CHAPT

7

lations, most packages use empirical models based on simplified photometric lculations. More accurate models, such as the radiosity algorithm, calculate ght intensities by considering the propagation of radiant energy between the urfaces and light sources in a scene. In the following sections, we first take a ook 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.

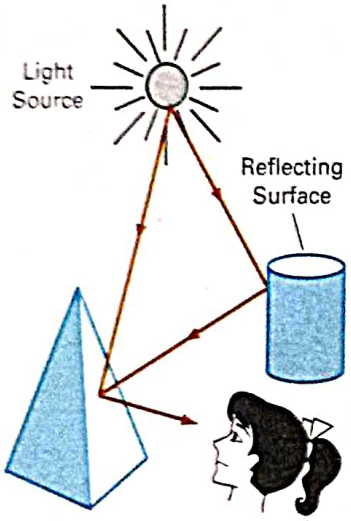


Figure 14-1
Light viewed from an opaque nonluminous surface is in general a combination of reflected light from a light source and reflections of light reflections from other surfaces.

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.

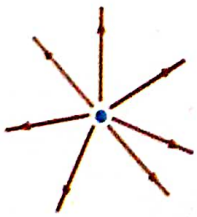


Figure 14-2
Diverging ray paths from a point light source.

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

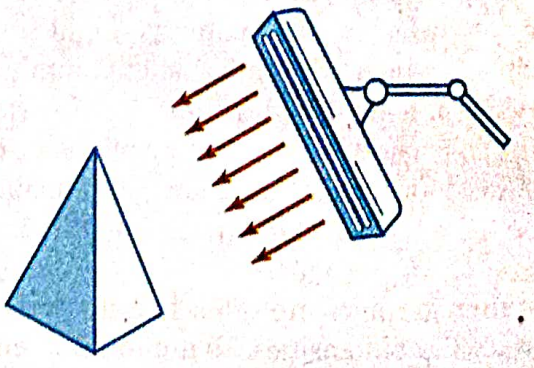


Figure 14-3
An object illuminated with a distributed light source.

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.

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.

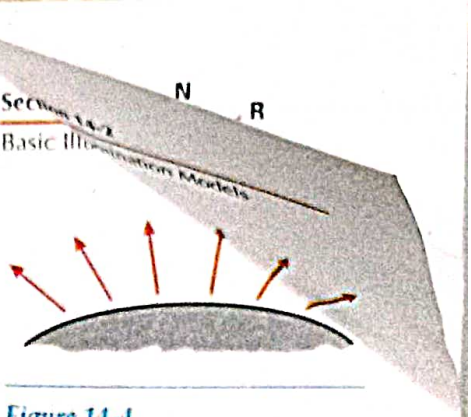


Figure 14-4
Diffuse reflections from a surface.

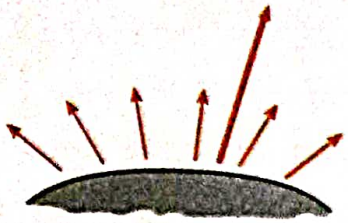


Figure 14-5
Specular reflection superimposed on diffuse reflection vectors.

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-

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 a 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.

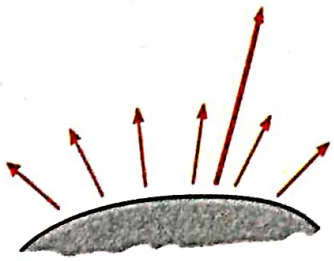


Figure 14-5
Specular reflection superimposed on diffuse reflection vectors.

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-

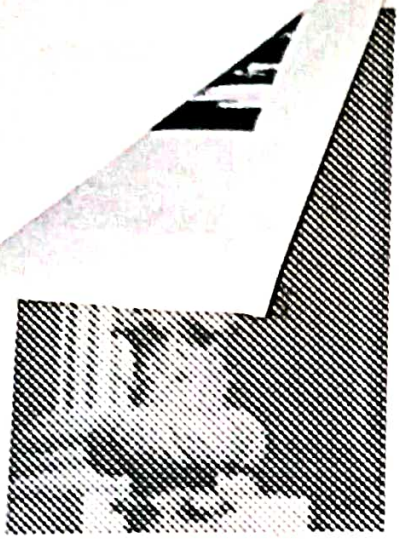


Figure 14-34
An enlarged section of a photograph reproduced with a halftoning method, showing how tones are represented with varying size dots.

