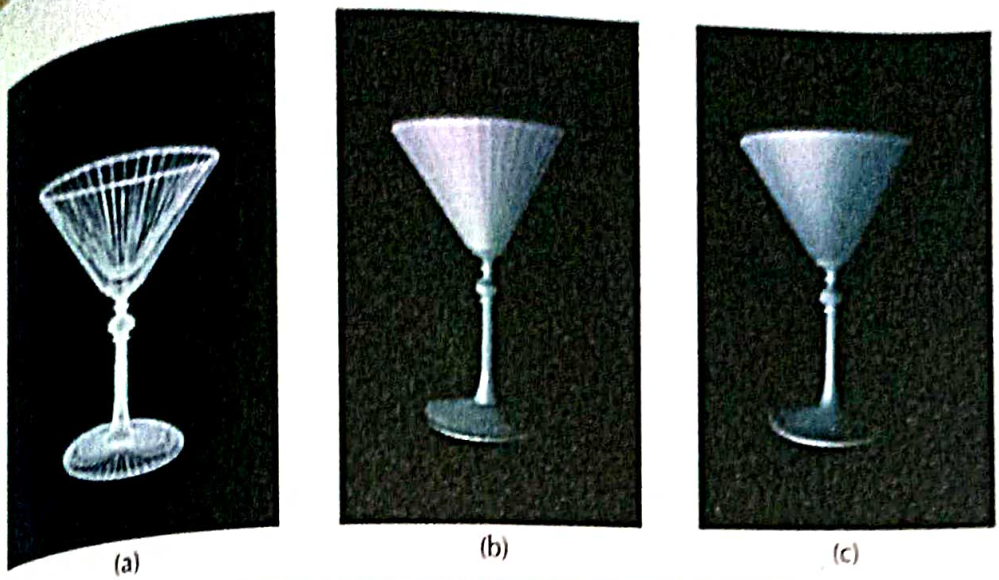


Figure 14-47
A polygon mesh approximation of an object (a) is rendered with flat shading (b) and with Gouraud shading (c).

Similar calculations 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 the Gouraud method appears in Fig. 14-47.

Gouraud shading 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 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 greatly reduces the Mach-band effect.

A polygon surface is rendered using Phong shading by carrying out 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.

Interpolation of surface normals along a polygon edge between two vertices is illustrated in Fig. 14-48. The normal vector **N** for the scan-line intersection point along the edge between vertices 1 and 2 can be obtained by vertically interpolating between edge endpoint normals:

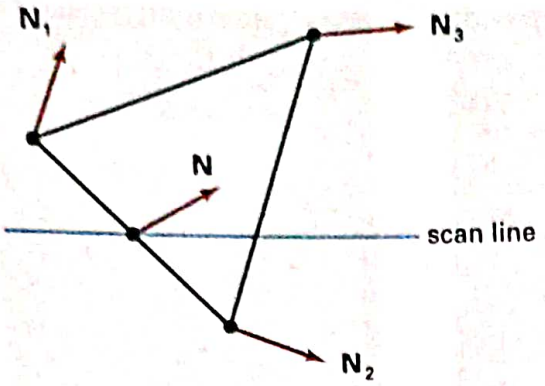


Figure 14-48
Interpolation of surface normals along a polygon edge.

$$\mathbf{N} = \frac{y - y_2}{y_1 - y_2}\mathbf{N}_1 + \frac{y_1 - y}{y_1 - y_2}\mathbf{N}_2 \tag{14-42}$$

Incremental methods are used to evaluate normals between scan lines and along each individual scan line. At each pixel position along a scan line, the illumination model is applied to determine the surface intensity at that point.

Intensity calculations using an approximated normal vector at each point along the scan line produce more accurate results than the direct interpolation of intensities, as in Gouraud shading. The trade-off, however, is that Phong shading requires considerably more calculations.

Fast Phong Shading

Surface rendering with Phong shading can be speeded up by using approximations in the illumination-model calculations of normal vectors. **Fast Phong shading** approximates the intensity calculations using a Taylor-series expansion and triangular surface patches.

Since Phong shading interpolates normal vectors from vertex normals, we can express the surface normal **N** at any point (x, y) over a triangle as

$$\mathbf{N} = \mathbf{A}x + \mathbf{B}y + \mathbf{C} \tag{14-43}$$

where vectors **A**, **B**, and **C** are determined from the three vertex equations:

$$\mathbf{N}_k = \mathbf{A}x_k + \mathbf{B}y_k + \mathbf{C}, \qquad k = 1, 2, 3 \tag{14-44}$$

with (x_k, y_k) denoting a vertex position.

Omitting the reflectivity and attenuation parameters, we can write the calculation for light-source diffuse reflection from a surface point (x, y) as

$$\begin{aligned} I_{\text{diff}}(x, y) &= \frac{\mathbf{L} \cdot \mathbf{N}}{|\mathbf{L}|\,|\mathbf{N}|} \\ &= \frac{\mathbf{L} \cdot (\mathbf{A}x + \mathbf{B}y + \mathbf{C})}{|\mathbf{L}|\,|\mathbf{A}x + \mathbf{B}y + \mathbf{C}|} \\ &= \frac{(\mathbf{L} \cdot \mathbf{A})x + (\mathbf{L} \cdot \mathbf{B})y + \mathbf{L} \cdot \mathbf{C}}{|\mathbf{L}|\,|\mathbf{A}x + \mathbf{B}y + \mathbf{C}|} \end{aligned} \tag{14-45}$$

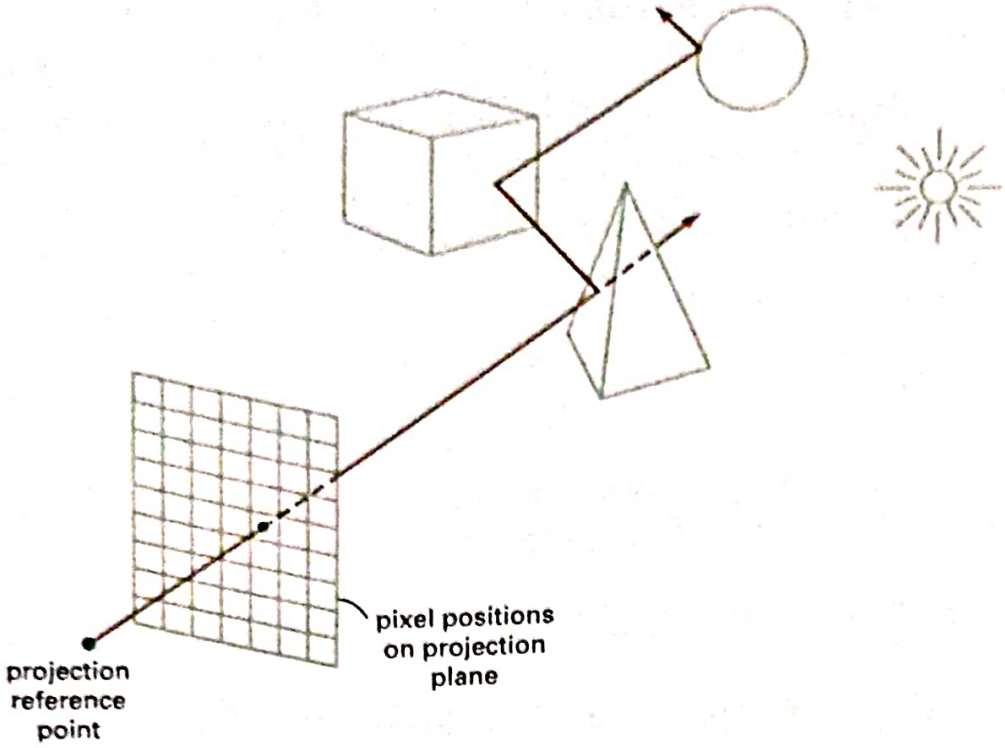


Figure 14-49
Tracing a ray from the projection reference point through a pixel position with multiple reflections and transmissions.

Basic Ray-Tracing Algorithm

We first set up a coordinate system with the pixel positions designated in the *xy* plane. The scene description is given in this reference frame (Fig. 14-51). From the center of projection, we then determine a ray path that passes through the center of each screen-pixel position. Illumination effects accumulated along this ray path are then assigned to the pixel. This rendering approach is based on the principles of geometric optics. Light rays from the surfaces in a scene emanate in all directions, and some will pass through the pixel positions in the projection plane. Since there are an infinite number of ray paths, we determine the contributions to a particular pixel by tracing a light path backward from the pixel to the scene. We first consider the basic ray-tracing algorithm with one ray per pixel, which is equivalent to viewing the scene through a pinhole camera.

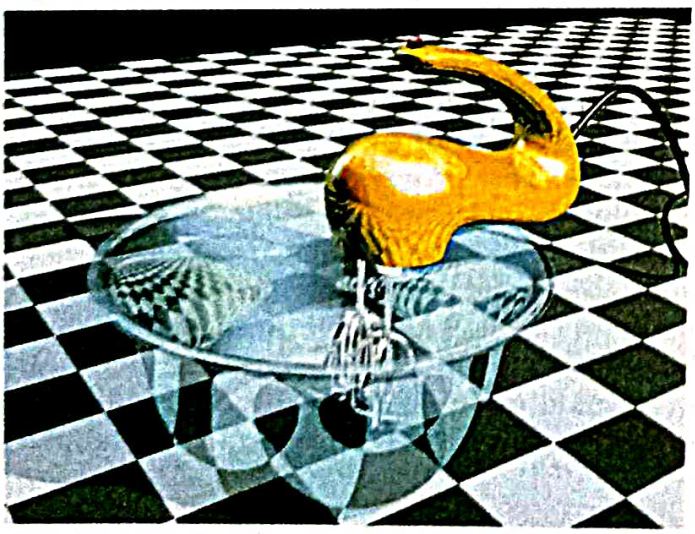


Figure 14-50
A ray-traced scene, showing global reflection and transmission illumination effects from object surfaces. (*Courtesy of Evans & Sutherland.*)

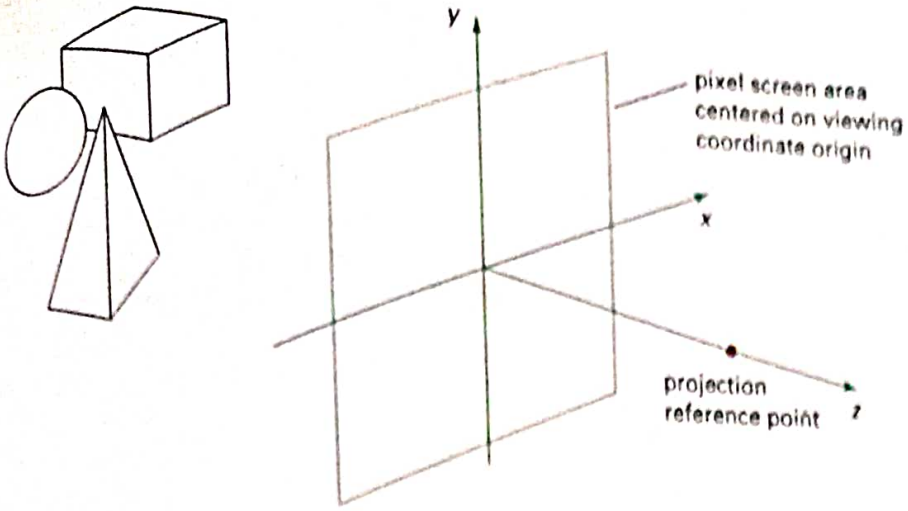


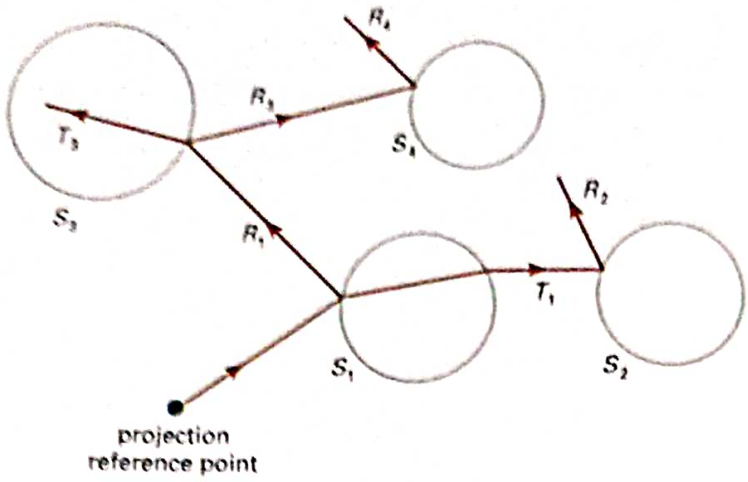
Figure 14-51
Ray-tracing coordinate-reference frame.

