Representations are needed for nonrigid, flexible, jointed objects. Work on transformations that bend and twist objects is described in Chapter 20. Many objects cannot be specified with total accuracy; rather, their shapes are defined by parameters constrained to lie within a range of values. These are known as "toleranced" objects, and correspond to real objects turned out by machines such as lathes and stampers [REQU84]. New representations are being developed to encode toleranced objects [GOSS88].

Common to all designed objects is the concept of "features," such as holes and chamfers, that are designed for specific purposes. One current area of research is exploring the possibility of recognizing features automatically and inferring the designer's intent for what each feature should accomplish [PRAT84]. This will allow the design to be checked to ensure that the features perform as intended. For example, if certain features are designed to give a part strength under pressure, then their ability to perform this function could be validated automatically. Future operations on the object could also be checked to ensure that the features' functionality was not compromised.

EXERCISES

12.1 Define the results of performing $\cup^*$ and $-^*$ for two polyhedral objects in the same way as the result of performing $\cap^*$ was defined in Section 12.2. Explain how the resulting object is constrained to be a regular set, and specify how the normal is determined for each of the object's faces.

12.2 Consider the task of determining whether or not a legal solid is the null object (which has no volume). How difficult is it to perform this test in each of the representations discussed?

12.3 Consider a system whose objects are represented as sweeps and can be operated on using the regularized Boolean set operators. What restrictions must be placed on the objects to ensure closure?

12.4 Implement the algorithms for performing Boolean set operations on quadtrees or on octrees.

12.5 Explain why an implementation of Boolean set operations on quadtrees or octrees does not need to address the distinction between the ordinary and regularized operations described in Section 12.2.

12.6 Although the geometric implications of applying the regularized Boolean set operators are unambiguous, it is less clear how object properties should be treated. For example, what properties should be assigned to the intersection of two objects made of different materials? In modeling actual objects, this question is of little importance, but in the artificial world of graphics, it is possible to intersect any two materials. What solutions do you think would be useful?

12.7 Explain how a quadtree or octree could be used to speed up 2D or 3D picking in a graphics package.

12.8 Describe how to perform point classification in primitive instancing, b-rep, spatial occupancy enumeration, and CSG.

13 Achromatic and Colored Light

The growth of raster graphics has made color and gray scale an integral part of contemporary computer graphics. Color is an immensely complex subject, one that draws on concepts and results from physics, physiology, psychology, art, and graphic design. Many researchers' careers have been fruitfully devoted to developing theories, measurement techniques, and standards for color. In this chapter, we introduce some of the areas of color that are most relevant to computer graphics.

The color of an object depends not only on the object itself, but also on the light source illuminating it, on the color of the surrounding area, and on the human visual system. Furthermore, some objects reflect light (wall, desk, paper), whereas others also transmit light (cellophane, glass). When a surface that reflects only pure blue light is illuminated with pure red light, it appears black. Similarly, a pure green light viewed through glass that transmits only pure red will also appear black. We postpone some of these issues by starting our discussion with achromatic sensations—that is, those described as black, gray, and white.

13.1 ACHROMATIC LIGHT

Achromatic light is what we see on a black-and-white television set or display monitor. An observer of achromatic light normally experiences none of the sensations we associate with red, blue, yellow, and so on. Quantity of light is the only attribute of achromatic light. Quantity of light can be discussed in the physics sense of energy, in which case the terms *intensity* and *luminance* are used, or in the psychological sense of perceived intensity, in which case the term *brightness* is used. As we shall discuss shortly, these two concepts are related but are not the same. It is useful to associate a scalar with different intensity levels, defining 0 as black and 1 as white; intensity levels between 0 and 1 represent different grays.

A black-and-white television can produce many different intensities at a single pixel position. Line printers, pen plotters, and electrostatic plotters produce only two levels: the white (or light gray) of the paper and the black (or dark gray) of the ink or toner deposited on the paper. Certain techniques, discussed in later sections, allow such inherently *bilevel* devices to produce additional intensity levels.

13.1.1 Selecting Intensities—Gamma Correction

Suppose we want to display 256 different intensities. Which 256 intensity levels should we use? We surely do not want 128 in the range of 0 to 0.1 and 128 more in the range of 0.9 to 1.0, since the transition from 0.1 to 0.9 would certainly appear discontinuous. We might initially distribute the levels evenly over the range 0 to 1, but this choice ignores an important characteristic of the eye: that it is sensitive to ratios of intensity levels rather than to absolute values of intensity. That is, we perceive the intensities 0.10 and 0.11 as differing just as much as the intensities 0.50 and 0.55. (This nonlinearity is easy to observe: Cycle through the settings on a three-way 50–100–150-watt lightbulb; you will see that the step from 50 to 100 seems much greater than the step from 100 to 150.) On a brightness (that is, perceived intensity) scale, the differences between intensities of 0.10 and 0.11 and between intensities of 0.50 and 0.55 are equal. Therefore, the intensity levels should be spaced logarithmically rather than linearly, to achieve equal steps in brightness.

To find 256 intensities starting with the lowest attainable intensity I_0 and going to a maximum intensity of 1.0, with each intensity r times higher than the preceding intensity, we use the following relations:

$$I_0 = I_0,\ I_1 = rI_0,\ I_2 = rI_1 = r^2I_0,\ I_3 = rI_2 = r^3I_0,\ \ldots,\ I_{255} = r^{255}I_0 = 1. \quad (13.1)$$

Therefore,

$$r = (1/I_0)^{1/255},\ I_j = r^jI_0 = (1/I_0)^{j/255}\,I_0 = I_0^{(255-j)/255} \quad \text{for } 0 \le j \le 255, \quad (13.2)$$

and in general for $n + 1$ intensities,

$$r = (1/I_0)^{1/n},\ I_j = I_0^{(n-j)/n} \quad \text{for } 0 \le j \le n. \quad (13.3)$$

With just four intensities ($n = 3$) and an I_0 of $\frac{1}{8}$ (an unrealistically large value chosen for illustration only), Eq. (13.3) tells us that $r = 2$, yielding intensity values of $\frac{1}{8}$, $\frac{1}{4}$, $\frac{1}{2}$, and 1.

The minimum attainable intensity I_0 for a CRT is anywhere from about $\frac{1}{200}$ up to $\frac{1}{40}$ of the maximum intensity of 1.0. Therefore, typical values of I_0 are between 0.005 and 0.025. The minimum is not 0, because of light reflection from the phosphor within the CRT. The ratio between the maximum and minimum intensities is called the *dynamic range*. The exact value for a specific CRT can be found by displaying a square of white on a field of black and measuring the two intensities with a photometer. This measurement is taken in a completely darkened room, so that reflected ambient light does not affect the intensities. With an I_0 of 0.02, corresponding to a dynamic range of 50, Eq. (13.2) yields $r = 1.0154595\ldots$, and the first few and last two intensities of the 256 intensities from Eq. (13.1) are 0.0200, 0.0203, 0.0206, 0.0209, 0.0213, 0.0216, . . ., 0.9848, 1.0000.

Displaying the intensities defined by Eq. (13.1) on a CRT is a tricky process, and recording them on film is even more difficult, because of the nonlinearities in the CRT and film. For instance, the intensity of light output by a phosphor is related to the number of electrons N in the beam by

$$I = kN^{\gamma} \quad (13.4)$$

for constants k and γ. The value of γ is in the range 2.2 to 2.5 for most CRTs. The number of electrons N is proportional to the control-grid voltage, which is in turn proportional to the value V specified for the pixel. Therefore, for some other constant K,

$$I = KV^{\gamma},\ \text{or}\ V = (I/K)^{1/\gamma}. \quad (13.5)$$

Now, given a desired intensity I, we first determine the nearest I_j by searching through a table of the available intensities as calculated from Eq. (13.1) or from its equivalent:

$$j = ROUND(log_r(I/I_0)). \quad (13.6)$$

After j is found, we calculate

$$I_j = r^jI_0. \quad (13.7)$$

The next step is to determine the pixel value V_j needed to create the intensity I_j, by using Eq. (13.5):

$$V_j = ROUND\ ((I_j/K)^{1/\gamma}). \quad (13.8)$$

If the raster display has no look-up table, then V_j is placed in the appropriate pixels. If there is a look-up table, then j is placed in the pixel and V_j is placed in entry j of the table.

The values of K, γ, and I_0 depend on the CRT in use, so in practice the look-up table is loaded by a method based on actual measurement of intensities [CATM79; COWA83; HALL89]. Use of the look-up table in this general manner is called *gamma correction*, so named for the exponent in Eq. (13.4). If the display has hardware gamma correction, then I_j rather than V_j is placed in either the refresh buffer or look-up table.

Without the use of ratio-based intensity values and gamma correction, quantization errors (from approximating a true intensity value with a displayable intensity value) will be more conspicuous near black than near white. For instance, with 4 bits and hence 16 intensities, a quantization round-off error of as much as $\frac{1}{32} = 0.031$ is possible. This is 50 percent of intensity value $\frac{1}{16}$, and only 3 percent of intensity value 1.0. Using the ratio-based intensities and gamma correction, the maximum quantization error as a percentage of brightness (perceived intensity) is constant.

A natural question is, "How many intensities are enough?" By "enough," we mean the number needed to reproduce a continuous-tone black-and-white image such that the reproduction appears to be continuous. This appearance is achieved when the ratio r is 1.01 or less (below this ratio, the eye cannot distinguish between intensities I_j and I_{j+1}) [WYSZ82, p. 569]. Thus, the appropriate value for n, the number of intensity levels, is found by equating r to 1.01 in Eq. (13.3):

$$r = (1/I_0)^{1/n} \quad \text{or} \quad 1.01 = (1/I_0)^{1/n}. \quad (13.9)$$

TABLE 13.1 DYNAMIC RANGE ($1/I_0$) AND NUMBER OF REQUIRED INTENSITIES $n = \log_{1.01}(1/I_0)$ FOR SEVERAL DISPLAY MEDIA

Display media	Typical dynamic range	Number of intensities, n
CRT	50–200	400–530
Photographic prints	100	465
Photographic slides	1000	700
Coated paper printed in B/W*	100	465
Coated paper printed in color	50	400
Newsprint printed in B/W	10	234

*B/W = black and white.