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

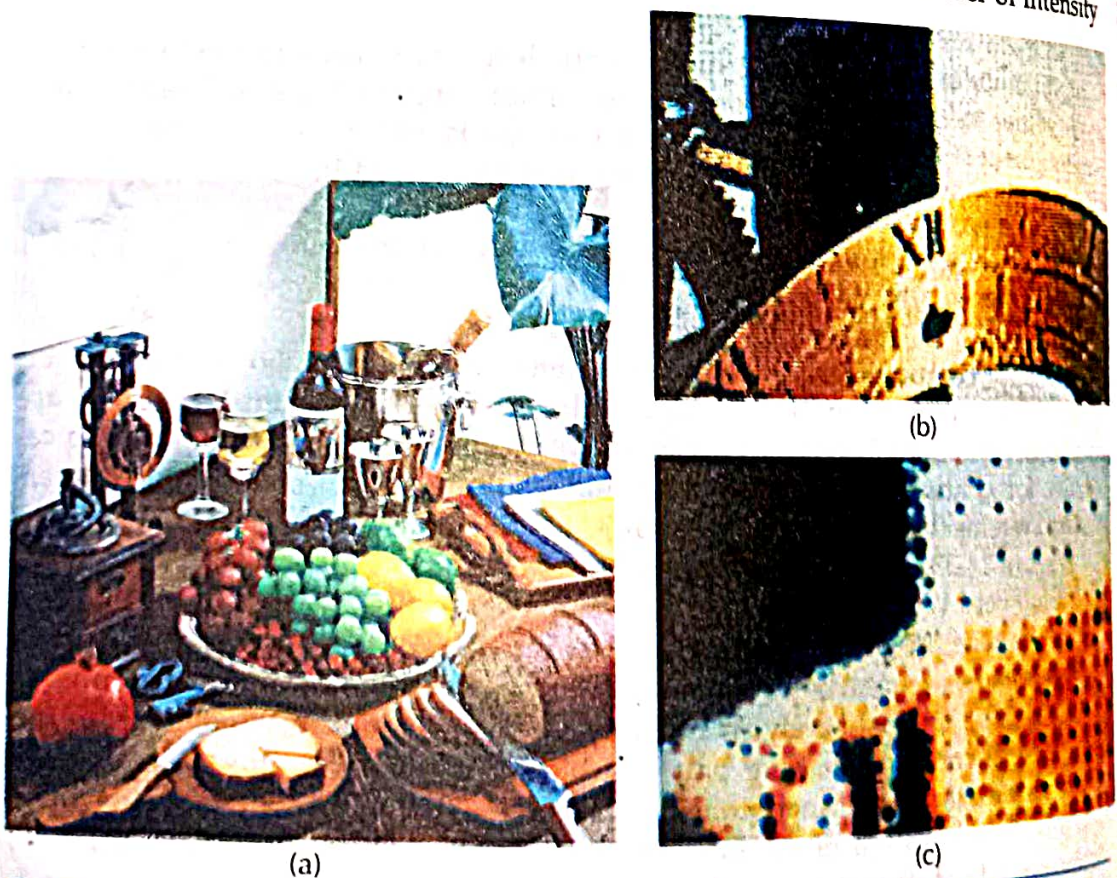


Figure 14-35
Color halftone dot patterns. The top half of the clock in the color halftone (a) is enlarged by a factor of 10 in (b) and by a factor of 50 in (c).