For each pixel ray, we test each surface in the scene to determine if it is intersected by the ray. If a surface is intersected, we calculate the distance from the pixel to the surface-intersection point. The smallest calculated intersection distance identifies the visible surface for that pixel. We then reflect the ray off the visible surface along a specular path (angle of reflection equals angle of incidence). If the surface is transparent, we also send a ray through the surface in the refraction direction. Reflection and refraction rays are referred to as *secondary rays*.

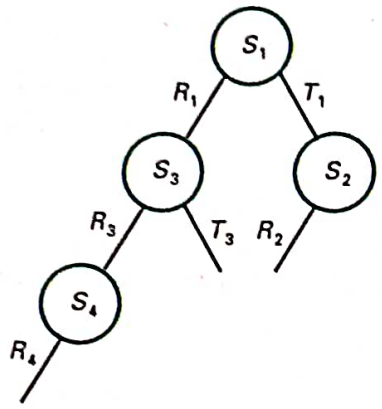
This procedure is repeated for each secondary ray: Objects are tested for intersection, and the nearest surface along a secondary ray path is used to recursively produce the next generation of reflection and refraction paths. As the rays from a pixel ricochet through the scene, each successively intersected surface is added to a binary *ray-tracing tree*, as shown in Fig. 14-52. We use left branches in the tree to represent reflection paths, and right branches represent transmission paths. Maximum depth of the ray-tracing trees can be set as a user option, or it can be determined by the amount of storage available. A path in the tree is then terminated if it reaches the preset maximum or if the ray strikes a light source.

The intensity assigned to a pixel is then determined by accumulating the intensity contributions, starting at the bottom (terminal nodes) of its ray-tracing tree. Surface intensity from each node in the tree is attenuated by the distance from the "parent" surface (next node up the tree) and added to the intensity of the parent surface. Pixel intensity is then the sum of the attenuated intensities at the root node of the ray tree. If no surfaces are intersected by a pixel ray, the ray-tracing tree is empty and the pixel is assigned the intensity value of the background. If a pixel ray intersects a nonreflecting light source, the pixel can be assigned the intensity of the source, although light sources are usually placed beyond the path of the initial rays.

Figure 14-53 shows a surface intersected by a ray and the unit vectors needed for the reflected light-intensity calculations. Unit vector $\mathbf{u}$ is in the direction of the ray path, $\mathbf{N}$ is the unit surface normal, $\mathbf{R}$ is the unit reflection vector, $\mathbf{L}$ is the unit vector pointing to the light source, and $\mathbf{H}$ is the unit vector halfway between $\mathbf{V}$ (opposite to $\mathbf{u}$) and $\mathbf{L}$. The path along $\mathbf{L}$ is referred to as the **shadow ray**. If any object intersects the shadow ray between the surface and the point light



(a)



(b)

Figure 14-52
(a) Reflection and refraction ray paths through a scene for a screen pixel. (b) Binary ray-tracing tree for the paths shown in (a).

source, the surface is in shadow with respect to that source. Ambient light at the surface is calculated as $k_a I_a$; diffuse reflection due to the source is proportional to $k_d(\mathbf{N} \cdot \mathbf{L})$; and the specular-reflection component is proportional to $k_s(\mathbf{H} \cdot \mathbf{N})^{n_s}$. As discussed in Section 14-2, the specular-reflection direction for the secondary ray path R depends on the surface normal and the incoming ray direction:

$$\mathbf{R} = \mathbf{u} - (2\mathbf{u} \cdot \mathbf{N})\mathbf{N} \tag{14-49}$$

For a transparent surface, we also need to obtain intensity contributions from light transmitted through the material. We can locate the source of this contribution by tracing a secondary ray along the transmission direction T, as shown in Fig. 14-54. The unit transmission vector can be obtained from vectors u and N as

$$\mathbf{T} = \frac{\eta_i}{\eta_r}\mathbf{u} - (\cos\theta_r - \frac{\eta_i}{\eta_r}\cos\theta_i)\mathbf{N} \tag{14-50}$$

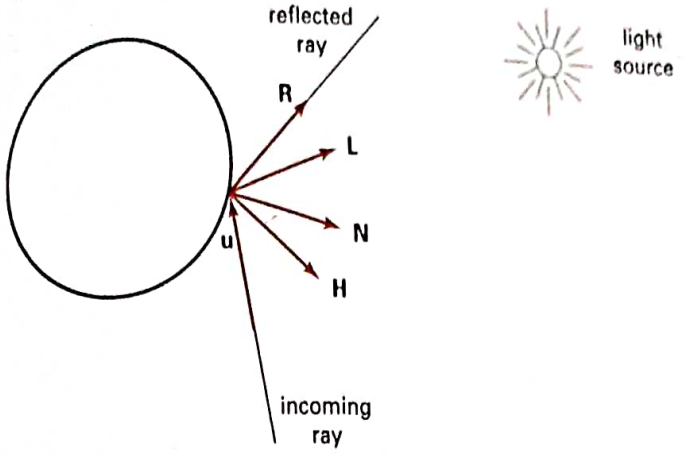


Figure 14-53
Unit vectors at the surface of an object intersected by an incoming ray along direction **u**.

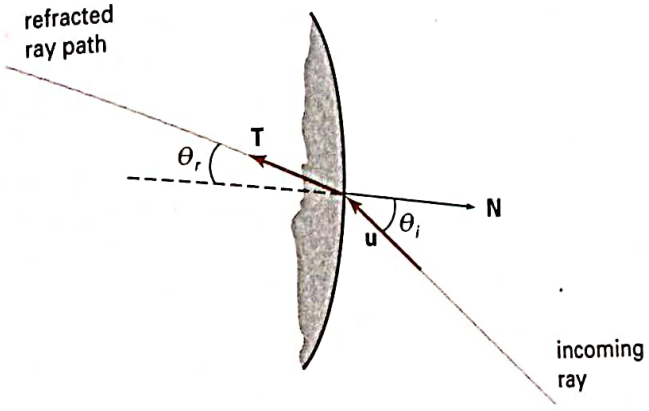


Figure 14-54
Refracted ray path **T** through a transparent material.

Parameters η_i and η_r are the indices of refraction in the incident material and the refracting material, respectively. Angle of refraction θ_r can be calculated from Snell's law:

$$\cos\theta_r = \sqrt{1 - \left(\frac{\eta_i}{\eta_r}\right)^2 (1 - \cos^2\theta_i)} \qquad (14\text{-}51)$$

Ray-Surface Intersection Calculations

A ray can be described with an initial position $\mathbf{P}_0$ and unit direction vector **u**, as illustrated in Fig. 14-55. The coordinates of any point **P** along the ray at a distance *s* from $\mathbf{P}_0$ is computed from the **ray equation**:

$$\mathbf{P} = \mathbf{P}_0 + s\mathbf{u} \qquad (14\text{-}52)$$

Initailly, $\mathbf{P}_0$ can be set to the position of the pixel on the projection plane, or it could be chosen to be the projection reference point. Unit vector **u** is initially ob-

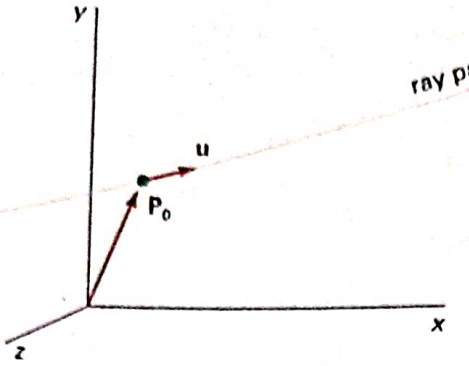


Figure 14-55
Describing a ray with an initial-position vector $\mathbf{P}_0$ and unit direction vector $\mathbf{u}$.

tained from the position of the pixel through which the ray passes and the projection reference point:

$$\mathbf{u} = \frac{\mathbf{P}_{pix} - \mathbf{P}_{prp}}{|\mathbf{P}_{pix} - \mathbf{P}_{prp}|} \tag{14-53}$$

At each intersected surface, vectors $\mathbf{P}_0$ and $\mathbf{u}$ are updated for the secondary rays at the ray-surface intersection point. For the secondary rays, reflection direction for $\mathbf{u}$ is $\mathbf{R}$ and the transmission direction is $\mathbf{T}$. To locate surface intersections, we simultaneously solve the ray equation and the surface equation for the individual objects in the scene.

The simplest objects to ray trace are spheres. If we have a sphere of radius r and center position $\mathbf{P}_c$ (Fig. 14-56), then any point $\mathbf{P}$ on the surface must satisfy the sphere equation:

$$|\mathbf{P} - \mathbf{P}_c|^2 - r^2 = 0 \tag{14-54}$$

Substituting the ray equation 14-52, we have

$$|\mathbf{P}_0 + s\mathbf{u} - \mathbf{P}_c|^2 - r^2 = 0 \tag{14-55}$$

If we let $\Delta\mathbf{P} = \mathbf{P}_c - \mathbf{P}_0$ and expand the dot product, we obtain the quadratic equation

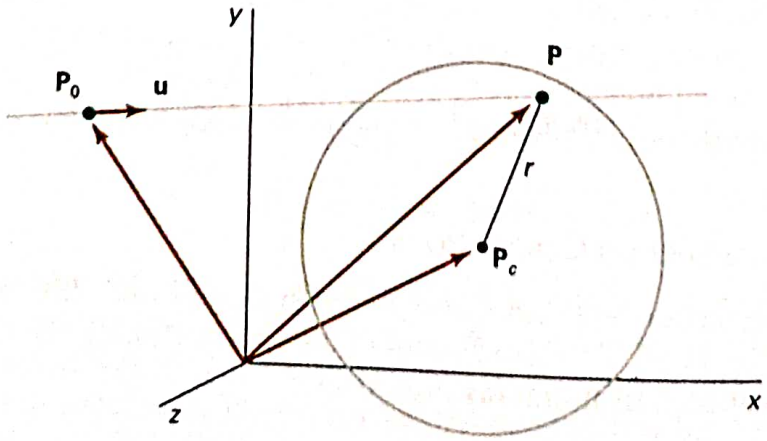


Figure 14-56
A ray intersecting a sphere with radius r centered on position $\mathbf{P}_c$.