Solving for n gives

$$n = \log_{1.01}(1/I_0), \tag{13.10}$$

where $1/I_0$ is the dynamic range of the device.

The dynamic range $1/I_0$ for several display media, and the corresponding n, which is the number of intensity levels needed to maintain $r = 1.01$ and at the same time to use the full dynamic range, are shown in Table 13.1. These are theoretical values, assuming perfect reproduction processes. In practice, slight blurring due to ink bleeding and small amounts of random noise in the reproduction decreases n considerably for print media. For instance, Fig. 13.1 shows a continuous-tone photograph; and the succeeding five Figs. 13.2 to 13.6 reproduce the same photograph at 4, 8, 16, 32, and 64 intensity levels. With four and eight levels, the transitions or contours between one intensity level and the next are quite conspicuous, because the ratio r between successive intensities is considerably greater that the ideal 1.01. Contouring is barely detectable with 32 levels, and for these particular images disappears with 64. This suggests that 64 intensity levels is the absolute minimum needed for contour-free printing of continuous-tone black-and-white images on paper such as is used in this book. For a well-adjusted CRT in a perfectly black room, however, the higher dynamic range means that many more levels are demanded.

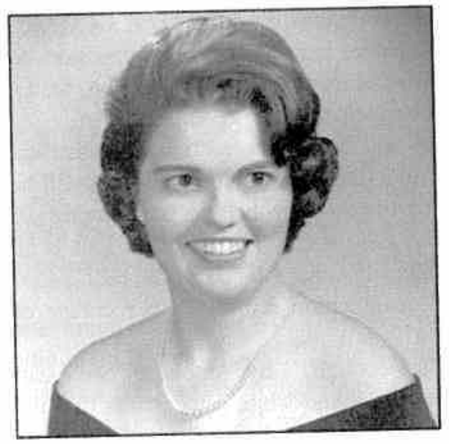
Fig. 13.1 A continuous-tone photograph.

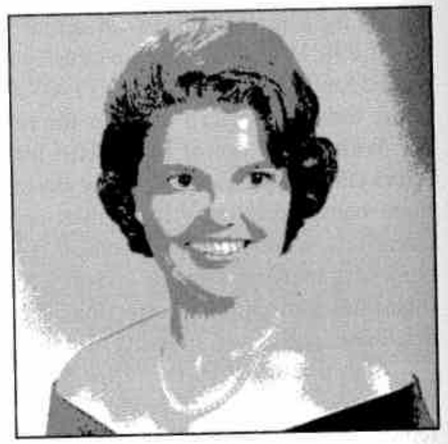
Fig. 13.2 A continuous-tone photograph reproduced with four intensity levels. (Courtesy of Alan Paeth, University of Waterloo Computer Graphics Lab.)

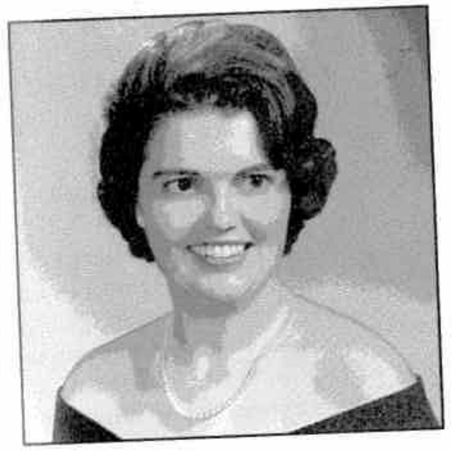
Fig. 13.3 A continuous-tone photograph reproduced with eight intensity levels. (Courtesy of Alan Paeth, University of Waterloo Computer Graphics Lab.)

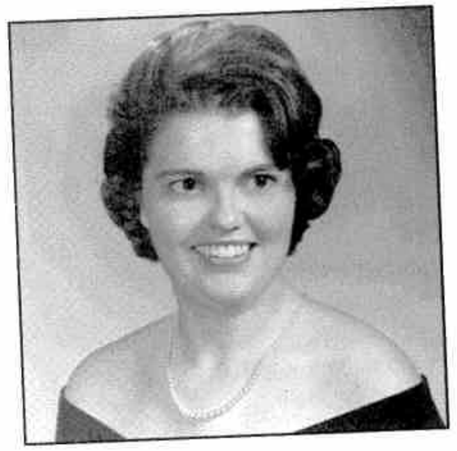
Fig. 13.4 A continuous-tone photograph reproduced with 16 intensity levels. (Courtesy of Alan Paeth, University of Waterloo Computer Graphics Lab.)

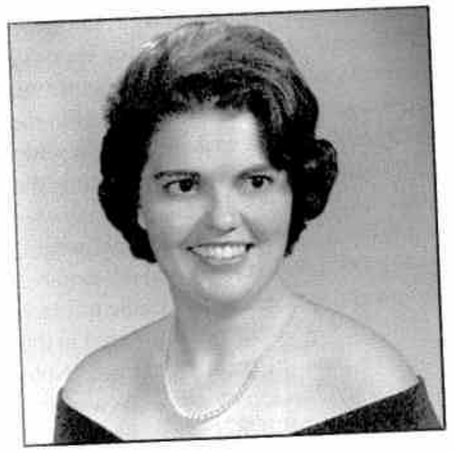
Fig. 13.5 A continuous-tone photograph reproduced with 32 intensity levels. (Courtesy of Alan Paeth, University of Waterloo Computer Graphics Lab.)

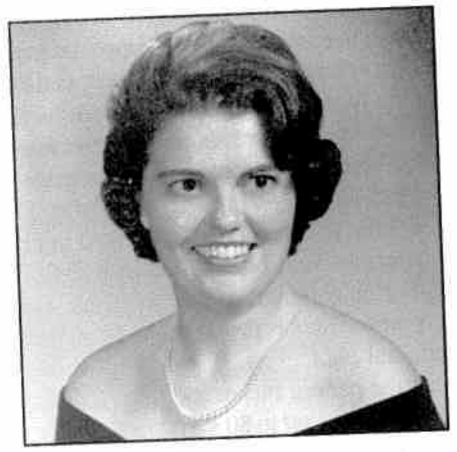
Fig. 13.6 A continuous-tone photograph reproduced with 64 intensity levels. Differences from the picture in Fig. 13.5 are quite subtle. (Courtesy of Alan Paeth, University of Waterloo Computer Graphics Lab.)

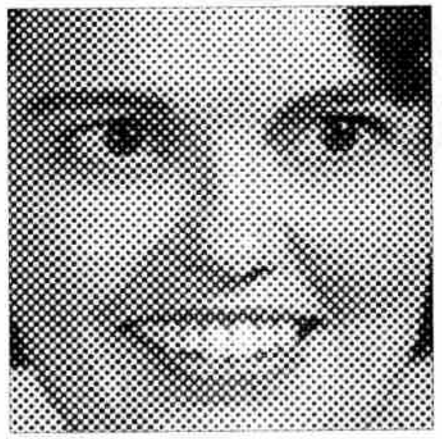

Fig. 13.7 Enlarged halftone pattern. Dot sizes vary inversely with intensity of original photograph. (Courtesy of Alan Paeth, University of Waterloo Computer Graphics Lab.)

13.1.2 Halftone Approximation

Many displays and hardcopy devices are bilevel—they produce just two intensity levels—and even 2- or 3-bit-per-pixel raster displays produce fewer intensity levels than we might desire. How can we expand the range of available intensities? The answer lies in the *spatial integration* that our eyes perform. If we view a very small area from a sufficiently large viewing distance, our eyes average fine detail within the small area and record only the overall intensity of the area.

This phenomenon is exploited in printing black-and-white photographs in newspapers, magazines, and books, in a technique called *halftoning* (also called *clustered-dot ordered dither*[1] in computer graphics). Each small resolution unit is imprinted with a circle of black ink whose area is proportional to the blackness $1 - I$ (where I = intensity) of the area in the original photograph. Figure 13.7 shows part of a halftone pattern, greatly enlarged. Note that the pattern makes a 45° angle with the horizontal, called the *screen angle*. Newspaper halftones use 60 to 80 variable-sized and variable-shaped areas [ULIC87] per inch, whereas halftones in magazines and books use 110 to 200 per inch.

Graphics output devices can approximate the variable-area circles of halftone reproduction. For example, a 2×2 pixel area of a bilevel display can be used to produce five different intensity levels at the cost of halving the spatial resolution along each axis. The patterns shown in Fig. 13.8 can be used to fill the 2×2 areas with the number of "on" pixels that is proportional to the desired intensity. Figure 13.9 shows a face digitized as a 351×351 image array and displayed with 2×2 patterns.

An $n \times n$ group of bilevel pixels can provide $n^2 + 1$ intensity levels. In general, there is a tradeoff between spatial resolution and intensity resolution. The use of a 3×3 pattern

[1]The "ordered dither" contrasts with "random dither," an infrequently used technique.

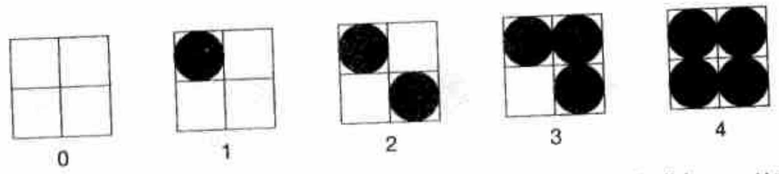

Fig. 13.8 Five intensity levels approximated with four 2×2 dither patterns.

cuts spatial resolution by one-third on each axis, but provides 10 intensity levels. Of course, the tradeoff choices are limited by our visual acuity (about 1 minute of arc in normal lighting), the distance from which the image is viewed, and the dots-per-inch resolution of the graphics device.

One possible set of patterns for the 3×3 case is shown in Fig. 13.10. Note that these patterns can be represented by the *dither matrix*

$$\begin{bmatrix} 6 & 8 & 4 \\ 1 & 0 & 3 \\ 5 & 2 & 7 \end{bmatrix}. \tag{13.11}$$

To display an intensity I, we turn on all pixels whose values are less than I.