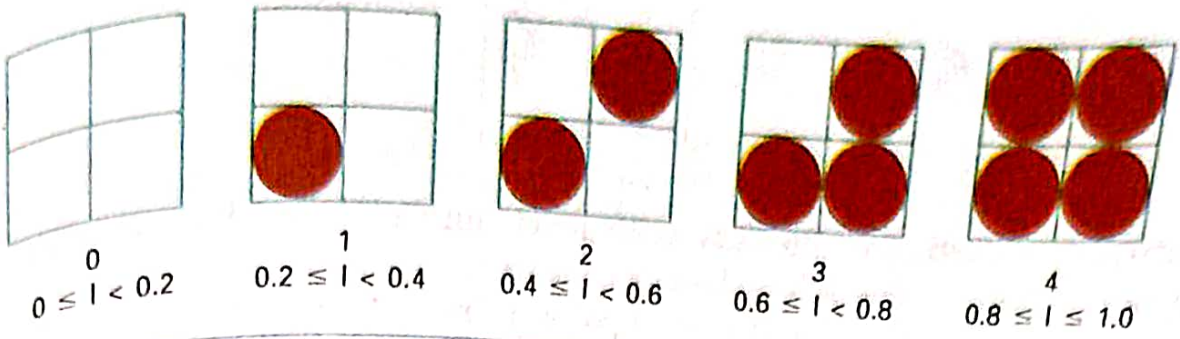


Figure 14-36
A 2 by 2 pixel grid used to display five intensity levels on a bilevel system. The intensity values that would be mapped to each grid are listed below each pixel pattern.

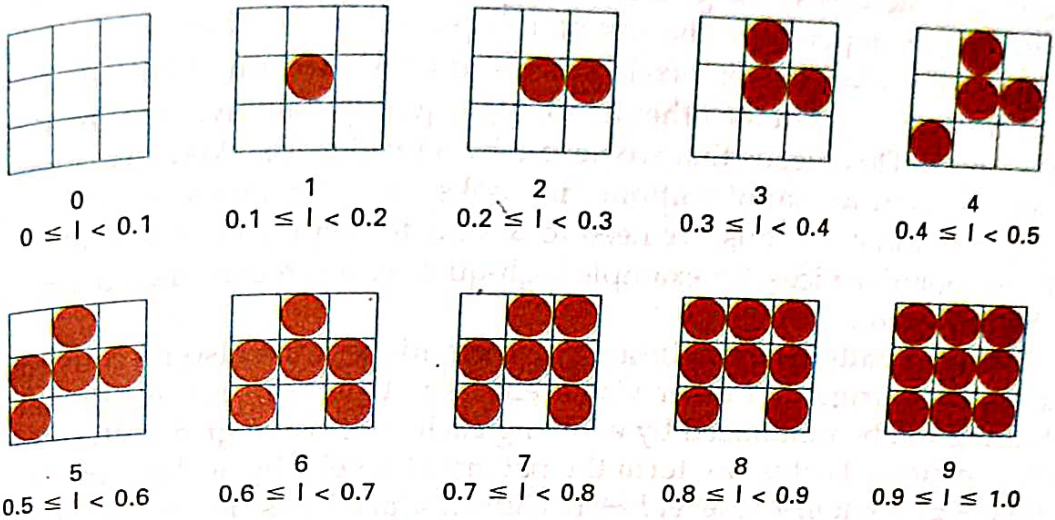


Figure 14-37
A 3 by 3 pixel grid can be used to display 10 intensities on a bilevel system. The intensity values that would be mapped to each grid are listed below each pixel pattern.

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 $0 \leq I < 0.2$, pattern 1 for $0.2 \leq I < 0.4$, and pattern 4 is used for $0.8 \leq I \leq 1.0$.

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.