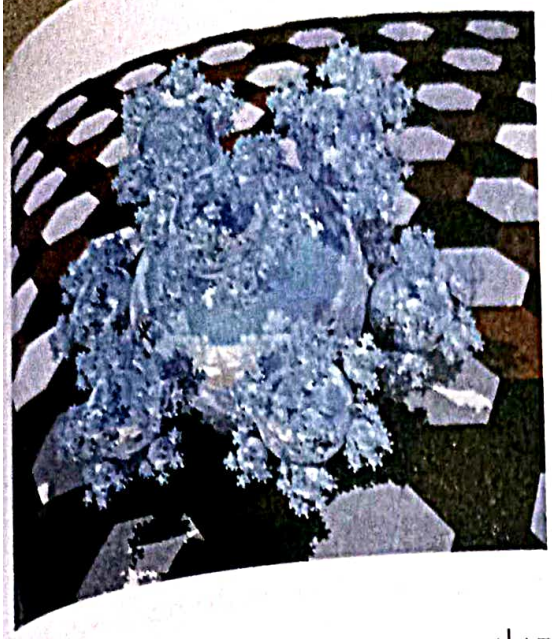


Figure 14-57
A "sphereflake" rendered with ray tracing using 7381 spheres and 3 light sources. (*Courtesy of Eric Haines, 3D/EYE Inc.*)

$$s^2 - 2(\mathbf{u} \cdot \Delta\mathbf{P})s + (|\Delta\mathbf{P}|^2 - r^2) = 0 \tag{14-56}$$

whose solution is

$$s = \mathbf{u} \cdot \Delta\mathbf{P} \pm \sqrt{(\mathbf{u} \cdot \Delta\mathbf{P})^2 - |\Delta\mathbf{P}|^2 + r^2} \tag{14-57}$$

If the discriminant is negative, the ray does not intersect the sphere. Otherwise, the surface-intersection coordinates are obtained from the ray equation 14-52 using the smaller of the two values from Eq. 14-57.

For small spheres that are far from the initial ray position, Eq. 14-57 is susceptible to roundoff errors. That is, if

$$r^2 << |\Delta\mathbf{P}|^2$$

we could lose the r^2 term in the precision error of $|\Delta\mathbf{P}|^2$. We can avoid this for most cases by rearranging the calculation for distance s as

$$s = \mathbf{u} \cdot \Delta\mathbf{P} \pm \sqrt{r^2 - |\Delta\mathbf{P} - (\mathbf{u} \cdot \Delta\mathbf{P})\mathbf{u}|^2} \tag{14-58}$$

Figure 14-57 shows a snowflake pattern of shiny spheres rendered with ray tracing to display global surface reflections.

Polyhedra require more processing than spheres to locate surface intersections. For that reason, it is often better to do an initial intersection test on a bounding volume. For example, Fig. 14-58 shows a polyhedron bounded by a sphere. If a ray does not intersect the sphere, we do not need to do any further testing on the polyhedron. But if the ray does intersect the sphere, we first locate "front" faces with the test

$$\mathbf{u} \cdot \mathbf{N} < 0 \tag{14-59}$$

where $\mathbf{N}$ is a surface normal. For each face of the polyhedron that satisifies inequality 14-59, we solve the plane equation

$$\mathbf{N} \cdot \mathbf{P} = -D \tag{14-60}$$

for surface position $\mathbf{P}$ that also satisfies the ray equation 14-52. Here, $\mathbf{N} = (A, B, C)$

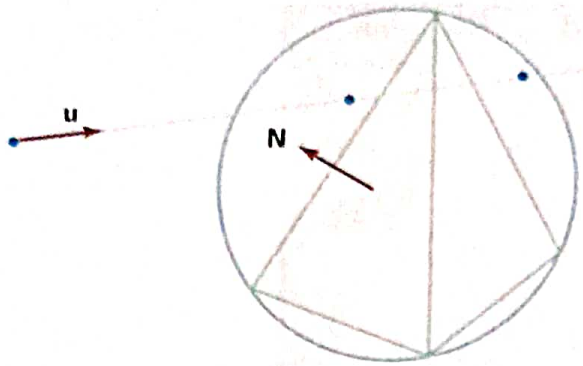


Figure 14-58
Polyhedron enclosed by a bounding sphere.

and D is the fourth plane parameter. Position $\mathbf{P}$ is both on the plane and on the ray path if

$$\mathbf{N} \cdot (\mathbf{P}_0 + s\mathbf{u}) = -D \tag{14-61}$$

And the distance from the initial ray position to the plane is

$$s = -\frac{D + \mathbf{N} \cdot \mathbf{P}_0}{\mathbf{N} \cdot \mathbf{u}} \tag{14-62}$$

This gives us a position on the infinite plane that contains the polygon face, but this position may not be inside the polygon boundaries (Fig. 14-59). So we need to perform an "inside-outside" test (Chapter 3) to determine whether the ray intersected this face of the polyhedron. We perform this test for each face satisfying inequality 14-59. The smallest distance s to an inside point identifies the intersected face of the polyhedron. If no intersection positions from Eq. 14-62 are inside points, the ray does not intersect the object.

Similar procedures are used to calculate ray-surface intersection positions for other objects, such as quadric or spline surfaces. We combine the ray equation with the surface definition and solve for parameter s. In many cases, numerical root-finding methods and incremental calculations are used to locate intersection

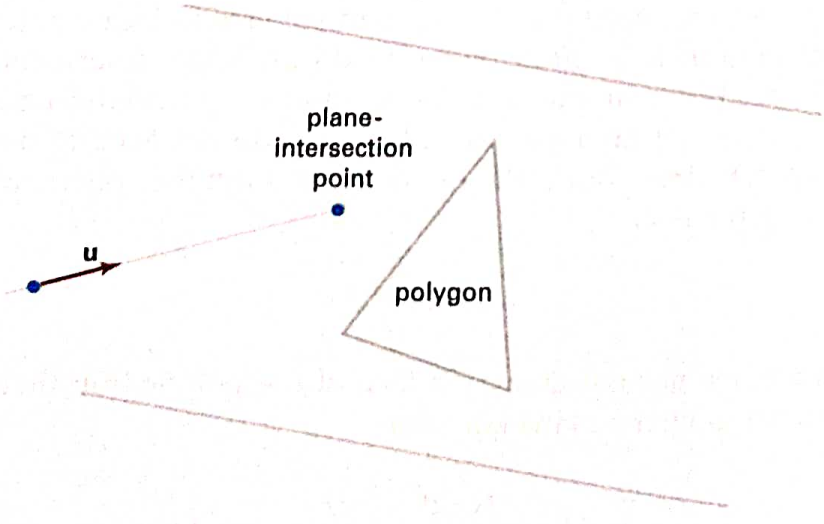


Figure 14-59
Ray intersection with the plane of a polygon.

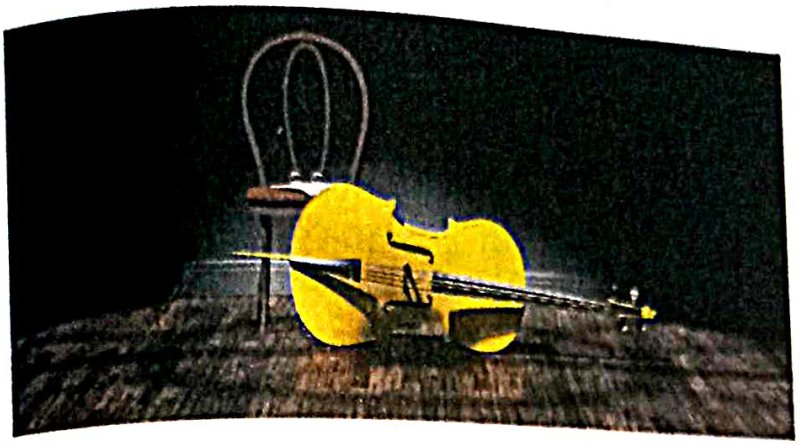


Figure 14-60
A ray-traced scene showing global reflection of surface-texture patterns. (*Courtesy of Sun Microsystems.*)

points over a surface. Figure 14-60 shows a ray-traced scene containing multiple objects and texture patterns.

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 (Fig. 14-61). 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

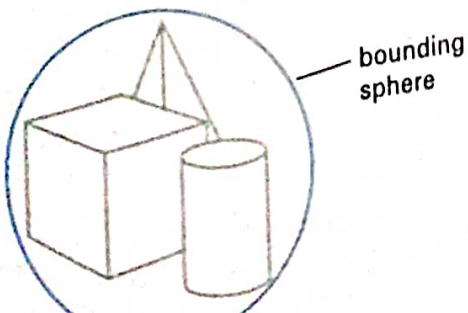


Figure 14-61
A group of objects enclosed within a bounding sphere.

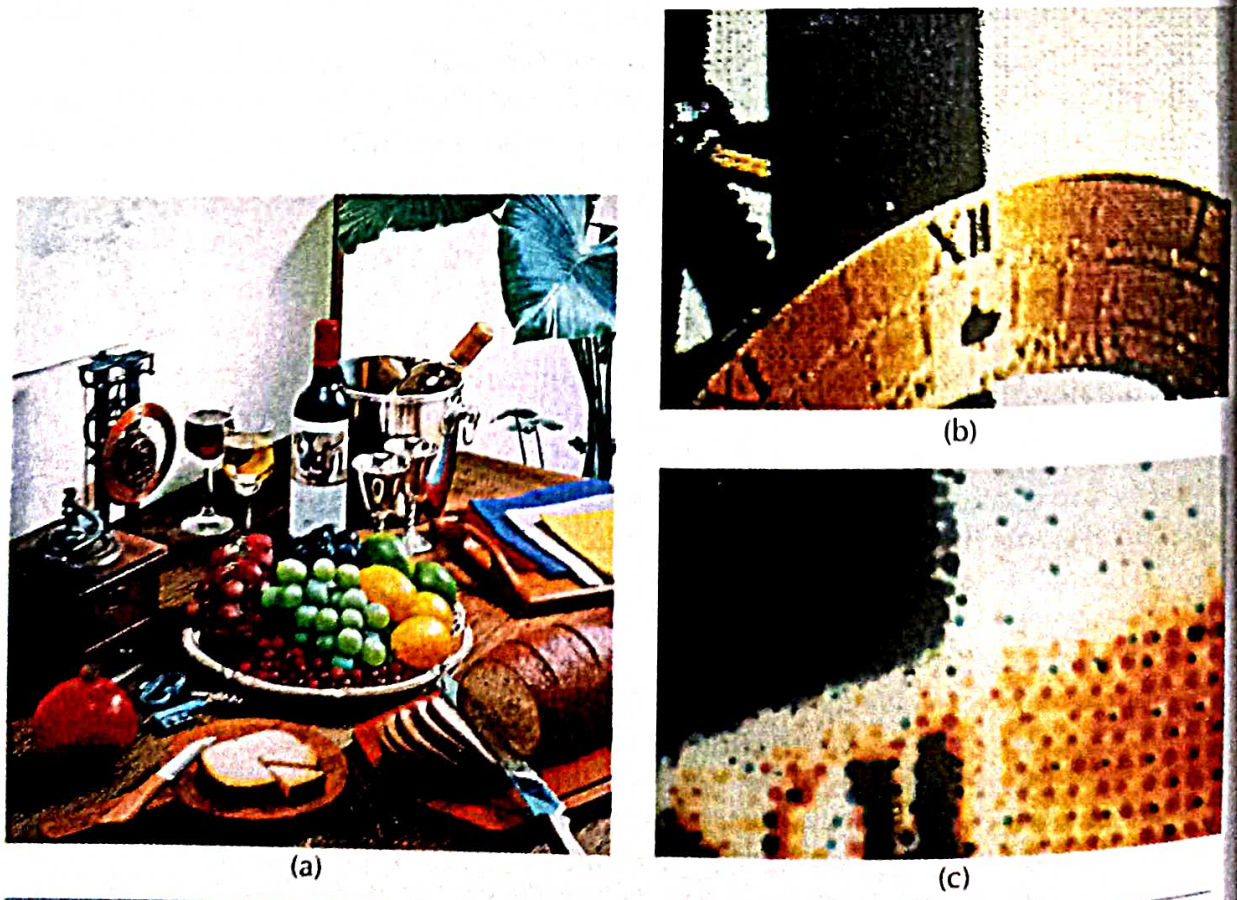
14-4 HALFTONE PATTERNS AND DITHERING TECHNIQUES

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). Figure 14-34 shows an enlarged section of a gray-scale halftone reproduction. Color halftones are printed using dots of various sizes and colors, as shown in Fig. 14-35. 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



(a) (b) (c)

Figure 14-35
Color halftone dot patterns. The top half of the clock in the color halftone (a) is enlarged by a factor of 10 in (b) and by a factor of 50 in (c).