The $n \times n$ pixel patterns used to approximate the halftones must be designed not to introduce visual artifacts in an area of identical intensity values. For instance, if the pattern in Fig. 13.11 were used, rather than the one in Fig. 13.10, horizontal lines would appear in any large area of the image of intensity 3. Second, the patterns must form a *growth sequence* so that any pixel intensified for intensity level j is also intensified for all levels $k > j$. This minimizes the differences in the patterns for successive intensity levels, thereby minimizing the contouring effects. Third, the patterns must grow outward from the center, to create the effect of increasing dot size. Fourth, for hardcopy devices such as laser printers and film recorders that are poor at reproducing isolated "on" pixels, all pixels that are "on" for a

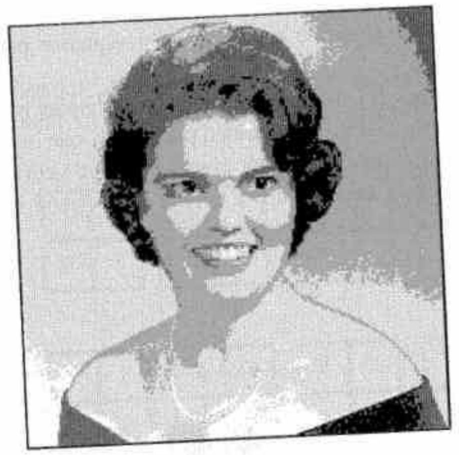

Fig. 13.9 A continuous-tone photograph, digitized to a resolution of 351×351 and displayed using the 2×2 patterns of Fig. 13.8. (Courtesy of Alan Paeth, University of Waterloo Computer Graphics Lab.)

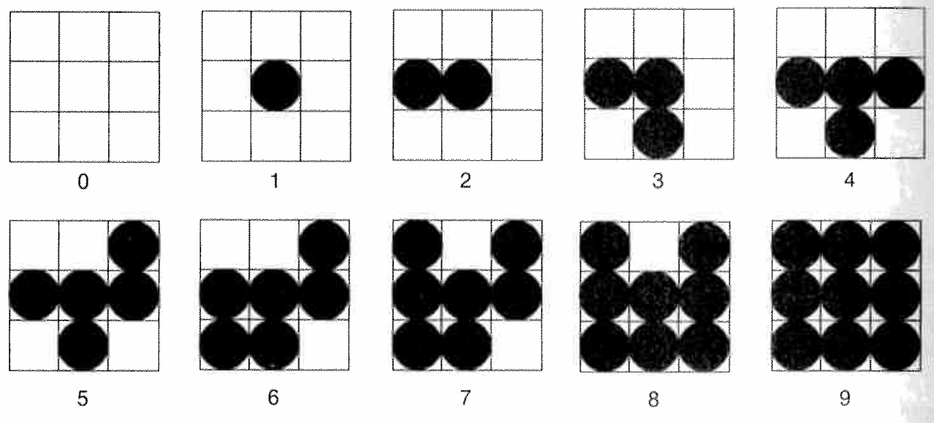

Fig. 13.10 Ten intensity levels approximated with 3 × 3 dither patterns.

particular intensity must be adjacent to other "on" pixels; a pattern such as that in Fig. 13.12 is not acceptable. This is the meaning of the term *clustered-dot* in "clustered-dot ordered dither." Holladay [HOLL80] has developed a widely used method for defining dither matrices for clustered-dot ordered dither. For high-quality reproduction of images, n must be 8 to 10, theoretically allowing 65 to 101 intensity levels. To achieve an effect equivalent to the 150-circle-per-inch printing screen, we thus require a resolution of from 150 × 8 = 1200 up to 150 × 10 = 1500 pixels per inch. High-quality film recorders can attain this resolution, but cannot actually show all the intensity levels because patterns made up of single black or white pixels may disappear.

Halftone approximation is not limited to bilevel displays. Consider a display with 2 bits per pixel and hence four intensity levels. The halftone technique can be used to increase the number of intensity levels. If we use a 2 × 2 pattern, we have a total of 4 pixels at our disposal, each of which can take on three values besides black; this allows us to display 4 × 3 + 1 = 13 intensities. One possible set of growth sequence patterns in this situation is shown in Fig. 13.13.

Unlike a laser printer, a CRT display is quite able to display individual dots. Hence, the clustering requirement on the patterns discussed previously can be relaxed and *dispersed-dot ordered dither* can be used. There are many possible dither matrices: Bayer [BAYE73] has developed dither matrices that minimize the texture introduced into the displayed images. For the 2 × 2 case, the dither matrix, called $D^{(2)}$, is

$$D^{(2)} = \begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix}. \qquad (13.12)$$

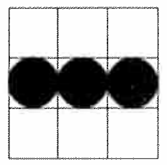

Fig. 13.11 A 3 × 3 dither pattern inappropriate for halftoning.

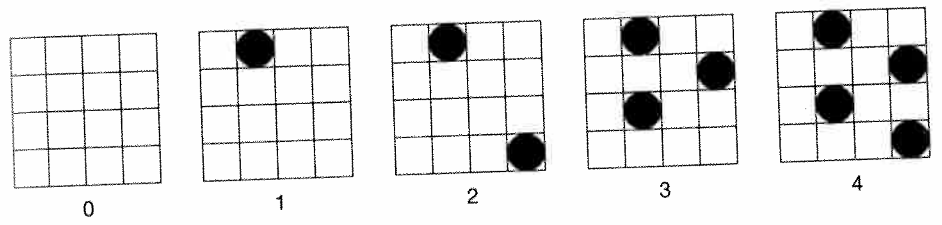

Fig. 13.12 Part of a 4 × 4 ordered dither dot pattern in which several of the patterns have single, nonadjacent dots. Such disconnected patterns are unacceptable for halftoning on laser printers and for printing presses.

This represents the set of patterns of Figure 13.8.

Larger dither matrices can be found using a recurrence relation [JUDI74] to compute $D^{(2n)}$ from $D^{(n)}$. With $U^{(n)}$ defined as an $n \times n$ matrix of 1s, that is,

$$U^{(n)} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & 1 & 1 & \dots & 1 \\ 1 & 1 & 1 & \dots & 1 \\ . & . & . & . & . \\ 1 & 1 & 1 & \dots & 1 \end{bmatrix}, \qquad (13.13)$$

the recurrence relation is

$$D^{(n)} = \begin{bmatrix} 4D^{(n/2)} + D^{(2)}_{00}U^{(n/2)} & 4D^{(n/2)} + D^{(2)}_{01}U^{(n/2)} \\ 4D^{(n/2)} + D^{(2)}_{10}U^{(n/2)} & 4D^{(n/2)} + D^{(2)}_{11}U^{(n/2)} \end{bmatrix}. \qquad (13.14)$$

Applying this relation to $D^{(2)}$ produces

$$D^{(4)} = \begin{bmatrix} 0 & 8 & 2 & 10 \\ 12 & 4 & 14 & 6 \\ 3 & 11 & 1 & 9 \\ 15 & 7 & 13 & 5 \end{bmatrix}. \qquad (13.15)$$

The techniques presented thus far have assumed that the image array being shown is smaller than the display device's pixel array, so that multiple display pixels can be used for one

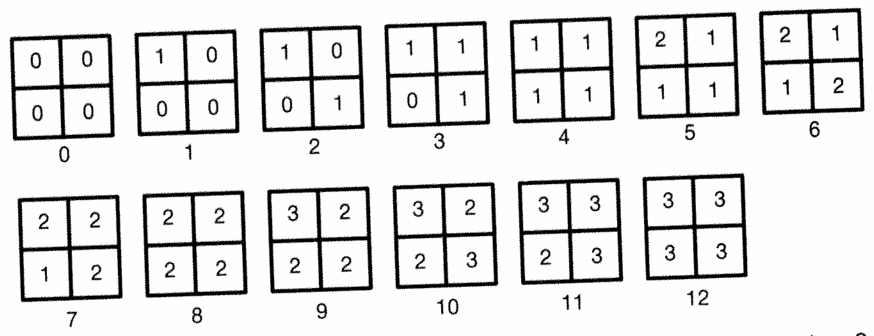

Fig. 13.13 Dither patterns for intensity levels 0 to 13 approximated using 2 × 2 patterns of pixels, with 2 bits (four intensity levels) per pixel. The intensities of the individual pixels sum to the intensity level for each pattern.

image pixel. What if the image and display device arrays are the same size? A simple adaptation of the ordered-dither (either clustered-dot or dispersed-dot) approach can be used. Whether or not to intensify the pixel at point (x, y) depends on the desired intensity $S(x, y)$ at that point and on the dither matrix. To display the point at (x, y), we compute

$$i = x \text{ modulo } n,$$
$$j = y \text{ modulo } n. \tag{13.16}$$

Then, if

$$S(x, y) > D_{ij}^{(n)}, \tag{13.17}$$

the point at (x, y) is intensified; otherwise, it is not. That is, 1 pixel in the image array controls 1 pixel in the display array. Notice that large areas of fixed image intensity are displayed exactly as when the image-array size is less than the display-array size, so the effect of equal image and display arrays is apparent only in areas where intensity varies.

Figure 13.14(a) is a face digitized at 512×512 and shown on a 512×512 bilevel display using $D^{(8)}$. Compare this bilevel result to the multilevel pictures shown earlier in this section. Further pictures displayed by means of ordered dither appear in [JARV76a; JARV76b; ULIC87], where more ways to display continuous-tone images on bilevel displays are described.