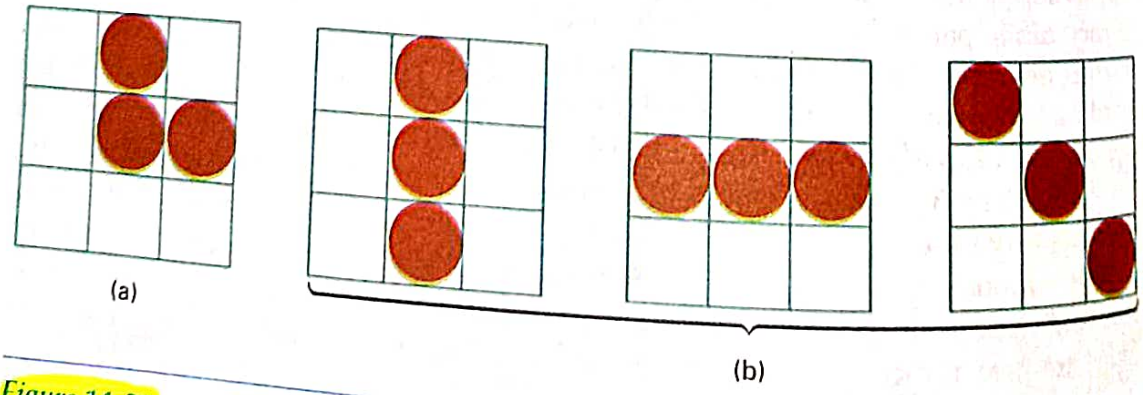


Figure 14-38
For a 3 by 3 pixel grid, pattern (a) is to be preferred to the patterns in (b) for representing the third intensity level above 0.

34	48	40	32	29	15	23	31
42	58	56	53	21	5	7	10
50	62	61	45	13	1	2	18
38	46	54	37	25	17	9	26
28	14	22	30	35	49	41	33
20	4	6	11	43	59	57	52
12	0	3	19	51	63	60	44
24	16	8	27	39	47	55	36

Figure 14-43
One possible distribution scheme for dividing the intensity array into 64 dot-diffusion classes, numbered from 0 through 63.

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

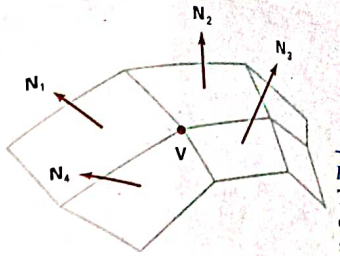
In this section, we consider the application of an illumination model to the rendering of standard graphics objects: those formed with polygon surfaces. The objects are usually polygon-mesh approximations of curved-surface objects, but they may also be polyhedra that are not curved-surface approximations. 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.



- All light sources illuminating the object are
so that **N** · **L** and the attenuation function a
- The viewing position is sufficiently far fro
stant over the surface.

Even if all of these conditions are not true, we
surface-lighting effects using small polygon fac
the intensity for each facet, say, at the center of

Gouraud Shading

This **intensity-interpolation** scheme, develop
ferred to as **Gouraud shading**, renders a poly
ing intensity values across the surface. Inte
matched with the values of adjacent polygo
eliminating the intensity discontinuities that c

Each polygon surface is rendered with G
following calculations:

- Determine the average unit normal vect
- Apply an illumination model to each ve
- Linearly interpolate the vertex intensitie

At each polygon vertex, we obtain a no
normals of all polygons sharing that vertex,
any vertex position **V**, we obtain the unit vert

$$\mathbf{N}_V = \frac{\sum_{k=1}^{n} \mathbf{N}}{\left|\sum_{k=1}^{n} \mathbf{N}\right.}$$

Once we have the vertex normals, we can d
from a lighting model.

Figure 14-45 demonstrates the next st
polygon edges. For each scan line, the inten
with a polygon edge is linearly interpolated
points. For the example in Fig. 14-45, the p
positions 1 and 2 is intersected by the scan
taining the intensity at point 4 is to interpo
only the vertical displacement of the scan li

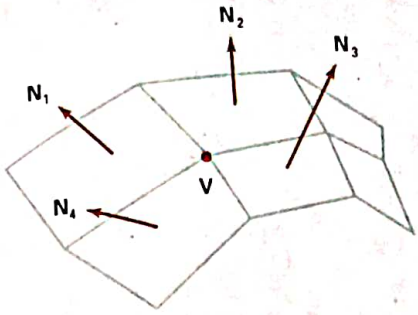


Figure 14-44
The normal vector at vertex **V** is calculated as the average of the surface normals for each polygon sharing that vertex.