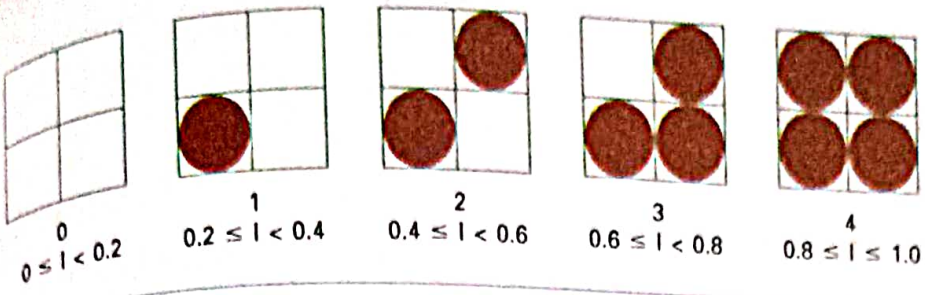


Figure 14-36
A 2 by 2 pixel grid used to display five intensity levels on a bilevel system. The intensity values that would be mapped to each grid are listed below each pixel pattern.

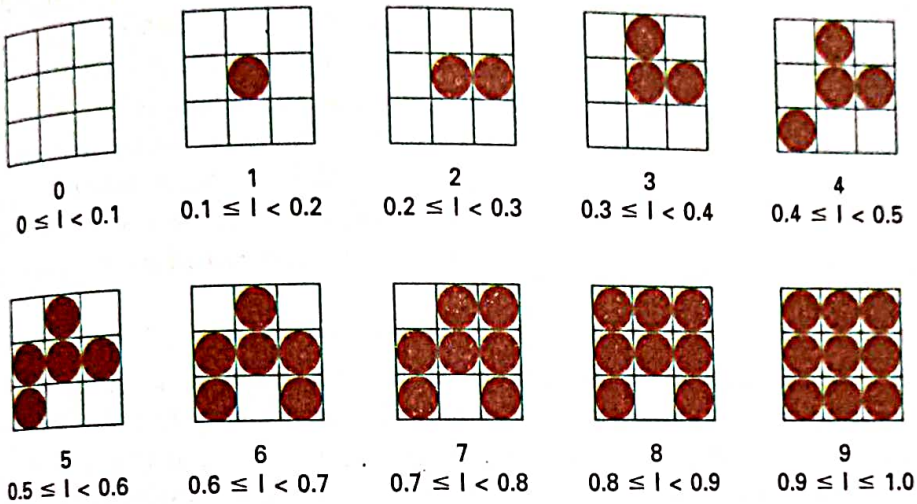


Figure 14-37
A 3 by 3 pixel grid can be used to display 10 intensities on a bilevel system. The intensity values that would be mapped to each grid are listed below each pixel pattern.

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 pixels for each grid on a bilevel system, we can represent $n^2 + 1$ intensity levels. Figure 14-36 shows one way to set up pixel patterns to represent five intensity levels that could be used with a bilevel system. In pattern 0, all pixels are turned off; in pattern 1, one pixel is turned on; and in pattern 4, all four pixels are turned on. An intensity value I in a scene is mapped to a particular pattern according to the range listed below each grid shown in the figure. Pattern 0 is used for $0 \le I < 0.2$, pattern 1 for $0.2 \le I < 0.4$, and pattern 4 is used for $0.8 \le I \le 1.0$.

With 3 by 3 pixel grids on a bilevel system, we can display 10 intensity levels. One way to set up the 10 pixel patterns for these levels is shown in Fig. 14-37. Pixel positions are chosen at each level so that the patterns approximate the increasing circle sizes used in halftone reproductions. That is, the "on" pixel positions are near the center of the grid for lower intensity levels and expand outward as the intensity level increases.

For any pixel-grid size, we can represent the pixel patterns for the various possible intensities with a "mask" of pixel position numbers. As an example, the following mask can be used to generate the nine 3 by 3 grid patterns for intensity levels above 0 shown in Fig. 14-37.

$$\begin{bmatrix} 8 & 3 & 7 \\ 5 & 1 & 2 \\ 4 & 9 & 6 \end{bmatrix} \tag{14-30}$$

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 (Figs. 14-36 and 14-37). 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. 14-38(a) than by any of the symmetrical arrangements in Fig. 14-38(b). 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 effectly reproduced. Therefore, a grid pattern with a single "on" pixel or one with isolated "on" pixels, as in Fig. 14-39, should be avoided.

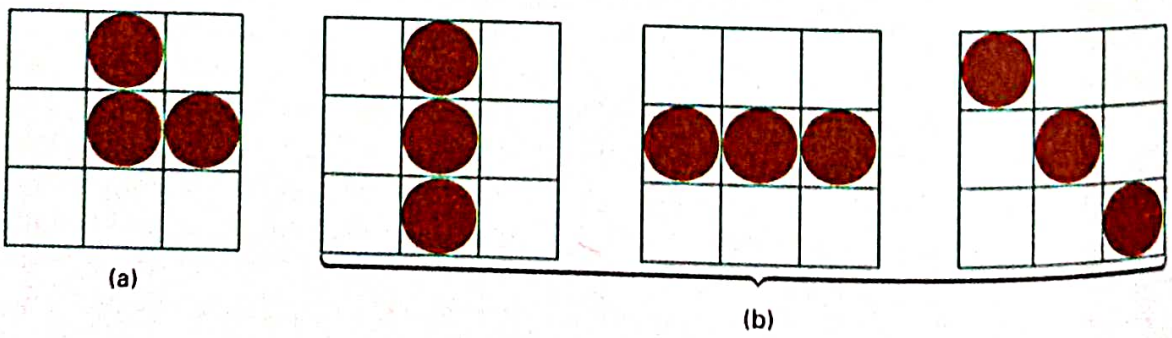


Figure 14-38
For a 3 by 3 pixel grid, pattern (a) is to be preferred to the patterns in (b) for representing the third intensity level above 0.

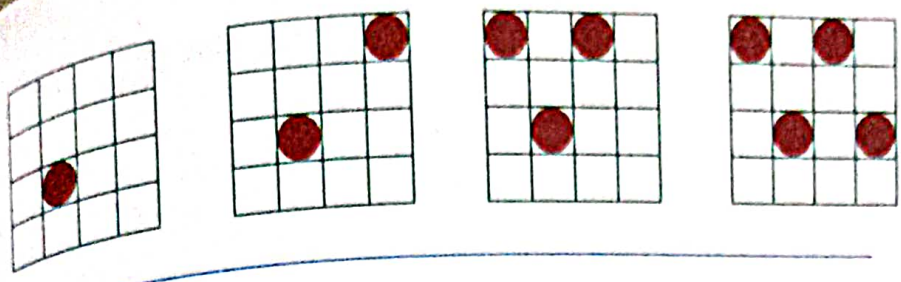


Figure 14-39
Halftone grid patterns with isolated pixels that cannot be effectively reproduced on some hard-copy devices.

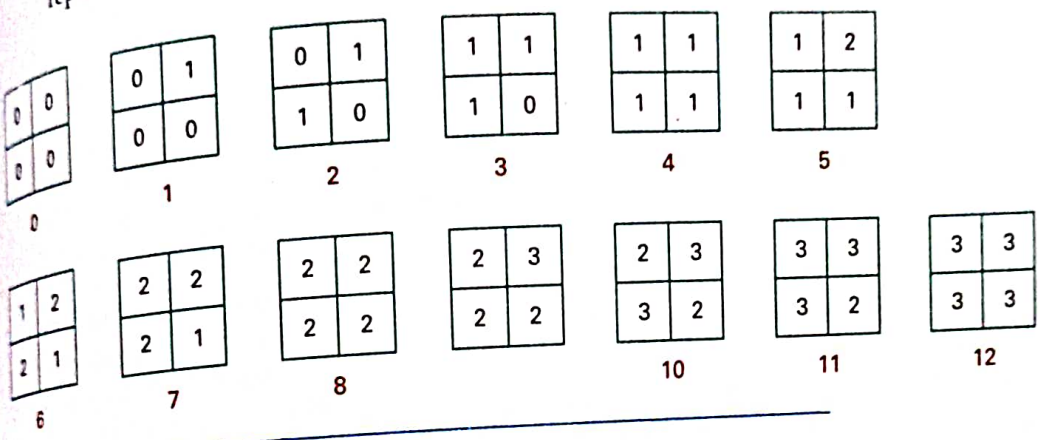


Figure 14-40
Intensity levels 0 through 12 obtained with halftone approximations using 2 by 2 pixel grids on a four-level system.

Halftone approximations also can be used to increase the number of intensity options on systems that are capable of displaying more than two intensities per pixel. For example, on a system that can display four intensity levels per pixel, we can use 2 by 2 pixel grids to extend the available intensity levels from 4 to 13. In Fig. 14-36, the four grid patterns above zero now represent several levels each, since each pixel position can display three intensity values above zero. Figure 14-40 shows one way to assign the pixel intensities to obtain the 13 distinct levels. Intensity levels for individual pixels are labeled 0 through 3, and the overall levels for the system are labeled 0 through 12.

Similarly, we can use pixel-grid patterns to increase the number of intensities that can be displayed on a color system. As an example, suppose we have a three-bit per pixel RGB system. This gives one bit per color gun in the monitor, providing eight colors (including black and white). Using 2 by 2 pixel-grid patterns, we now have 12 phosphor dots that can be used to represent a particular color value, as shown in Fig. 14-41. Each of the three RGB colors has four phosphor dots in the pattern, which allows five possible settings per color. This gives us a total of 125 different color combinations.

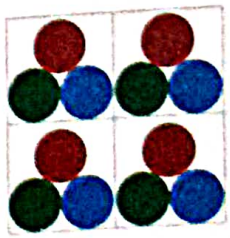


Figure 14-41
An RGB 2 by 2 pixel-grid pattern.

Dithering Techniques

The term **dithering** is used in various contexts. Primarily, it refers to techniques for approximating halftones without reducing resolution, as pixel-grid patterns do. But 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 added to pixel intensities to break up contours are often referred to as *dither noise*. Various algorithms have been used to generate the ran-

dom distributions. The effect is to add noise over an entire picture, which tends to soften intensity boundaries.

Ordered-dither methods generate intensity variations with a one-to-one mapping of points in a scene to the display pixels. To obtain n^2 intensity levels, we set up an n by n dither matrix D_n, whose elements are distinct positive integers in the range 0 to $n^2 - 1$. For example, we can generate four intensity levels with

$$D_2 = \begin{bmatrix} 3 & 1 \\ 0 & 2 \end{bmatrix} \tag{14-31}$$

and we can generate nine intensity levels with

$$D_3 = \begin{bmatrix} 7 & 2 & 6 \\ 4 & 0 & 1 \\ 3 & 8 & 5 \end{bmatrix} \tag{14-32}$$

The matrix elements for D_2 and D_3 are in the same order as the pixel mask for setting up 2 by 2 and 3 by 3 pixel grids, respectively. For a bilevel system, we then determine display intensity values by comparing input intensities to the matrix elements. Each input intensity is first scaled to the range $0 \le I \le n^2$. If the intensity I is to be applied to screen position (x, y), we calculate row and column numbers for the dither matrix as

$$i = (x \bmod n) + 1, \qquad j = (y \bmod n) + 1 \tag{14-33}$$

If $I > D_n(i,j)$, we turn on the pixel at position (x, y). Otherwise, the pixel is not turned on.

Elements of the dither matrix are assigned in accordance with the guidelines discussed for pixel grids. That is, we want to minimize added visual effect in a displayed scene. Order dither produces constant-intensity areas identical to those generated with pixel-grid patterns when the values of the matrix elements correspond to the grid mask. Variations from the pixel-grid displays occur at boundaries of the intensity levels.