Error diffusion, another way to handle the case when the image and display array sizes are equal, was developed by Floyd and Steinberg [FLOY75]; its visual results are often satisfactory. The error (i.e., the difference between the exact pixel value and the approximated value actually displayed) is added to the values of the four image-array pixels to the right of and below the pixel in question: $\frac{7}{16}$ of the error to the pixel to the right, $\frac{3}{16}$ to the pixel below and to the left, $\frac{5}{16}$ to the pixel immediately below, and $\frac{1}{16}$ to the pixel below

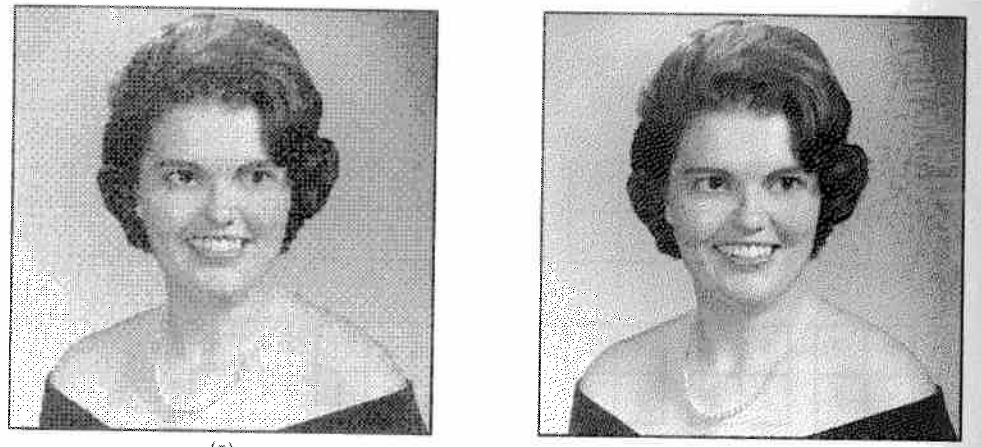

Fig. 13.14 A continuous-tone photograph reproduced with (a) $D^{(8)}$ ordered dither, and (b) Floyd-Steinberg error diffusion. (Courtesy of Alan Paeth, University of Waterloo Computer Graphics Lab.)

and to the right. This has the effect of spreading, or diffusing, the error over several pixels in the image array. Figure 13.14(b) was created using this method.

Given a picture S to be displayed in the intensity matrix I, the modified values in S and the displayed values in I are computed for pixels in scan-line order, working downward from the topmost scanline.

```
double error;
K = Approximate (S[x][y]);     /* Approximate S to nearest displayable intensity */
I[x][y] = K;                   /* Draw the pixel at (x, y). */
error = S[x][y] - K;           /* Error term */

/* Step 1: spread 7/16 of error into the pixel to the right, at (x + 1, y). */
S[x + 1][y] += 7 * error/16;

/* Step 2: spread 3/16 of error into pixel below and to the left. */
S[x - 1][y - 1] += 3 * error/16;

/* Step 3: spread 5/16 of error into pixel below. */
S[x][y - 1] += 5 * error/16;

/* Step 4: spread 1/16 of error below and to the right. */
S[x + 1][y - 1] += error/16;
```

To avoid introducing visual artifacts into the displayed image, we must ensure that the four errors sum exactly to *error*; no roundoff errors can be allowed. This can be done by calculating the step 4 error term as *error* minus the error terms from the first three steps. The function *Approximate* returns the displayable intensity value closest to the actual pixel value. For a bilevel display, the value of S is simply rounded to 0 or 1.

Even better results can be obtained by alternately scanning left to right and right to left; on a right-to-left scan, the left–right directions for errors in steps 1, 2, and 4 are reversed. For a detailed discussion of this and other error-diffusion methods, see [ULIC87]. Other approaches are discussed in [KNUT87].

Suppose the size of the image array is less than the size of the display array, the number of intensities in the image and display are equal, and we wish to display the image at the size of the display array. A simple case of this is an 8-bit-per-pixel, 512×512 image and an 8-bit-per-pixel, 1024×1024 display. If we simply replicate image pixels horizontally and vertically in the display array, the replicated pixels on the display will form squares that are quite obvious to the eye. To avoid this problem, we can interpolate intensities to calculate the pixel values. For instance, if an image S is to be displayed at double resolution, then the intensities I to display (with $x = 2x'$ and $y = 2y'$) are

$$I[x][y] = S[x'][y'];$$
$$I[x+1][y] = \tfrac{1}{2}(S[x'][y'] + S[x'+1][y']);$$
$$I[x][y+1] = \tfrac{1}{2}(S[x'][y'] + S[x'][y'+1]);$$
$$I[x+1][y+1] = \tfrac{1}{4}(S[x'][y'] + S[x'+1][y'] + S[x'][y'+1] + S[x'+1][y'+1]);$$

See Section 17.4 for a discussion of two-pass scaling transformations of images, and Section 3.17 for a description of the filtering and image-reconstruction techniques that are applicable to this problem.

13.2 CHROMATIC COLOR

The visual sensations caused by colored light are much richer than those caused by achromatic light. Discussions of color perception usually involve three quantities, known as hue, saturation, and lightness. *Hue* distinguishes among colors such as red, green, purple, and yellow. *Saturation* refers to how far color is from a gray of equal intensity. Red is highly saturated; pink is relatively unsaturated; royal blue is highly saturated; sky blue is relatively unsaturated. Pastel colors are relatively unsaturated; unsaturated colors include more white light than do the vivid, saturated colors. *Lightness* embodies the achromatic notion of perceived intensity of a reflecting object. *Brightness*, a fourth term, is used instead of lightness to refer to the perceived intensity of a self-luminous (i.e., emitting rather than reflecting light) object, such as a light bulb, the sun, or a CRT.

It is necessary to specify and measure colors if we are to use them precisely in computer graphics. For reflected light, we can do this by visually comparing a sample of unknown color against a set of "standard" samples. The unknown and sample colors must be viewed under a standard light source, since the perceived color of a surface depends both on the surface and on the light under which the surface is viewed. The widely used Munsell color-order system includes sets of published standard colors [MUNS76] organized in a 3D space of hue, value (what we have defined as lightness), and chroma (saturation). Each color is named, and is ordered so as to have an equal perceived "distance" in color space (as judged by many observers) from its neighbors. [KELL76] gives an extensive discussion of standard samples, charts depicting the Munsell space, and tables of color names.

In the printing industry and graphic design profession, colors are typically specified by matching to printed color samples. Color Plate I.33 is taken from a widely used color-matching system.

Artists often specify color as different tints, shades, and tones of strongly saturated, or pure, pigments. A *tint* results from adding white pigment to a pure pigment, thereby decreasing saturation. A *shade* comes from adding a black pigment to a pure pigment, thereby decreasing lightness. A *tone* is the consequence of adding both black and white pigments to a pure pigment. All these steps produce different colors of the same hue, with varying saturation and lightness. Mixing just black and white pigments creates grays. Figure 13.15 shows the relationship of tints, shades, and tones. The percentage of pigments that must be mixed to match a color can be used as a color specification. The Ostwald [OSTW31] color-order system is similar to the artist's model.

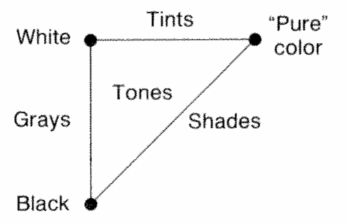

Fig. 13.15 Tints, tones, and shades.

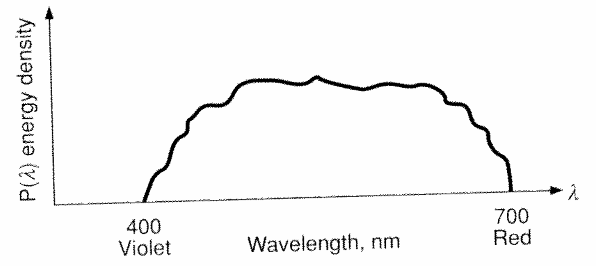

Fig. 13.16 Typical spectral energy distribution $P(\lambda)$ of a light.

13.2.1 Psychophysics

The Munsell and artists' pigment-mixing methods are subjective: They depend on human observers' judgments, the lighting, the size of the sample, the surrounding color, and the overall lightness of the environment. An objective, quantitative way of specifying colors is needed, and for this we turn to the branch of physics known as *colorimetry*. Important terms in colorimetry are dominant wavelength, excitation purity, and luminance.

Dominant wavelength is the wavelength of the color we "see" when viewing the light, and corresponds to the perceptual notion of hue[2]; *excitation purity* corresponds to the saturation of the color; *luminance* is the amount or intensity of light. The excitation purity of a colored light is the proportion of pure light of the dominant wavelength and of white light needed to define the color. A completely pure color is 100 percent saturated and thus contains no white light, whereas mixtures of a pure color and white light have saturations somewhere between 0 and 100 percent. White light and hence grays are 0 percent saturated, containing no color of any dominant wavelength. The correspondences between these perceptual and colorimetry terms are as follows:

Perceptual term	**Colorimetry**
Hue	Dominant wavelength
Saturation	Excitation purity
Lightness (reflecting objects)	Luminance
Brightness (self-luminous objects)	Luminance

Fundamentally, light is electromagnetic energy in the 400- to 700-nm wavelength part of the spectrum, which is perceived as the colors from violet through indigo, blue, green, yellow, and orange to red. Color Plate I.34 shows the colors of the spectrum. The amount of energy present at each wavelength is represented by a spectral energy distribution $P(\lambda)$, such as shown in Fig. 13.16 and Color Plate I.34. The distribution represents an infinity of numbers, one for each wavelength in the visible spectrum (in practice, the distribution is represented by a large number of sample points on the spectrum, as measured by a

[2]Some colors, such as purple, have no true dominant wavelength, as we shall see later.

spectroradiometer). Fortunately, we can describe the visual effect of any spectral distribution much more concisely by the triple (dominant wavelength, excitation purity, luminance). This implies that many different spectral energy distributions produce the same color: They "look" the same. Hence the relationship between spectral distributions and colors is many-to-one. Two spectral energy distributions that look the same are called *metamers*.

Figure 13.17 shows one of the infinitely many spectral distributions $P(\lambda)$, or metamers, that produces a certain color sensation. At the dominant wavelength, there is a spike of energy of level e_2. White light, the uniform distribution of energy at level e_1, is also present. The excitation purity depends on the relation between e_1 and e_2: when $e_1 = e_2$, excitation purity is 0 percent; when $e_1 = 0$, excitation purity is 100 percent. Brightness, which is proportional to the integral of the product of the curve and the luminous efficiency function (defined later), depends on both e_1 and e_2. In general, spectral distributions may be more complex than the one shown, and it is not possible to determine the dominant wavelength merely by looking at the spectral distributions. In particular, the dominant wavelength may *not* be the one whose component in the spectral distribution is largest.