- 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:

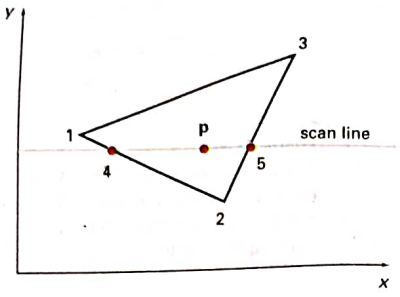


Figure 14-45
For Gouraud shading, the intensity at point 4 is linearly interpolated from the intensities at vertices 1 and 2. The intensity at point 5 is linearly interpolated from intensities at vertices 2 and 3. An interior point **p** is then assigned an intensity value that is linearly interpolated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

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

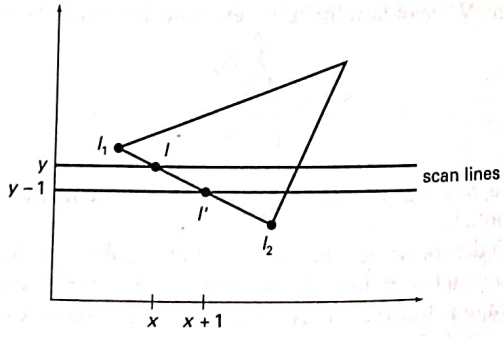


Figure 14-46
Incremental interpolation of intensity values along a polygon edge for successive scan lines.

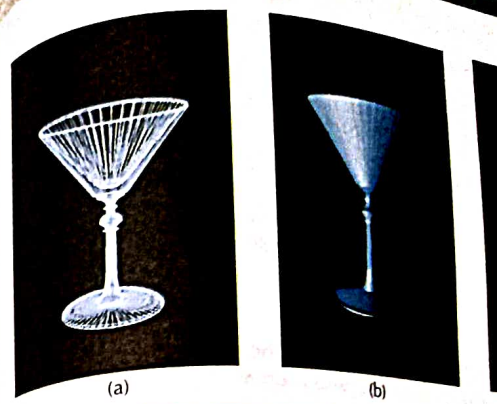


Figure 14-47
A polygon mesh approximation of an object (a) is rendered with shading (b) and with Gouraud shading (c).

Similar calculations are used to obtain intensities at suc positions along each scan line.

When surfaces are to be rendered in color, the inte ponent is calculated at the vertices. Gouraud shading hidden-surface algorithm to fill in the visible polygons example of an object shaded with the Gouraud method a

Gouraud shading removes the intensity discontinu constant-shading model, but it has some other deficie surface are sometimes displayed with anomalous shapes interpolation can cause bright or dark intensity streaks, pear on the surface. These effects can be reduced by di greater number of polygon faces or by using other meth ing, that require more calculations.

Phong Shading

A more accurate method for rendering a polygon surfa vectors, and then apply the illumination model to method, developed by Phong Bui Tuong, is called Ph **vector interpolation shading**. It displays more realisti and greatly reduces the Mach-band effect.

A polygon surface is rendered using Phong shadi lowing steps:

- Determine the average unit normal vector at each
- Linearly interpolate the vertex normals over the s
- Apply an illumination model along each scan pixel intensities for the surface points.

Interpolation of surface normals along a polygon is illustrated in Fig. 14-48. The normal vector **N** fo point along the edge between vertices 1 and 2 can be polating between edge endpoint normals:

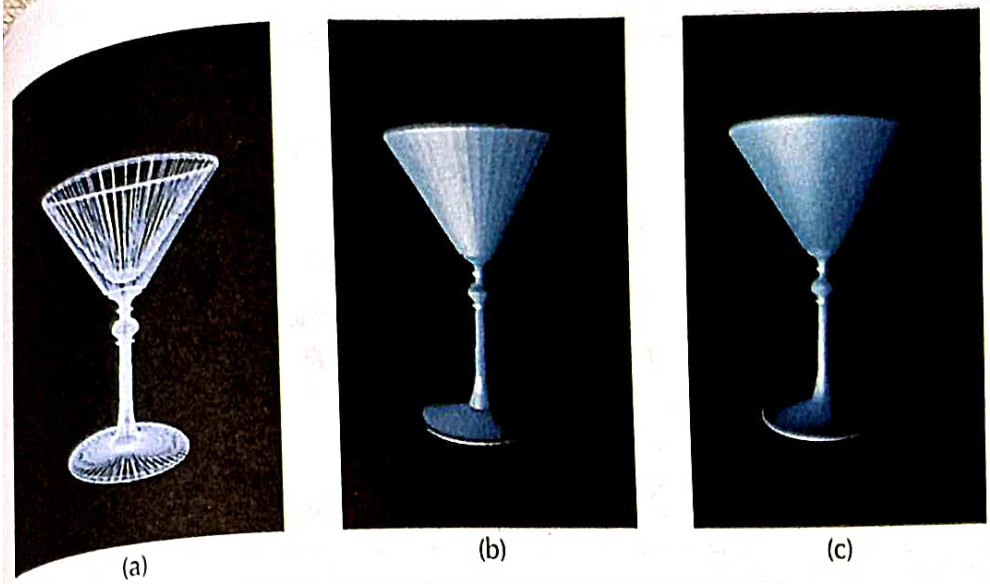


Figure 14-47
A polygon mesh approximation of an object (a) is rendered with flat shading (b) and with Gouraud shading (c).

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:

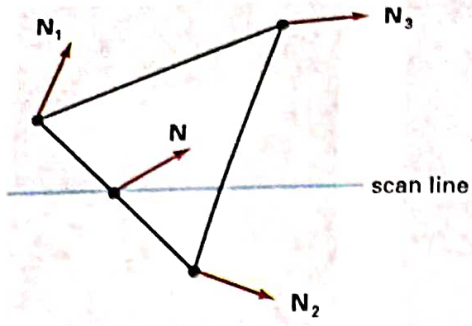


Figure 14-48
Interpolation of surface normals along a polygon edge.

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

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

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

We can rewrite this expression in the form

$$I_{\text{diff}}(x, y) = \frac{ax + by + c}{(dx^2 + exy + fy^2 + gx + hy + i)^{1/2}} \tag{14-46}$$

where parameters such as a, b, c, and d are used to represent the various dot products. For example,

$$a = \frac{\mathbf{L} \cdot \mathbf{A}}{|\mathbf{L}|} \tag{14-47}$$

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

$$I_{\text{diff}}(x, y) = T_5x^2 + T_4xy + T_3y^2 + T_2x + T_1y + T_0 \tag{14-48}$$

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 $(\mathbf{N} \cdot \mathbf{H})^{n_s}$ in the basic illumination model. In addition, we can generalize the algorithm to include polygons other than triangles and finite viewing positions.

14-6 RAY-TRACING METHODS

In Section 10-15, 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 (Section 13-10). **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. An example of the global reflection and transmission effects possible with ray tracing is shown in Fig. 14-50.