Typically, the number of intensity levels is taken to be a multiple of 2. Higher-order dither matrices are then obtained from lower-order matrices with the recurrence relation:

$$D_n = \begin{bmatrix} 4D_{n/2} + D_2(1,1)U_{n/2} & 4D_{n/2} + D_2(1,2)U_{n/2} \\ 4D_{n/2} + D_2(2,1)U_{n/2} & 4D_{n/2} + D_2(2,2)U_{n/2} \end{bmatrix} \tag{14-34}$$

assuming $n \ge 4$. Parameter $U_{n/2}$ is the "unity" matrix (all elements are 1). As an example, if D_2 is specified as in Eq. 14-31, then recurrence relation 14-34 yields

$$D_4 = \begin{bmatrix} 15 & 7 & 13 & 5 \\ 3 & 11 & 1 & 9 \\ 12 & 4 & 14 & 6 \\ 0 & 8 & 2 & 10 \end{bmatrix} \tag{14-35}$$

Another method for mapping a picture with m by n points to a display area with m by n pixels 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. We do this by first scanning across the rows of **M**, 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 value stored in matrix **M** and the displayed intensity level at each pixel position is distributed to neighboring elements in **M**, using the following simplified algorithm:

```
for (i=0; i<m; i++)
        for (j=0; j<n; j++) {
                /* 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};
                M_{i,j+1} := M_{i,j+1} + α · err;
                M_{i+1,j−1} := M_{i+1,j−1} + β · err;
                M_{i+1,j} := M_{i+1,j} + γ · err;
                M_{i+1,j+1} := M_{i+1,j+1} + δ · err;
        }
```

Once the elements of matrix **I** have been assigned intensity-level values, we then map the matrix to some area of a display device, such as a printer or video monitor. Of course, we cannot disperse the error past the last matrix column ($j = n$) or below the last matrix row ($i = m$). For a bilevel system, the available intensity levels are 0 and 1. Parameters for distributing the error can be chosen to satisfy the following relationship

$$\alpha + \beta + \gamma + \delta \leq 1 \tag{14-36}$$

One choice for the error-diffusion parameters that produces fairly good results is $(\alpha, \beta, \gamma, \delta) = (7/16, 3/16, 5/16, 1/16)$. Figure 14-42 illustrates the error distribution using these parameter values. Error diffusion sometimes produces "ghosts" in a picture by repeating, or echoing, certain parts of the picture, particularly with facial features such as hairlines and nose outlines. Ghosting can be re-

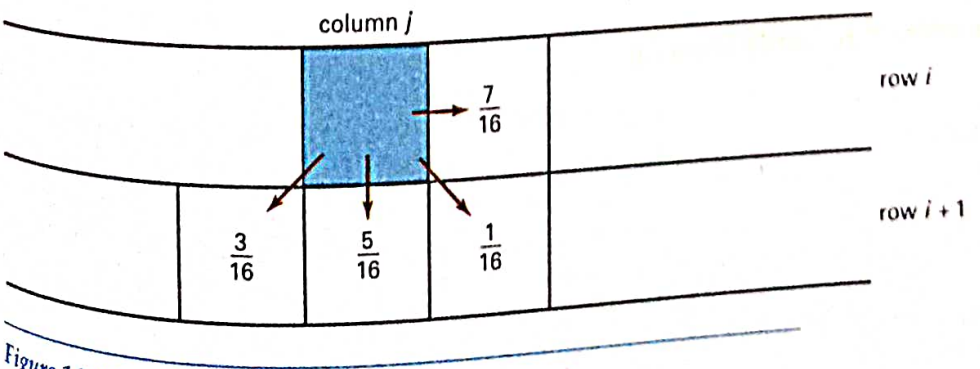


Figure 14-42
Fraction of intensity error that can be distributed to neighboring pixel positions using an error-diffusion scheme.

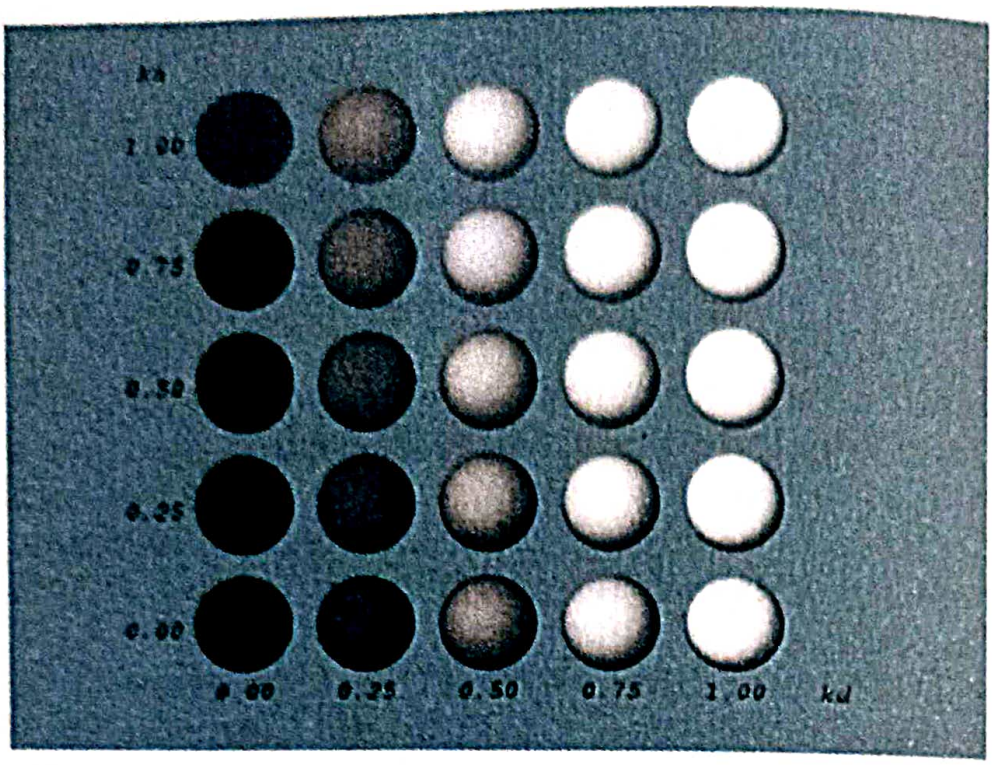


Figure 14-11
Diffuse reflections from a spherical surface illuminated with ambient light and a single point source for values of k_a and k_d in the interval (0, 1).

where both k_a and k_d depend on surface material properties and are assigned values in the range from 0 to 1. Figure 14-11 shows a sphere displayed with surface intensitities calculated from Eq. 14-4 for values of parameters k_a and k_d between 0 and 1.

Specular Reflection and the Phong Model

When we look at an illuminated shiny surface, such as polished metal, an apple, or a person's forehead, we see a highlight, or bright spot, at certain viewing di-

rections. 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 14-12 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**. In this figure, we use **R** to represent the unit vector in the direction of ideal 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 $\cos^{n_s}\phi$. 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 n_s (say, 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. Figures 14-13 and 14-14 show the effect of n_s on the angular range for which we can expect to see specular reflections.

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**, $W(\theta)$, for each surface. Figure 14-15 shows the general variation of $W(\theta)$ over the range $\theta = 0°$ to $\theta = 90°$ for a few materials. In general, $W(\theta)$ tends to increase as the angle of incidence increases. At $\theta = 90°$, $W(\theta) = 1$ 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 $W(\theta)$, we can write the Phong specular-reflection model as

$$I_{spec} = W(\theta) I_l \cos^{n_s}\phi \qquad (14\text{-}5)$$

where I_l is the intensity of the light source, and ϕ is the viewing angle relative to the specular-reflection direction **R**.

Figure 14-13
Modeling specular reflections (shaded area) with parameter n_s.

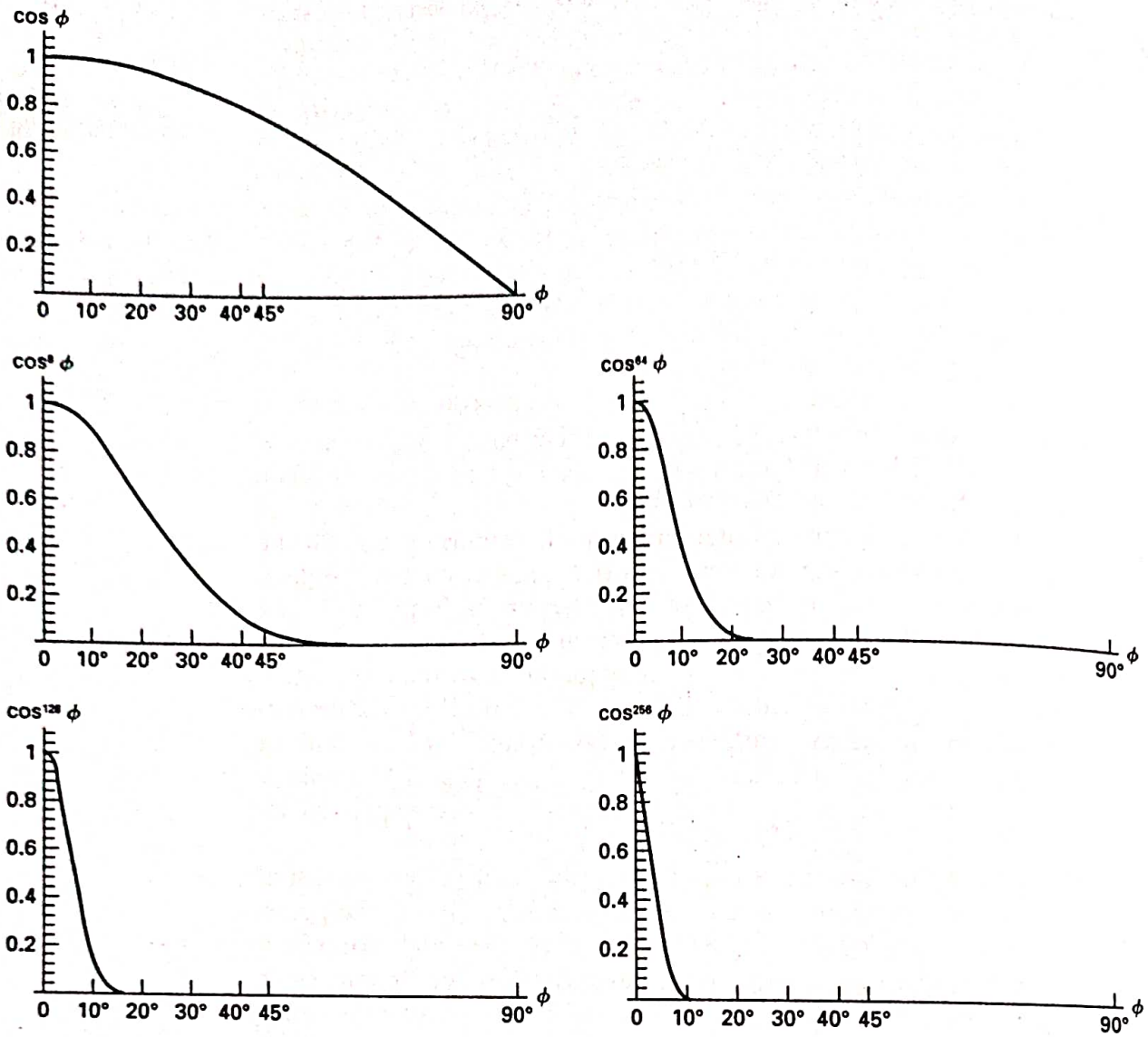


Figure 14-14
Plots of $\cos^{n_s}\phi$ for several values of specular parameter n_s.