How does this discussion relate to the red, green, and blue phosphor dots on a color CRT? And how does it relate to the *tristimulus theory* of color perception, which is based on the hypothesis that the retina has three kinds of color sensors (called cones), with peak sensitivity to red, green, or blue lights? Experiments based on this hypothesis produce the spectral-response functions of Fig. 13.18. The peak blue response is around 440 nm; that for green is about 545 nm; that for red is about 580 nm. (The terms "red" and "green" are somewhat misleading here, as the 545 nm and 580 nm peaks are actually in the yellow range.) The curves suggest that the eye's response to blue light is much less strong than is its response to red or green.

Figure 13.19 shows the *luminous-efficiency function*, the eye's response to light of constant luminance, as the dominant wavelength is varied: our peak sensitivity is to yellow-green light of wavelength around 550 nm. There is experimental evidence that this curve is just the sum of the three curves shown in Fig. 13.18.

The tristimulus theory is intuitively attractive because it corresponds loosely to the notion that colors can be specified by positively weighted sums of red, green, and blue (the so-called primary colors). This notion is almost true: The three color-matching functions in

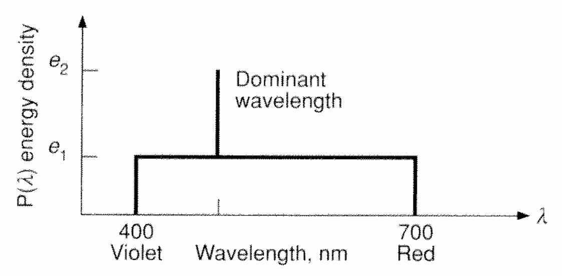

Fig. 13.17 Spectral energy distribution, $P(\lambda)$, illustrating dominant wavelength, excitation purity, and luminance.

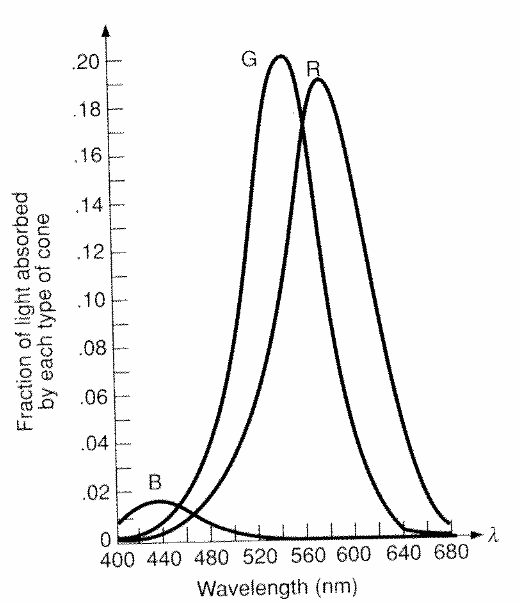

Fig. 13.18 Spectral-response functions of each of the three types of cones on the human retina.

Fig. 13.20 show the amounts of red, green, and blue light needed by an average observer to match a color of constant luminance, for all values of dominant wavelength in the visible spectrum.

A negative value in Fig. 13.20 means that we cannot match the color by adding together the primaries. However, if one of the primaries is added to the color sample, the sample can then be matched by a mixture of the other two primaries. Hence, negative values in Fig. 13.20 indicate that the primary was added to the color being matched. The need for negative values does not mean that the notion of mixing red, green, and blue to obtain other

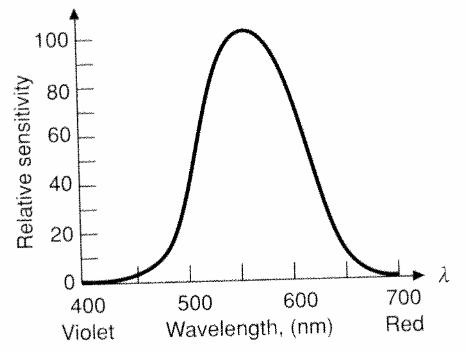

Fig. 13.19 Luminous-efficiency function for the human eye.

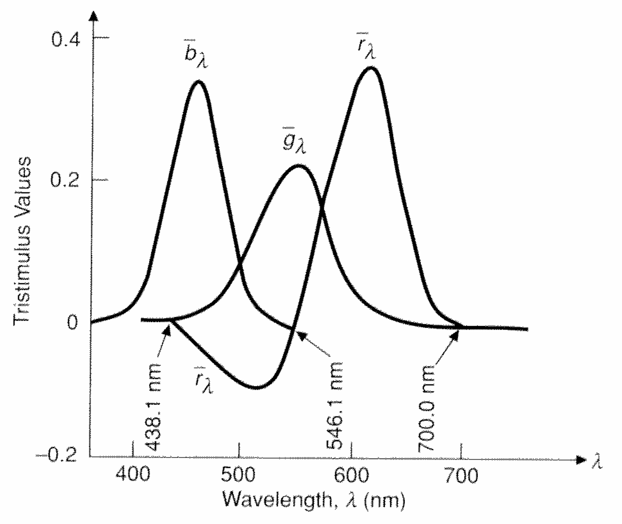

Fig. 13.20 Color-matching functions, showing the amounts of three primaries needed to match all the wavelengths of the visible spectrum.

colors is invalid; on the contrary, a huge range of colors can be matched by positive amounts of red, green, and blue. Otherwise, the color CRT would not work! It does mean, however, that certain colors cannot be produced by RGB mixes, and hence cannot be shown on an ordinary CRT.

The human eye can distinguish hundreds of thousands of different colors in color space, when different colors are judged side by side by different viewers who state whether the colors are the same or different. As shown in Fig. 13.21, when colors differ only in hue, the wavelength between just noticably different colors varies from more than 10 nm at the extremes of the spectrum to less than 2 nm around 480 nm (blue) and 580 nm (yellow).

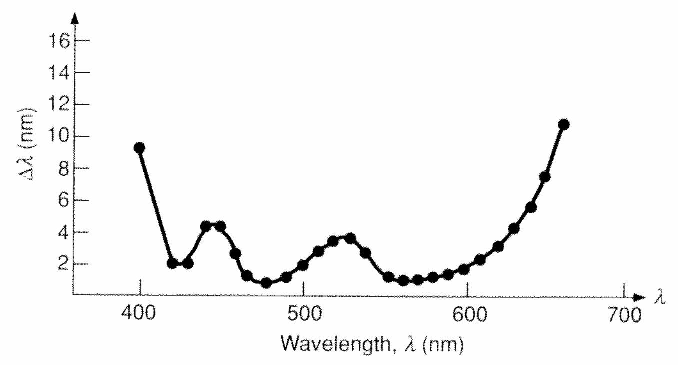

Fig. 13.21 Just-noticable color differences as a function of wavelength λ. The ordinate indicates the minimum detectable difference in wavelength between adjacent color samples. (Source: Data are from [BEDF58].)

Except at the spectrum extremes, however, most distinguished hues are within 4 nm. Altogether about 128 fully saturated hues can be distinguished.

The eye is less sensitive to hue changes in less saturated light. This is not surprising: As saturation tends to 0 percent, all hues tend to white. Sensitivity to changes in saturation for a fixed hue and lightness is greater at the extremes of the visible spectrum, where about 23 distinguishable steps exist. Around 575 nm, only 16 saturation steps can be distinguished [JONE26].

13.2.2 The CIE Chromaticity Diagram

Matching and therefore defining a colored light with a mixture of three fixed primaries is a desirable approach to specifying color, but the need for negative weights suggested by Fig. 13.20 is awkward. In 1931, the *Commission Internationale de l'Éclairage* (*CIE*) defined three standard primaries, called **X**, **Y**, and **Z**, to replace red, green, and blue in this matching process. The three corresponding color-matching functions, $\bar{x}_\lambda$, $\bar{y}_\lambda$, and $\bar{z}_\lambda$, are shown in Fig. 13.22. The primaries can be used to match, with only positive weights, all the colors we can see. The **Y** primary was intentionally defined to have a color-matching function $\bar{y}_\lambda$ that exactly matches the luminous-efficiency function of Fig. 13.19. Note that $\bar{x}_\lambda$, $\bar{y}_\lambda$, and $\bar{z}_\lambda$ are not the spectral distributions of the **X**, **Y**, and **Z** colors, just as the curves in Fig. 13.20 are not the spectral distributions of red, green, and blue. They are merely auxilliary functions used to compute how much of **X**, **Y**, and **Z** should be mixed together to generate a metamer of any visible color.

The color-matching functions are defined tabularly at 1-nm intervals, and are found in texts such as [WYSZ82; BILL81]. The distributions were defined for color samples that

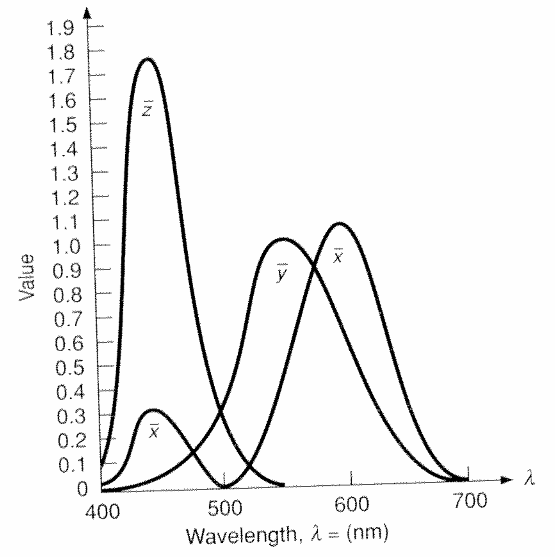

Fig. 13.22 The color-matching functions $\bar{x}_\lambda$, $\bar{y}_\lambda$, and $\bar{z}_\lambda$, for the 1931 CIE **X**, **Y**, **Z** primaries.

subtend a 2° field of view on the retina. The original 1931 tabulation is normally used in work relevant to computer graphics. A later 1964 tabulation for a 10° field of view is not generally useful, because it emphasizes larger areas of constant color than are normally found in graphics.

The three CIE color-matching functions are linear combinations of the color-matching functions from Fig. 13.20. This means that the definition of a color with red, green, and blue lights can be converted via a linear transformation into its definition with the CIE primaries, and vice versa.