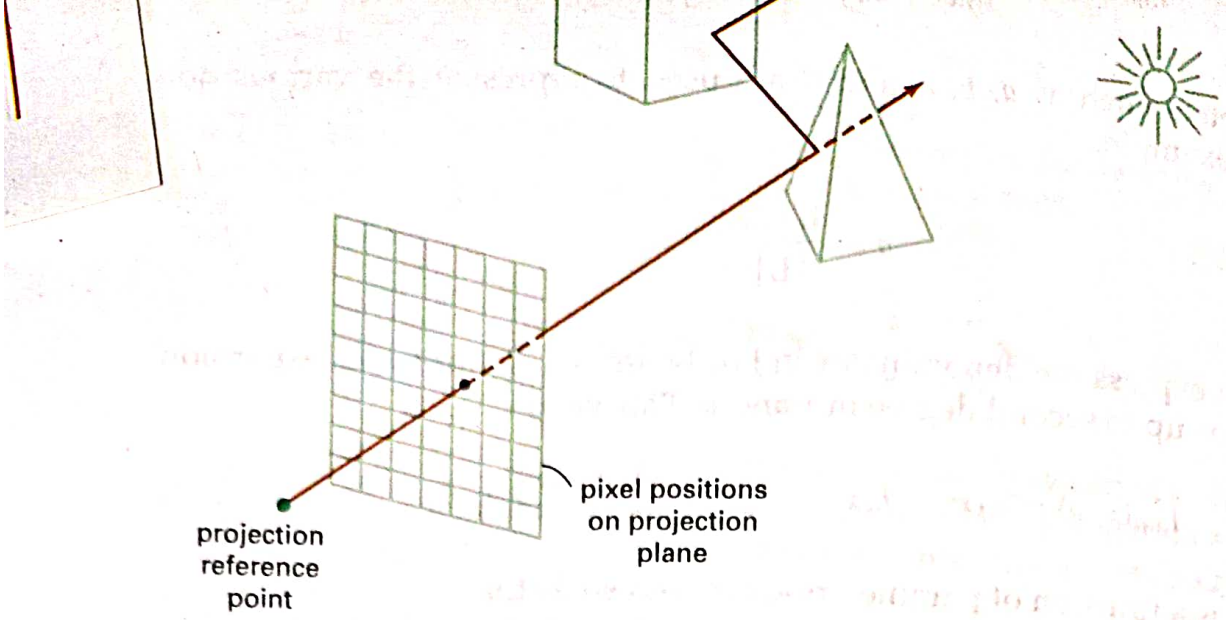


Figure 14-49
Tracing a ray from the projection reference point through a pixel position with multiple reflections and transmissions.

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.

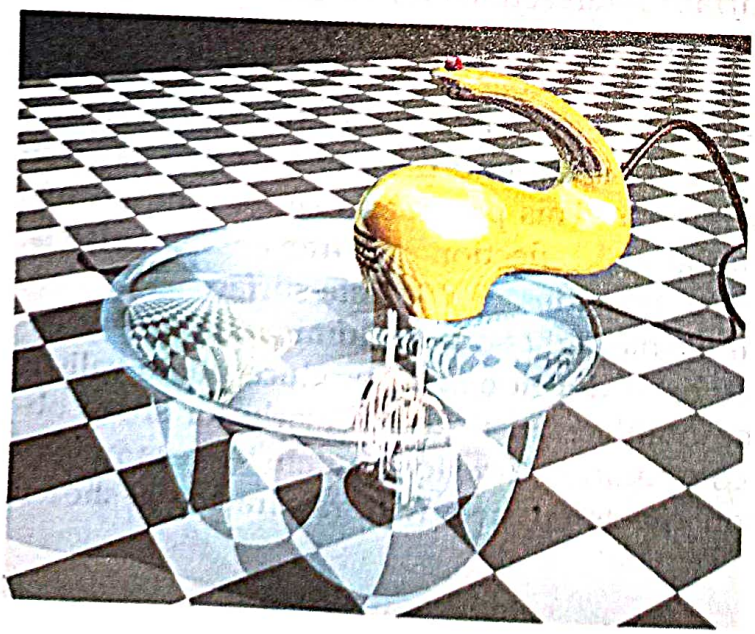


Figure 14-50
A ray-traced scene, showing global reflection and transmission illumination effects from object surfaces. (*Courtesy of Evans & Sutherland.*)

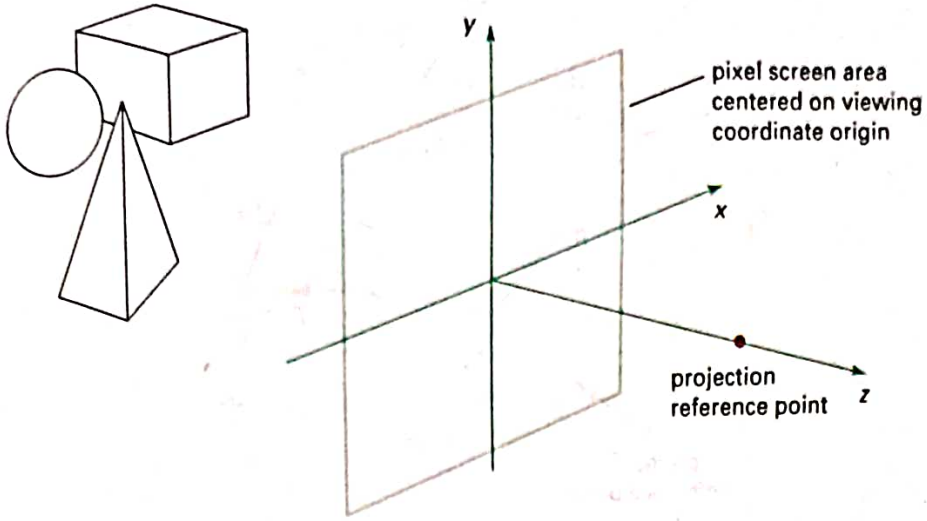


Figure 14-51
Ray-tracing coordinate-reference frame.

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