As seen in Fig. 14-15, transparent materials, such as glass, only exhibit appreciable specular reflections as θ approaches 90°. At $\theta = 0°$, about 4 percent of the incident light on a glass surface is reflected. And for most of the range of θ, the reflected intensity is less than 10 percent of the incident intensity. But for many opaque materials, specular reflection is nearly constant for all incidence angles. In this case, we can reasonably model the reflected light effects by replacing $W(\theta)$ with a constant specular-reflection coefficient k_s. We then simply set k_s equal to some value in the range 0 to 1 for each surface.

Since $\mathbf{V}$ and $\mathbf{R}$ are unit vectors in the viewing and specular-reflection directions, we can calculate the value of $\cos\phi$ with the dot product $\mathbf{V} \cdot \mathbf{R}$. Assuming the specular-reflection coefficient is a constant, we can determine the intensity of the specular reflection at a surface point with the calculation

$$I_{spec} = k_s I_l (V \cdot \mathbf{R})^{n_s} \tag{14-6}$$

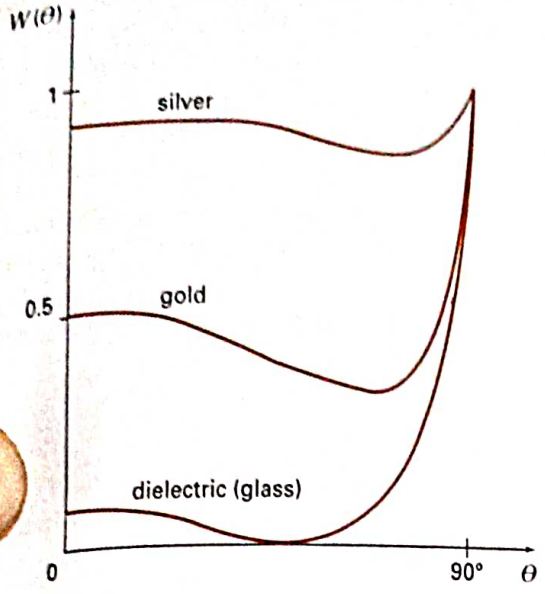


Figure 14-15
Approximate variation of the specular-reflection coefficient as a function of angle of incidence for different materials.

Vector **R** in this expression can be calculated in terms of vectors **L** and **N**. As seen in Fig. 14-16, the projection of **L** onto the direction of the normal vector is obtained with the dot product $\mathbf{N} \cdot \mathbf{L}$. Therefore, from the diagram, we have

$$\mathbf{R} + \mathbf{L} = (2\mathbf{N} \cdot \mathbf{L})\mathbf{N}$$

and the specular-reflection vector is obtained as

$$\mathbf{R} = (2\mathbf{N} \cdot \mathbf{L})\mathbf{N} - \mathbf{L} \tag{14-7}$$

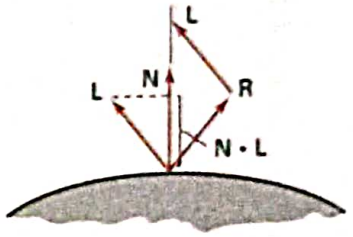


Figure 14-16
Calculation of vector **R** by considering projections onto the direction of the normal vector **N**.

Figure 14-17 illustrates specular reflections for various values of k_s and n_s on a sphere illuminated with a single point light source.

A somewhat simplified Phong model is obtained by using the *halfway vector* **H** between **L** and **V** to calculate the range of specular reflections. If we replace $\mathbf{V} \cdot \mathbf{R}$ in the Phong model with the dot product $\mathbf{N} \cdot \mathbf{H}$, this simply replaces the empirical $\cos\phi$ calculation with the empirical $\cos\alpha$ calculation (Fig. 14-18). The halfway vector is obtained as

$$\mathbf{H} = \frac{\mathbf{L} + \mathbf{V}}{|\mathbf{L} + \mathbf{V}|} \tag{14-8}$$

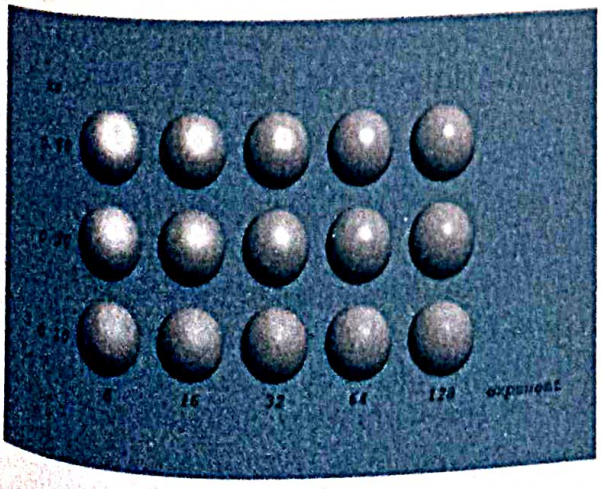


Figure 14-17
Specular reflections from a spherical surface for varying specular parameter values and a single light source.

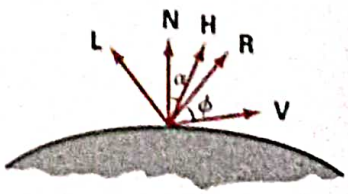


Figure 14-18
Halfway vector **H** along the bisector of the angle between **L** and **V**.

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

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

Combined Diffuse and Specular Reflections with Multiple Light Sources

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

$$\begin{aligned} I &= I_{\text{diff}} + I_{\text{spec}} \\ &= k_a I_a + k_d I_l (\mathbf{N} \cdot \mathbf{L}) + k_s I_l (\mathbf{N} \cdot \mathbf{H})^{n_s} \end{aligned} \tag{14-9}$$

Figure 14-19 illustrates surface lighting effects produced by the various terms in Eq. 14-9. 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:

$$I = k_a I_a + \sum_{i=1}^{n} I_{li} [k_d (\mathbf{N} \cdot \mathbf{L}_i) + k_s (\mathbf{N} \cdot \mathbf{H}_i)^{n_s}] \tag{14-10}$$

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. *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

culations, most packages use empirical models based on simplified photometric calculations. More accurate 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, we first take a look at the basic illumination models often used in graphics packages; then we discuss more accurate, but more time-consuming, methods for calculating surface intensities. And we explore the various surface-rendering algorithms for applying the lighting models to obtain the appropriate shading over visible surfaces in a scene.

14-1 LIGHT SOURCES

When we view an opaque nonluminous object, we see reflected light from the surfaces of the object. The total reflected light is the sum of the contributions from **light sources** and other reflecting surfaces in the scene (Fig. 14-1). Thus, a surface that is not directly exposed to a light source may still be visible if nearby objects are illuminated. Sometimes, light sources are referred to as *light-emitting sources*; and reflecting surfaces, such as the walls of a room, are termed *light-reflecting sources*. We will use the term *light source* to mean an object that is emitting radiant energy, such as a light bulb or the sun.

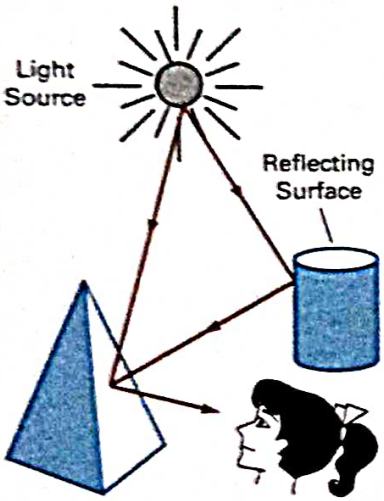


Figure 14-1
Light viewed from an opaque nonluminous surface is in general a combination of reflected light from a light source and reflections of light reflections from other surfaces.

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. 14-2. 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 Fig. 14-3, is more accurately modeled as a **distributed light source**. In this case, the illumination effects cannot 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 part is 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

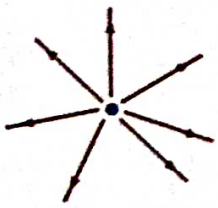


Figure 14-2
Diverging ray paths from a point light source.

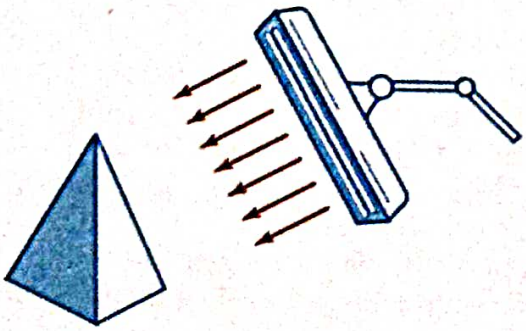


Figure 14-3
An object illuminated with a distributed light source.

surface, some of the incident light will be reflected and some will be transmitted through the material.

Surfaces that are rough, or grainy, tend to scatter the reflected light in all directions. This scattered light is called **diffuse reflection**. A very rough matte surface produces primarily diffuse reflections, so that the surface appears equally bright from all viewing directions. Figure 14-4 illustrates diffuse light scattering from a surface. What we call the color of an object is the color of the diffuse reflection of the incident light. A blue object illuminated by a white light source, for example, reflects the blue component of the white light and totally absorbs all other components. If the blue object is viewed under a red light, it appears black since all of the incident light is absorbed.

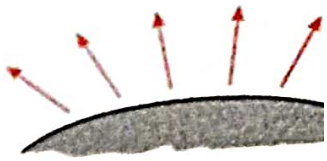


Figure 14-4
Diffuse reflections from surface.

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. An illustration of specular reflection is shown in Fig. 14-5.

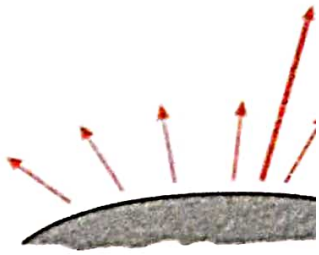


Figure 14-5
Specular reflection superimposed on diffu reflection vectors.

14-2 BASIC ILLUMINATION MODELS

Here we discuss 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 set 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 be 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 is a constant for each surface, 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 an 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 re-

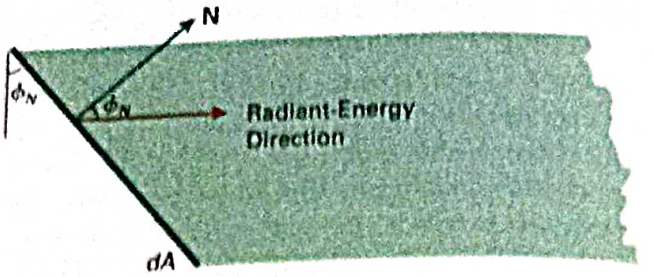


Figure 14-6
Radiant energy from a surface area dA in direction ϕ_N relative to the surface normal direction.

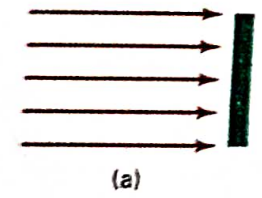
flected can be set for each surface with parameter k_d, the **diffuse-reflection coefficient**, or **diffuse reflectivity**. Parameter k_d is assigned a constant value in the interval 0 to 1, according to the reflecting properties we want the surface to have. If we want a highly reflective surface, we 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 time being we will assume k_d is a constant.

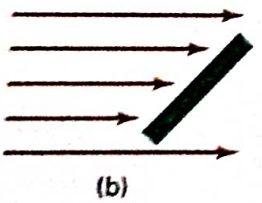
If a surface is exposed only to ambient light, we can express the intensity of the diffuse reflection at any point on the surface as

$$I_{\text{ambdiff}} = k_d I_a \tag{14-1}$$

Since ambient light produces a flat uninteresting shading for each surface (Fig. 14-19(b)), 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's 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 $\cos\phi_N$ (Fig. 14-6). The light intensity, though, depends on the radiant energy per projected area perpendicular to direction ϕ_N, which is $dA\cos\phi_N$. Thus, for Lambertian reflection, the intensity of light is the same over all viewing directions. We discuss photometry concepts and terms, such as radiant energy, in greater detail in Section 14-7.





e 14-7
face perpendicular to
rection of the incident
a) is more illuminated
n equal-sized surface at
que angle (b) to the
ng light direction.