The amounts of **X**, **Y**, and **Z** primaries needed to match a color with a spectral energy distribution $P(\lambda)$, are:

$$X = k \int P(\lambda)\, \bar{x}_\lambda \, d\lambda, \quad Y = k \int P(\lambda)\, \bar{y}_\lambda \, d\lambda, \quad Z = k \int P(\lambda)\, \bar{z}_\lambda \, d\lambda. \qquad (13.18)$$

For self-luminous objects like a CRT, k is 680 lumens/watt. For reflecting objects, k is usually selected such that bright white has a Y value of 100; then other Y values will be in the range of 0 to 100. Thus,

$$k = \frac{100}{\int P_w(\lambda)\bar{y}_\lambda \, d\lambda} \qquad (13.19)$$

where $P_w(\lambda)$ is the spectral energy distribution for whatever light source is chosen as the standard for white. In practice, these integrations are performed by summation, as none of the energy distributions are expressed analytically.

Figure 13.23 shows the cone-shaped volume of XYZ space that contains visible colors. The volume extends out from the origin into the postive octant, and is capped at the smooth curved line terminating the cone.

Let (X, Y, Z) be the weights applied to the CIE primaries to match a color **C**, as found using Eq. (13.18). Then $\mathbf{C} = X\,\mathbf{X} + Y\,\mathbf{Y} + Z\,\mathbf{Z}$. We define *chromaticity* values (which depend only on dominant wavelength and saturation and are independent of the amount of

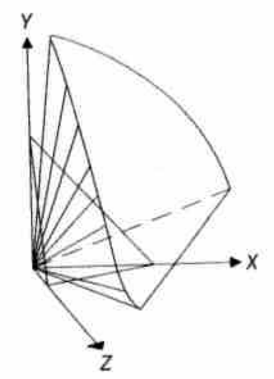

Fig. 13.23 The cone of visible colors in CIE color space is shown by the lines radiating from the origin. The $X + Y + Z = 1$ plane is shown. (Courtesy of Gary Meyer, Program of Computer Graphics, Cornell University, 1978.)

luminous energy) by normalizing against $X + Y + Z$, which can be thought of as the total amount of light energy:

$$x = \frac{X}{(X + Y + Z)}, \; y = \frac{Y}{(X + Y + Z)}, \; z = \frac{Z}{(X + Y + Z)}. \qquad (13.20)$$

Notice that $x + y + z = 1$. That is, x, y, and z are on the $(X + Y + Z = 1)$ plane of Fig. 13.23. Color plate II.1 shows the $X + Y + Z = 1$ plane as part of CIE space, and also shows an orthographic view of the plane along with the projection of the plane onto the (X, Y) plane. This latter projection is just the CIE chromaticity diagram.

If we specify x and y, then z is determined by $z = 1 - x - y$. We cannot recover X, Y, and Z from x and y, however. To recover them, we need one more piece of information, typically Y, which carries luminance information. Given (x, y, Y), the transformation to the corresponding (X, Y, Z) is

$$X = \frac{x}{y}Y, \quad Y = Y, \; Z = \frac{1 - x - y}{y}Y. \qquad (13.21)$$

Chromaticity values depend only on dominant wavelength and saturation and are independent of the amount of luminous energy. By plotting x and y for all visible colors, we obtain the CIE chromaticity diagram shown in Fig. 13.24, which is the projection onto the (X, Y) plane of the $(X + Y + Z = 1)$ plane of Fig. 13.23. The interior and boundary of the horseshoe-shaped region represent all visible chromaticity values. (All perceivable colors with the same chromaticity but different luminances map into the same point within this region.) The 100 percent spectrally pure colors of the spectrum are on the curved part of the boundary. A standard white light, meant to approximate sunlight, is formally defined by a

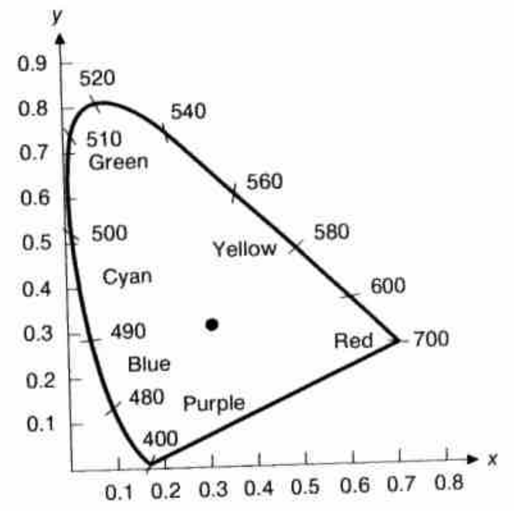

Fig. 13.24 The CIE chromaticity diagram. Wavelengths around the periphery are in nanometers. The dot marks the position of illuminant C.

light source *illuminant C*, marked by the center dot. It is near but not at the point where $x = y = z = \frac{1}{3}$. Illuminant C was defined by specifying a spectral power distribution that is close to daylight at a correlated color temperature of 6774° Kelvin.

The CIE chromaticity diagram is useful in many ways. For one, it allows us to measure the dominant wavelength and excitation purity of any color by matching the color with a mixture of the three CIE primaries. (Instruments called *colorimeters* measure tristimulus X, Y, and Z values, from which chromaticity coordinates are computed using Eq. (13.20). *Spectroradiometers* measure both the spectral energy distribution and the tristimulus values.) Now suppose the matched color is at point A in Fig. 13.25. When two colors are added together, the new color lies somewhere on the straight line in the chromaticity diagram connecting the two colors being added. Therefore, color A can be thought of as a mixture of "standard" white light (illuminant C) and the pure spectral light at point B. Thus, B defines the dominant wavelength. The ratio of length AC to length BC, expressed as a percentage, is the excitation purity of A. The closer A is to C, the more white light A includes and the less pure it is.

The chromaticity diagram factors out luminance, so color sensations that are luminance-related are excluded. For instance; brown, which is an orange-red chromaticity at very low luminance relative to its surrounding area, does not appear. It is thus important to remember that the chromaticity diagram is not a full color palette. There is an infinity of planes in (X, Y, Z) space, each of which projects onto the chromaticity diagram and each of which loses luminance information in the process. The colors found on each such plane are all different.

Complementary colors are those that can be mixed to produce white light (such as D and E in Fig. 13.25). Some colors (such as F in Fig. 13.25) cannot be defined by a

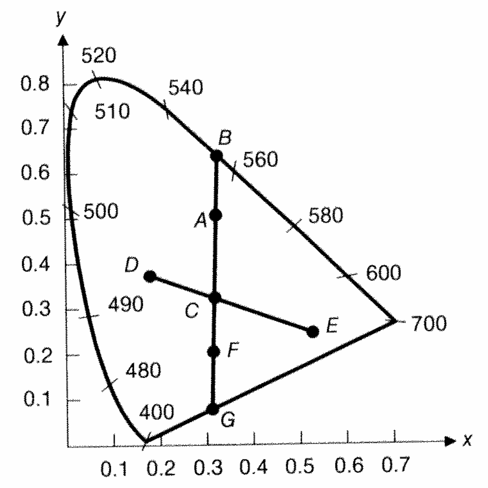

Fig. 13.25 Colors on the chromaticity diagram. The dominant wavelength of color A is that of color B. Colors D and E are complementary colors. The dominant wavelength of color F is defined as the complement of the dominant wavelength of color A.

dominant wavelength and are thus called *nonspectral*. In this case, the dominant wavelength is said to be the complement of the wavelength at which the line through F and C intersects the horseshoe part of the curve at point B, and is designated by a "c" (here about 555 nm c). The excitation purity is still defined by the ratio of lengths (here CF to CG). The colors that must be expressed by using a complementary dominant wavelength are the purples and magentas; they occur in the lower part of the CIE diagram. Complementary colors still can be made to fit the dominant wavelength model of Fig. 13.17, in the sense that if we take a flat spectral distribution and delete some of the light at frequency B, the resulting color will be perceived as F.

Another use of the CIE chromaticity diagram is to define *color gamuts*, or color ranges, that show the effect of adding colors together. Any two colors, say I and J in Fig. 13.26, can be added to produce any color along their connecting line by varying the relative amounts of the two colors being added. A third color K (see Fig. 13.26) can be used with various mixtures of I and J to produce the gamut of all colors in triangle IJK, again by varying relative amounts. The shape of the diagram shows why visible red, green, and blue cannot be additively mixed to match all colors: No triangle whose vertices are within the visible area can completely cover the visible area.

The chromaticity diagram is also used to compare the gamuts available on various color display and hardcopy devices. Color Plate II.2 shows the gamuts for a color television monitor, film, and print. The chromaticity coordinates for the phosphors in two typical monitors are these:

	Short-persistence phosphors			**Long-persistence phosphors**		
	Red	Green	Blue	Red	Green	Blue
x	0.61	0.29	0.15	0.62	0.21	0.15
y	0.35	0.59	0.063	0.33	0.685	0.063

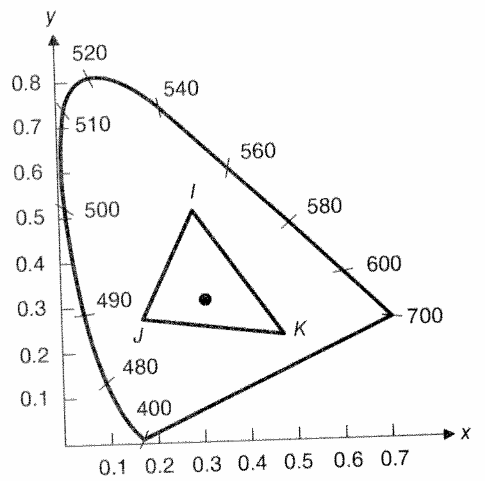

Fig. 13.26 Mixing colors. All colors on the line IJ can be created by mixing colors I and J; all colors in the triangle IJK can be created by mixing colors I, J, and K.

The smallness of the print gamut with respect to the color-monitor gamut suggests that, if images originally seen on a monitor must be faithfully reproduced by printing, a reduced gamut of colors should be used with the monitor. Otherwise, accurate reproduction will not be possible. If, however, the goal is to make a pleasing rather than an exact reproduction, small differences in color gamuts are less important. A discussion of color-gamut compression can be found in [HALL89].