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 at 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 angle between the surface normal and the incoming light direction increases, less of the incident light falls on the surface, as shown in Fig. 14-7. This figure shows a beam of light rays incident on two equal-area plane surface patches with different spatial orientations relative to the incident light direction from a distant source (par-

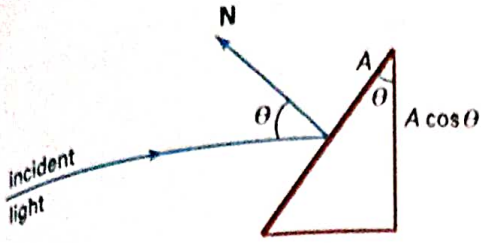


Figure 14-8
An illuminated area projected perpendicular to the path of the incoming light rays.

allel incoming rays). If we denote the **angle of incidence** between the incoming light direction and the surface normal as θ (Fig. 14-8), then the projected area of a surface patch perpendicular to the light direction is proportional to $\cos\theta$. Thus, the amount of illumination (or the "number of incident light rays" cutting across the projected surface patch) depends on $\cos\theta$. If the incoming light from the source is perpendicular to the surface at a particular point, that point is fully illuminated. As the angle of illumination moves away from the surface normal, the brightness of the point drops off. If I_l is the intensity of the point light source, then the diffuse reflection equation for a point on the surface can be written as

$$I_{l,\text{diff}} = k_d I_l \cos\theta \tag{14-2}$$

A surface is illuminated by a point source only if the angle of incidence is in the range 0° to 90° ($\cos\theta$ is in the interval from 0 to 1). When $\cos\theta$ is negative, the light source is "behind" the surface.

If **N** is the unit normal vector to a surface and **L** is the unit direction vector to the point light source from a position on the surface (Fig. 14-9), then $\cos\theta = \mathbf{N}\cdot\mathbf{L}$ and the diffuse reflection equation for single point-source illumination is

$$I_{l,\text{diff}} = k_d I_l(\mathbf{N}\cdot\mathbf{L}) \tag{14-3}$$

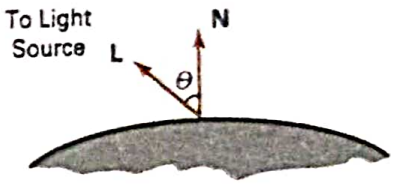


Figure 14-9
Angle of incidence θ between the unit light-source direction vector **L** and the unit surface normal **N**.

Reflections for point-source illumination are calculated in world coordinates or viewing coordinates before shearing and perspective transformations are applied. These transformations may transform the orientation of normal vectors so that they are no longer perpendicular to the surfaces they represent. Transformation procedures for maintaining the proper orientation of surface normals are discussed in Chapter 11.

Figure 14-10 illustrates the application of Eq. 14-3 to positions over the surface of a sphere, using various values of parameter k_d between 0 and 1. Each projected pixel position for the surface was assigned an intensity as calculated by the diffuse reflection equation for a point light source. The renderings in this figure illustrate single point-source lighting with no other lighting effects. This is what we might expect to see if we shined a small light on the object in a completely darkened room. For general scenes, however, we expect some background lighting effects in addition to the illumination effects produced by a direct light source.

We can combine the ambient and point-source intensity calculations to obtain an expression for the total diffuse reflection. In addition, many graphics packages introduce an **ambient-reflection coefficient** k_a to modify the ambient-light intensity I_a for each surface. This simply provides us with an additional parameter to adjust the light conditions in a scene. Using parameter k_a, we can write the total diffuse reflection equation as

$$I_{\text{diff}} = k_a I_a + k_d I_l(\mathbf{N}\cdot\mathbf{L}) \tag{14-4}$$

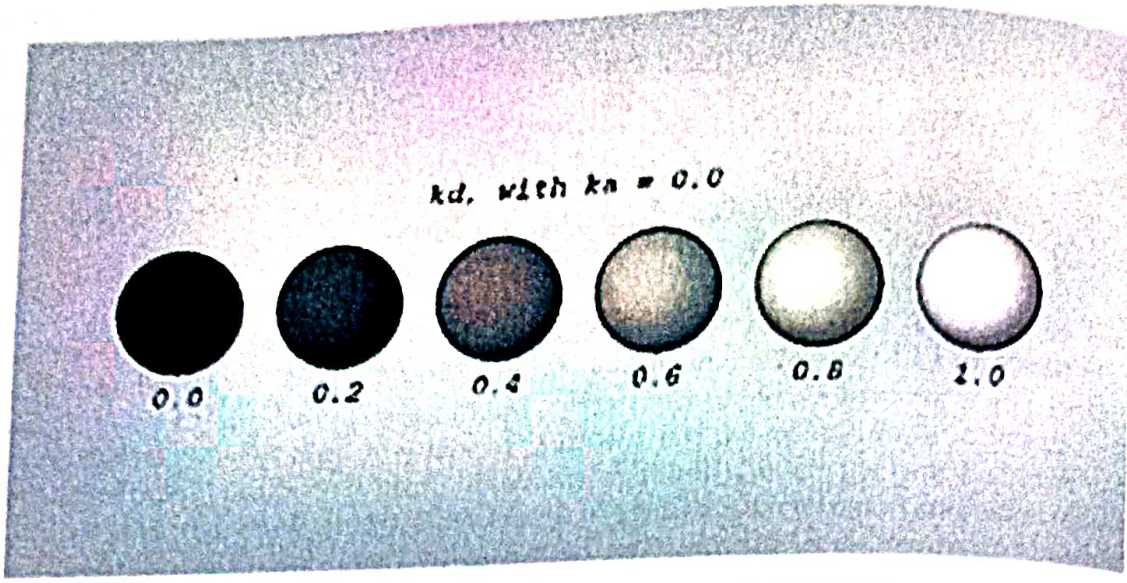


Figure 14-10
Diffuse reflections from a spherical surface illuminated by a point light source for values of the diffuse reflectivity coefficient in the interval $0 \leq k_d \leq 1$.

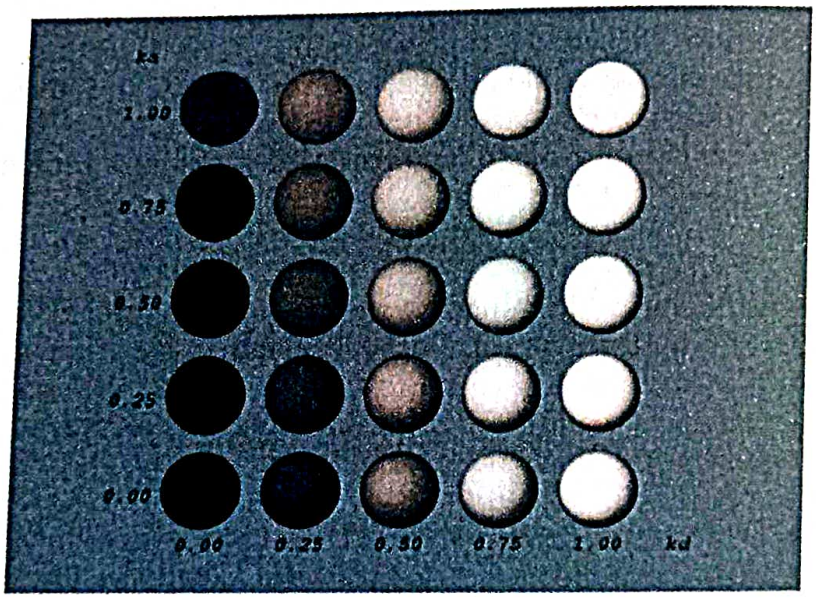


Figure 14-11
Diffuse reflections from a spherical surface illuminated with ambient light and a single point source for values of k_a and k_d in the interval (0, 1).

where both k_a and k_d depend on surface material properties and are assigned values in the range from 0 to 1. Figure 14-11 shows a sphere displayed with surface intensitities calculated from Eq. 14-4 for values of parameters k_a and k_d between 0 and 1.

Specular Reflection and the Phong Model

When we look at an illuminated shiny surface, such as polished metal, an apple, or a person's forehead, we see a highlight, or bright spot, at certain viewing di-

Figure 14-80
A room scene rendered with distributed ray-tracing methods. (*Courtesy of John Snyder, Jed Lengyel, Devendra Kalra, and Al Barr, Computer Graphics Lab, California Institute of Technology. Copyright © 1988 Caltech.*)

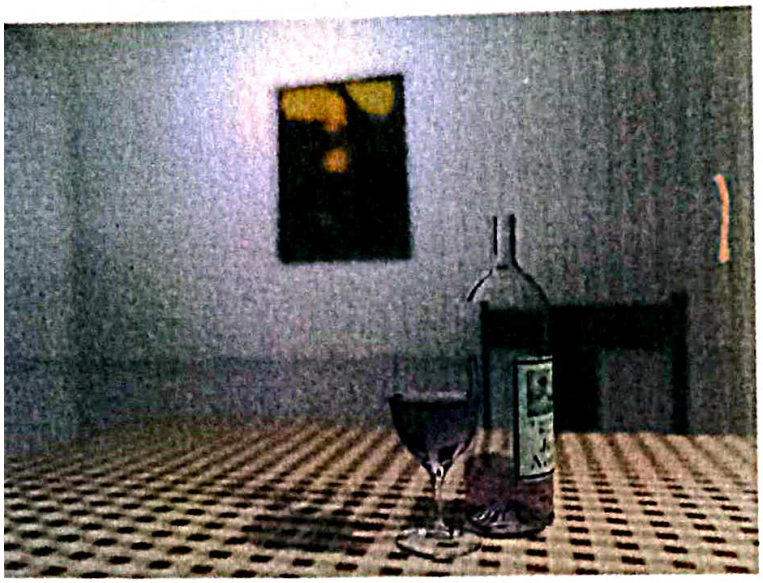


Figure 14-81
A scene showing the focusing, antialiasing, and illumination effects possible with a combination of ray-tracing and radiosity methods. Realistic physical models of light illumination were used to generate the refraction effects, including the caustic in the shadow of the glass. (*Courtesy of Peter Shirley, Department of Computer Science, Indiana University.*)

14-7 RADIOSITY LIGHTING MODEL

We can accurately model diffuse reflections 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

In this method, we need to consider the radiant-energy interactions between all surfaces in a scene. We do this by determining the differential amount of radiant energy dB leaving each surface point in the scene and summing the energy contributions over all surfaces to obtain the amount of energy transfer between surfaces. With reference to Fig. 14-82, dB is the visible radiant energy emanating from the surface point in the direction given by angles θ and ϕ within differential solid angle $d\omega$ per unit time per unit surface area. Thus, dB has units of *joules/(second · meter²)*, or *watts/meter²*.

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)*:

$$I = \frac{dB}{d\omega \cos \phi} \tag{14-68}$$

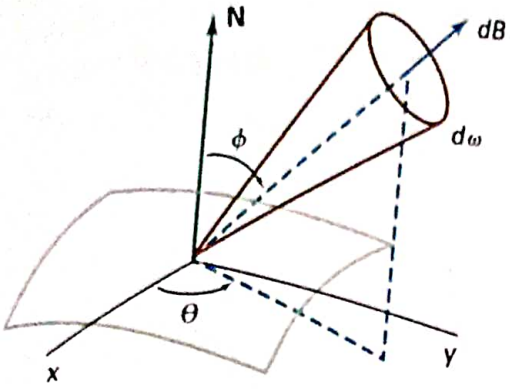


Figure 14-82
Visible radiant energy emitted from a surface point in direction (θ, ϕ) within solid angle $d\omega$.

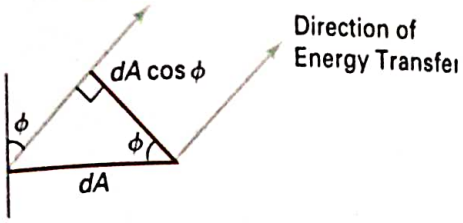


Figure 14-83
For a unit surface element, the projected area perpendicular to the direction of energy transfer is equal to $\cos\phi$.

Assuming the surface is an ideal diffuse reflector, we can set intensity I to a constant for all viewing directions. Thus, $dB/d\omega$ is proportional to the projected surface area (Fig. 14-83). To obtain the total rate of energy radiation from the surface point, we need to sum the radiation for all directions. That is, we want the total energy emanating from a hemisphere centered on the surface point, as in Fig. 14-84:

$$B = \int_{\text{hemi}} dB \tag{14-69}$$

For a perfect diffuse reflector, I is a constant, so we can express radiant energy B as

$$B = I \int_{\text{hemi}} \cos\phi \, d\omega \tag{14-70}$$

Also, the differential element of solid angle $d\omega$ can be expressed as (Appendix A)

$$d\omega = \frac{dS}{r^2} = \sin\phi \, d\phi \, d\theta$$

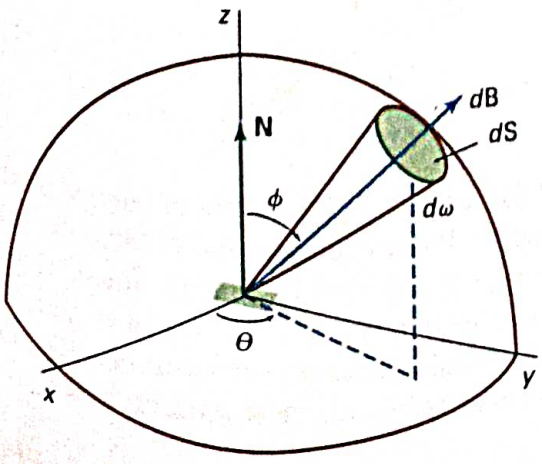


Figure 14-84
Total radiant energy from a surface point is the sum of the contributions in all directions over a hemisphere centered on the surface point.

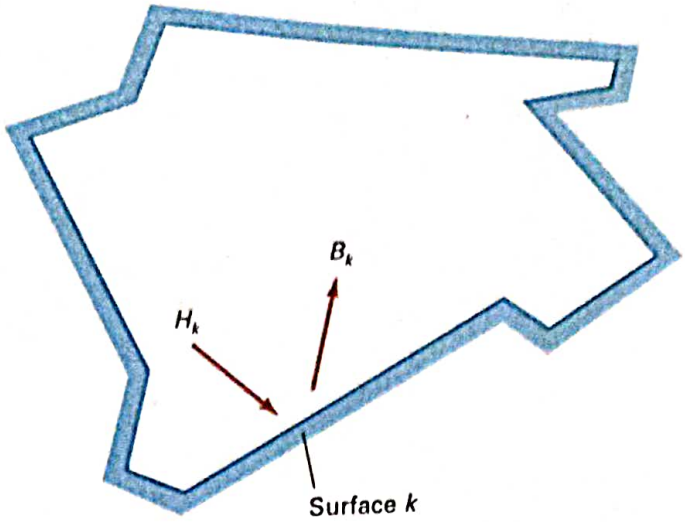


Figure 14-85
An enclosure of surfaces for the radiosity model.

so that

$$B = I\int_0^{2\pi}\int_0^{\pi/2} \cos\phi \sin\phi \, d\phi \, d\theta \qquad (14\text{-}71)$$
$$= I\pi$$

A model for the light reflections from the various surfaces is formed by setting up an "enclosure" of surfaces (Fig. 14-85). Each surface in the enclosure is either a reflector, an emitter (light source), or a combination reflector-emitter. We designate radiosity parameter B_k as the total rate of energy leaving surface k per unit area. Incident-energy parameter H_k is the sum of the energy contributions from all surfaces in the enclosure arriving at surface k per unit time per unit area. That is,

$$H_k = \sum_j B_j F_{jk} \qquad (14\text{-}72)$$

where parameter F_{jk} is the *form factor* for surfaces j and k. Form factor F_{jk} is the fractional amount of radiant energy from surface j that reaches surface k.

For a scene with n surfaces in the enclosure, the radiant energy from surface k is described with the **radiosity equation**:

$$B_k = E_k + \rho_k H_k$$
$$= E_k + \rho_k \sum_{j=1}^{n} B_j F_{jk} \qquad (14\text{-}73)$$

If surface k is not a light source, $E_k = 0$. Otherwise, E_k is the rate of energy emitted from surface k per unit area (*watts/meter*^{2}). Parameter ρ_k is the reflectivity factor for surface k (percent of incident light that is reflected in all directions). This reflectivity factor is related to the diffuse reflection coefficient used in empirical illumination models. Plane and convex surfaces cannot "see" themselves, so that no self-incidence takes place and the form factor F_{kk} for these surfaces is 0.

To obtain the illumination effects over the various surfaces in the enclosure, we need to solve the simultaneous radiosity equations for the n surfaces given the array values for E_k, ρ_k, and F_{jk}. That is, we must solve

$$(1 - \rho_k F_{kk})B_k - \rho_k \sum_{j \neq k} B_j F_{jk} = E_k, \qquad k = 1, 2, 3, \ldots, n \tag{14-74}$$

or

$$\begin{bmatrix} 1-\rho_1 F_{11} & -\rho_1 F_{12} & \cdots & -\rho_1 F_{1n} \\ -\rho_2 F_{21} & 1-\rho_2 F_{22} & \cdots & -\rho_2 F_{2n} \\ \vdots & \vdots & & \vdots \\ -\rho_n F_{n1} & -\rho_2 F_{n2} & \cdots & 1-\rho_n F_{nn} \end{bmatrix} \cdot \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_n \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_n \end{bmatrix} \tag{14-75}$$

We then convert to intensity values I_k by dividing the radiosity values B_k by π. For color scenes, we can calculate the individual RGB components of the radiosity (B_{kR}, B_{kG}, B_{kB}) from the color components of ρ_k and E_k.

Before we can solve Eq. 14-74, we need to determine the values for form factors F_{jk}. We do this by considering the energy transfer from surface j to surface k (Fig. 14-86). The rate of radiant energy falling on a small surface element dA_k from area element dA_j is

$$dB_j\, dA_j = (I_j \cos\phi_j\, d\omega) dA_j \tag{14-76}$$

But solid angle $d\omega$ can be written in terms of the projection of area element dA_k perpendicular to the direction dB_j:

$$d\omega = \frac{dA}{r^2} = \frac{\cos\phi_k dA_k}{r^2} \tag{14-77}$$

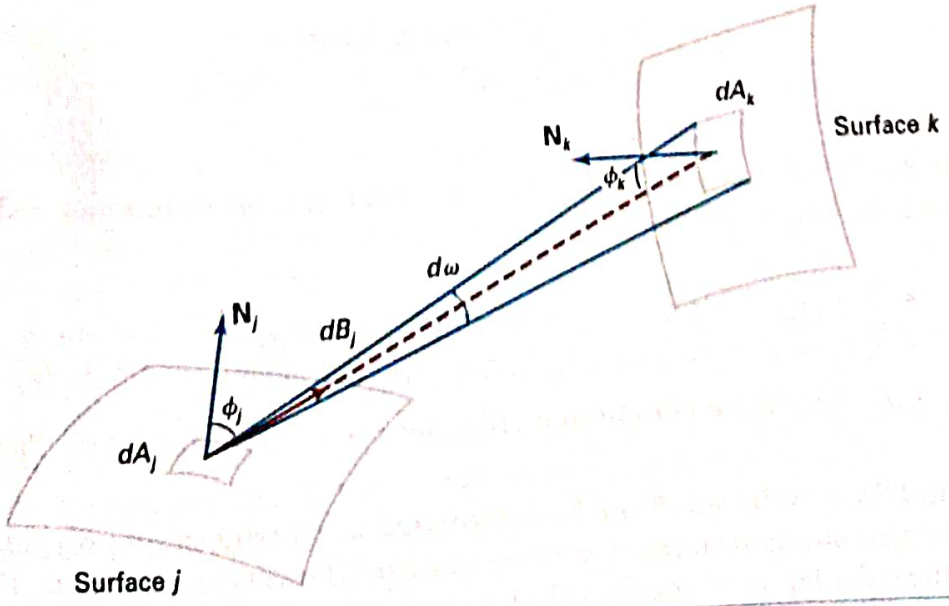


Figure 14-86
Rate of energy transfer dB_j from a surface element with area dA_j to surface element dA_k.

so we can express Eq. 14-76 as

$$dB_j\, dA_j = \frac{I_j \cos\phi_j \cos\phi_k\, dA_j\, dA_k}{r^2} \tag{14-78}$$

The form factor between the two surfaces is the percent of energy emanating from area dA_j that is incident on dA_k:

$$F_{dA_j, dA_k} = \frac{\text{energy incident on } dA_k}{\text{total energy leaving } dA_j} \tag{14-79}$$

$$= \frac{I_j \cos\phi_j \cos\phi_k\, dA_j\, dA_k}{r^2} \cdot \frac{1}{B_j\, dA_j}$$

Also $B_j = \pi I_j$, so that

$$F_{dA_j, dA_k} = \frac{\cos\phi_j \cos\phi_k\, dA_k}{\pi r^2} \tag{14-80}$$

The fraction of emitted energy from area dA_j incident on the entire surface k is then

$$F_{dA_j, A_k} = \int_{\text{surf}_j} \frac{\cos\phi_j \cos\phi_k}{\pi r^2}\, dA_k \tag{14-81}$$

where A_k is the area of surface k. We now can define the form factor between the two surfaces as the area average of the previous expression:

$$F_{jk} = \frac{1}{A_j} \int_{\text{surf}_j} \int_{\text{surf}_k} \frac{\cos\phi_j \cos\phi_k}{\pi r^2}\, dA_k\, dA_j \tag{14-82}$$

Integrals 14-82 are evaluated using numerical integration techniques and stipulating the following conditions:

- $\sum_{k=1}^{n} F_{jk} = 1$, for all k (conservation of energy)
- $A_j F_{jk} = A_k F_{kj}$ (uniform light reflection)
- $F_{jj} = 0$, for all j (assuming only plane or convex surface patches)

Each surface in the scene can be subdivided into many small polygons, and the smaller the polygon areas, the more realistic the display appears. We can speed up the calculation of the form factors by using a hemicube to approximate the hemisphere. This replaces the spherical surface with a set of linear (plane) surfaces. Once the form factors are evaluated, we can solve the simultaneous lin-

ear equations 14-74 using, say, Gaussian elimination or LU decomposition methods (Appendix A). Alternatively, we can start with approximate values for the B_j and solve the set of linear equations iteratively using the Gauss–Seidel method. At each iteration, we calculate an estimate of the radiosity for surface patch k using the previously obtained radiosity values in the radiosity equation:

$$B_k = E_k + \rho_k \sum_{j=1}^{n} B_j F_{jk} \quad .$$

We can then display the scene at each step, and an improved surface rendering is viewed at each iteration until there is little change in the calculated radiosity values.

Progressive Refinement Radiosity Method

Although the radiosity method produces highly realistic surface rendings, there are tremendous storage requirements, and considerable processing time is needed to calculate the form factors. Using *progressive refinement*, we can restructure the iterative radiosity algorithm to speed up the calculations and reduce storage requirements at each iteration.

From the radiosity equation, the radiosity contribution between two surface patches is calculated as

$$B_k \text{ due to } B_j = \rho_k B_j F_{jk} \tag{14-83}$$

Reciprocally,

for all j (14-84)