There is a problem with the CIE system. Consider the distance from color $C_1 = (X_1, Y_1, Z_1)$ to color $C_1 + \Delta C$, and the distance from color $C_2 = (X_2, Y_2, Z_2)$ to color $C_2 + \Delta C$, where $\Delta C = (\Delta X, \Delta Y, \Delta Z)$. Both distances are equal to ΔC, yet in general they will not be perceived as being equal. This is because of the variation, throughout the spectrum, of the just noticable differences discussed in Section 13.2.1. A *perceptually uniform* color space is needed, in which two colors that are equally distant are *perceived* as equally distant by viewers.

The 1976 CIE LUV uniform color space was developed in response to this need. With (X_n, Y_n, Z_n) as the coordinates of the color that is to be defined as white, the space is defined by

$$\begin{aligned}
L^* &= 116\,(Y/Y_n)^{1/3} - 16,\ Y/Y_n > 0.01\\
u^* &= 13\,L^*\,(u' - u'_n),\\
v^* &= 13\,L^*\,(v' - v'_n),\\
u' &= \frac{4X}{X + 15Y + 3Z},\quad v' = \frac{9Y}{X + 15Y + 3Z},\\
u'_n &= \frac{4X_n}{X_n + 15Y_n + 3Z_n},\quad v'_n = \frac{9Y_n}{X_n + 15Y_n + 3Z_n}.
\end{aligned} \tag{13.22}$$

The shape of the 3D volume of visible colors defined by these equations is of course different from that for CIE (X, Y, Z) space itself (Fig. 13.23).

With this background on color, we now turn our attention to color in computer graphics.

13.3 COLOR MODELS FOR RASTER GRAPHICS

A color model is a specification of a 3D color coordinate system and a visible subset in the coordinate system within which all colors in a particular color gamut lie. For instance, the RGB color model is the unit cube subset of the 3D Cartesian coordinate system.

The purpose of a color model is to allow convenient specification of colors within some color gamut. Our primary interest is the gamut for color CRT monitors, as defined by the RGB (red, green, blue) primaries in Color Plate II.2. As we see in this color plate, a color gamut is a subset of all visible chromaticities. Hence, a color model cannot be used to specify all visible colors. This is emphasized in Fig. 13.27, which shows that the gamut of a CRT monitor is a subset of (X, Y, Z) space.

Three hardware-oriented color models are RGB, used with color CRT monitors, YIQ, the broadcast TV color system, and CMY (cyan, magenta, yellow) for some color-printing devices. Unfortunately, none of these models are particularly easy to use, because they do not relate directly to intuitive color notions of hue, saturation, and brightness. Therefore,

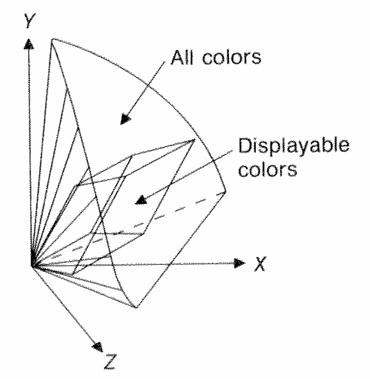

Fig. 13.27 The color gamut for a typical color monitor within the CIE color space. The range of colors that can be displayed on the monitor is clearly smaller than that of all colors visible in CIE space. (Courtesy of Gary Meyer, Program of Computer Graphics, Cornell University, 1978.)

another class of models has been developed with ease of use as a goal. Several such models are described in [GSPC79; JOBL78; MEYE80; SMIT78]. We discuss three, the HSV (sometimes called HSB), HLS, and HVC models.

With each model is given a means of converting to some other specification. For RGB, this conversion is to CIE's (X, Y, Z) space. This conversion is important, because CIE is the worldwide standard. For all of the other models, the conversion is to RGB; hence, we can convert from, say, HSV to RGB to the CIE standard.

13.3.1 The RGB Color Model

The red, green, and blue (RGB) color model used in color CRT monitors and color raster graphics employs a Cartesian coordinate system. The RGB primaries are *additive* primaries; that is, the individual contributions of each primary are added together to yield the result, as suggested in Color Plate II.3. The subset of interest is the unit cube shown in Fig. 13.28. The main diagonal of the cube, with equal amounts of each primary, represents the gray levels: black is (0, 0, 0); white is (1, 1, 1). Color Plates II.4 and II.5 show several views of the RGB color model.

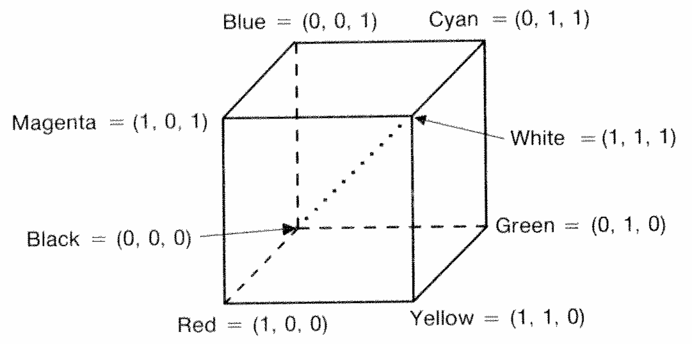

Fig. 13.28 The RGB cube. Grays are on the dotted main diagonal.

The color gamut covered by the RGB model is defined by the chromaticities of a CRT's phosphors. Two CRTs with different phosphors will cover different gamuts. To convert colors specified in the gamut of one CRT to the gamut of another CRT, we can use the transformations M_1 and M_2 from the RGB color space of each monitor to the (X, Y, Z) color space. The form of each transformation is:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} X_r & X_g & X_b \\ Y_r & Y_g & Y_b \\ Z_r & Z_g & Z_b \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}. \tag{13.23}$$

where X_r, X_g, and X_b are the weights applied to the monitor's RGB colors to find X, and so on.

Defining M as the 3×3 matrix of color-matching coefficients, we write Eq. (13.23) as

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = M \begin{bmatrix} R \\ G \\ B \end{bmatrix}. \tag{13.24}$$

With M_1 and M_2 the matricies that convert from each of the two monitor's gamuts to CIE, $M_2^{-1}M_1$ converts from the RGB of monitor 1 to the RGB of monitor 2. This matrix product is all that is needed if the color in question lies in the gamuts of both monitors. What if a color C_1 is in the gamut of monitor 1 but is not in the gamut of monitor 2? The corresponding color $C_2 = M_2^{-1}M_1C_1$ will be outside the unit cube and hence will not be displayable. A simple but not very satisfactory solution is to clamp the color values—that is, to replace values of R, G, or B that are less than 0 with 0, and values that are greater than 1 with 1. More satisfactory but also more complex approaches are described in [HALL89].

The chromaticity coordinates for the RGB phosphors are usually available from CRT manufacturers' specifications. If not, a colorimeter can also be used to measure the chromaticity coordinates directly, or a spectroradiometer can be used to measure $P(\lambda)$, which can then be converted to chromaticity coordinates using Eqs. (13.18), (13.19), and (13.20). Denoting the coordinates by (x_r, y_r) for red, (x_g, y_g) for green, and (x_b, y_b) for blue, and defining C_r as

$$C_r = X_r + Y_r + Z_r, \tag{13.25}$$

we can write, for the red primary;

$$x_r = \frac{X_r}{X_r + Y_r + Z_r} = \frac{X_r}{C_r}, \; X_r = x_r\, C_r,$$

$$y_r = \frac{Y_r}{X_r + Y_r + Z_r} = \frac{Y_r}{C_r}, \; Y_r = y_r\, C_r,$$

$$z_r = (1 - x_r - y_r) = \frac{Z_r}{X_r + Y_r + Z_r} = \frac{Z_r}{C_r}, \; Z_r = z_r\, C_r. \tag{13.26}$$

With similar definitions for C_g and C_b, Eq. (13.23) can be rewritten as:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} x_rC_r & x_gC_g & x_bC_b \\ y_rC_r & y_gC_g & y_bC_b \\ (1 - x_r - y_r)C_r & (1 - x_g - y_g)C_g & (1 - x_b - y_b)C_b \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}. \tag{13.27}$$

The unknowns C_r, C_g, and C_b can be found in one of two ways [MEYE83]. First, the luminances Y_r, Y_g, and Y_b of maximum-brightness red, green, and blue may be known or can be measured with a high-quality photometer (inexpensive meters can be off by factors of 2 to 10 on the blue reading). These measured luminances can be combined with the known y_r, y_g, and y_b to yield

$$C_r = Y_r/y_r, \; C_g = Y_g/y_g, \; C_b = Y_b/y_b. \tag{13.28}$$

These values are then substituted into Eq. (13.27), and the conversion matrix M is thus expressed in terms of the known quantities (x_r, y_r), (x_g, y_g), (x_b, y_b), Y_r, Y_g, and Y_b.

We can also remove the unknown variables from Eq. (13.27) if we know or can measure the X_w, Y_w, and Z_w for the white color produced when $R = G = B = 1$. For this case, Eq. (13.27) can be rewritten as

$$\begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} = \begin{bmatrix} x_r & x_g & x_b \\ y_r & y_g & y_b \\ (1 - x_r - y_r) & (1 - x_g - y_g) & (1 - x_b - y_b) \end{bmatrix} \begin{bmatrix} C_r \\ C_g \\ C_b \end{bmatrix}, \tag{13.29}$$

solved for C_r, C_g, and C_b (the only unknowns), and the resulting values substituted into Eq. (13.27). Alternatively, it may be that the white color is given by x_w, y_w, and Y_w; in this case, before solving Eq. (13.29), we first find

$$X_w = x_w \frac{Y_w}{y_w}, \; Z_w = z_w \frac{Y_w}{y_w}. \tag{13.30}$$

13.3.2 The CMY Color Model

Cyan, magenta, and yellow are the complements of red, green, and blue, respectively. When used as filters to subtract color from white light, they are called *subtractive primaries*. The subset of the Cartesian coordinate system for the CMY model is the same as that for RGB except that white (full light) instead of black (no light) is at the origin. Colors are specified by what is removed or subtracted from white light, rather than by what is added to blackness.

A knowledge of CMY is important when dealing with hardcopy devices that deposit colored pigments onto paper, such as electrostatic and ink-jet plotters. When a surface is coated with cyan ink, no red light is reflected from the surface. Cyan subtracts red from the reflected white light, which is itself the sum of red, green, and blue. Hence, in terms of the additive primaries, cyan is white minus red, that is, blue plus green. Similarly, magenta absorbs green, so it is red plus blue; yellow absorbs blue, so it is red plus green. A surface coated with cyan and yellow absorbs red and blue, leaving only green to be reflected from illuminating white light. A cyan, yellow, and magenta surface absorbs red, green, and blue, and therefore is black. These relations, diagrammed in Fig. 13.29, can be seen in Color Plate II.6 and are represented by the following equation:

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}. \tag{13.31}$$

The unit column vector is the RGB representation for white and the CMY representation for black.

The conversion from RGB to CMY is then

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} C \\ M \\ Y \end{bmatrix}. \tag{13.32}$$

These straightforward transformations can be used for converting the eight colors that can be achieved with binary combinations of red, green, and blue into the eight colors achievable with binary combinations of cyan, magenta, and yellow. This conversion is relevant for use on ink-jet and xerographic color printers.

Another color model, CMYK, uses black (abbreviated as K) as a fourth color. CMYK is used in the four-color printing process of printing presses and some hard-copy devices.

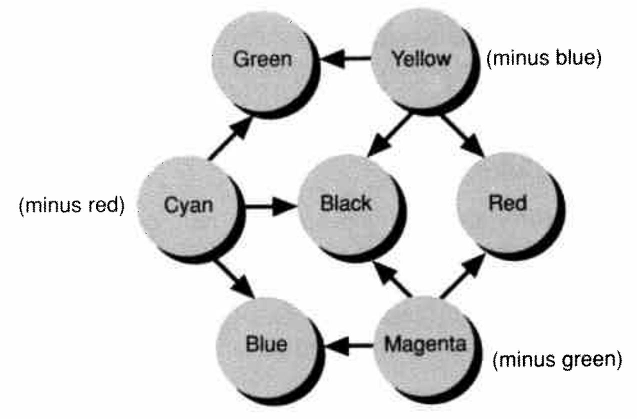

Fig. 13.29 Subtractive primaries (cyan, magenta, yellow) and their mixtures. For instance, cyan and yellow combine to green.

Given a CMY specification, black is used in place of equal amounts of C, M, and Y, according to the relations:

$$\begin{aligned} K &= \min(C, M, Y); \\ C &= C - K; \\ M &= M - K; \\ Y &= Y - K; \end{aligned} \tag{13.34}$$

This is discussed further in Section 13.4 and in [STON88].

13.3.3 The YIQ Color Model

The YIQ model is used in U.S. commercial color television broadcasting and is therefore closely related to color raster graphics. YIQ is a recoding of RGB for transmission efficiency and for downward compatibility with black-and-white television. The recoded signal is transmitted using the National Television System Committee (NTSC) [PRIT77] system.

The Y component of YIQ is not yellow but luminance, and is defined to be the same as the CIE **Y** primary. Only the Y component of a color TV signal is shown on black-and-white televisions: the chromaticity is encoded in I and Q. The YIQ model uses a 3D Cartesian coordinate system, with the visible subset being a convex polyhedron that maps into the RGB cube.

The RGB-to-YIQ mapping is defined as follows:

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.523 & 0.311 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}. \tag{13.33}$$

The quantities in the first row reflect the relative importance of green and red and the relative unimportance of blue in brightness. The inverse of the RGB-to-YIQ matrix is used for the YIQ-to-RGB conversion.

Equation (13.33) assumes that the RGB color specification is based on the standard NTSC RGB phosphor, whose CIE coordinates are

	Red	**Green**	**Blue**
x	0.67	0.21	0.14
y	0.33	0.71	0.08

and for which the white point, illuminant C, is $x_w = 0.31$, $y_w = 0.316$, and $Y_w = 100.0$.

Specifying colors with the YIQ model solves a potential problem with material being prepared for broadcast television: Two different colors shown side by side on a color monitor will appear to be different, but, when converted to YIQ and viewed on a monochrome monitor, they may appear to be the same. This problem can be avoided by specifying the two colors with different Y values in the YIQ color model space (i.e., by adjusting only the Y value to disambiguate them).

The YIQ model exploits two useful properties of our visual system. First, the system is more sensitive to changes in luminance than to changes in hue or saturation; that is, our ability to discriminate spatially color information is weaker than our ability to discriminate spatially monochrome information. This suggests that more bits of bandwidth should be used to represent Y than are used to represent I and Q, so as to provide higher resolution in Y. Second, objects that cover a very small part of our field of view produce a limited color sensation, which can be specified adequately with one rather than two color dimensions. This suggests that either I or Q can have a lower bandwidth than the other. The NTSC encoding of YIQ into a broadcast signal uses these properties to maximize the amount of information transmitted in a fixed bandwidth: 4 MHz is assigned to Y, 1.5 to I, and 0.6 to Q. Further discussion of YIQ can be found in [SMIT78; PRIT77].

13.3.4 The HSV Color Model

The RGB, CMY, and YIQ models are hardware-oriented. By contrast, Smith's HSV (hue, saturation, value) model [SMIT78] (also called the HSB model, with B for brightness) is user-oriented, being based on the intuitive appeal of the artist's tint, shade, and tone. The coordinate system is cylindrical, and the subset of the space within which the model is defined is a hexcone, or six-sided pyramid, as in Fig. 13.30. The top of the hexcone corresponds to $V = 1$, which contains the relatively bright colors. The colors of the $V = 1$ plane are *not* all of the same perceived brightness, however. Color Plates II.7 and II.8 show the color model.

Hue, or H, is measured by the angle around the vertical axis, with red at 0°, green at 120°, and so on (see Fig. 13.30). Complementary colors in the HSV hexcone are 180°

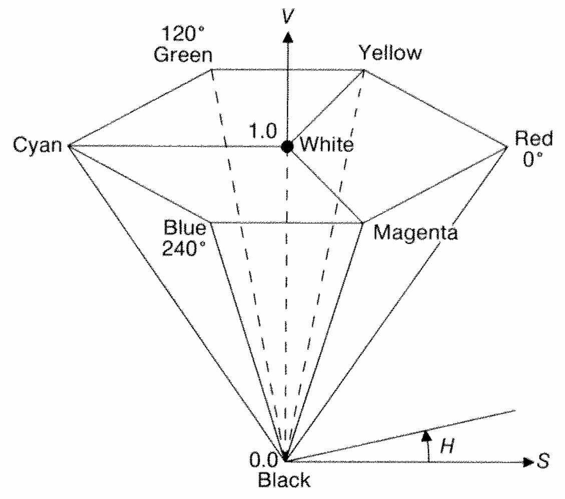

Fig. 13.30 Single-hexcone HSV color model. The $V = 1$ plane contains the RGB model's $R = 1$, $G = 1$, and $B = 1$ planes in the regions shown.

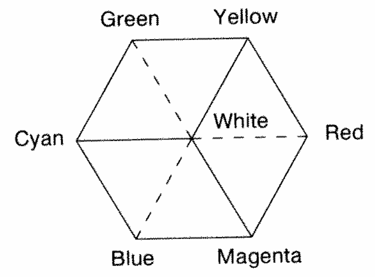

Fig. 13.31 RGB color cube viewed along the principal diagonal. Visible edges of the cube are solid; invisible edges are dashed.

opposite one another. The value of S is a ratio ranging from 0 on the center line (V axis) to 1 on the triangular sides of the hexcone. Saturation is measured relative to the color gamut represented by the model, which is, of course, a subset of the entire CIE chromaticity diagram. Therefore, saturation of 100 percent in the model is less than 100 percent excitation purity.

The hexcone is one unit high in V, with the apex at the origin. The point at the apex is black and has a V coordinate of 0. At this point, the values of H and S are irrelevant. The point $S = 0$, $V = 1$ is white. Intermediate values of V for $S = 0$ (on the center line) are the grays. When $S = 0$, the value of H is irrelevant (called by convention UNDEFINED). When S is not zero, H is relevant. For example, pure red is at $H = 0$, $S = 1$, $V = 1$. Indeed, any color with $V = 1$, $S = 1$ is akin to an artist's pure pigment used as the starting point in mixing colors. Adding white pigment corresponds to decreasing S (without changing V). Shades are created by keeping $S = 1$ and decreasing V. Tones are created by decreasing both S and V. Of course, changing H corresponds to selecting the pure pigment with which to start. Thus, H, S, and V correspond to concepts from the artists' color system, and are not exactly the same as the similar terms introduced in Section 13.2.

The top of the HSV hexcone corresponds to the projection seen by looking along the principal diagonal of the RGB color cube from white toward black, as shown in Fig. 13.31. The RGB cube has subcubes, as illustrated in Fig. 13.32. Each subcube, when viewed along

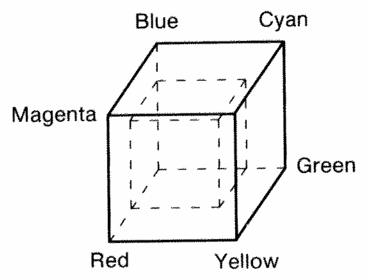

Fig. 13.32 RGB cube and a subcube.

```
void RGB_To_HSV (double r, double g, double b, double *h, double *s, double *v)
/* Given: r, g, b, each in [0,1]. */
/* Desired: h in [0,360), s and v in [0,1] except if s = 0, then h = UNDEFINED, */
/* which is some constant defined with a value outside the interval [0,360]. */
{
   double max = Maximum (r, g, b);
   double min = Minimum (r, g, b);
   *v = max;                            /* This is the value v. */
   /* Next calculate saturation, s. Saturation is 0 if red, green and blue are all 0 */
   *s = (max != 0.0) ? ((max - min) / max) : 0.0;
   if (*s == 0.0)
      *h = UNDEFINED;
   else {                               /* Chromatic case: Saturation is not 0, */
      double delta = max - min;         /* so determine hue. */
      if (r == max)
         *h = (g - b) / delta;          /* Resulting color is between yellow and magenta */
      else if (g == max)
         *h = 2.0 + (b - r) / delta;    /* Resulting color is between cyan and yellow */
      else if (b == max)
         *h = 4.0 + (r - g) / delta;    /* Resulting color is between magenta and cyan */
      *h *= 60.0;                       /* Convert hue to degrees */
      if (*h < 0.0)
         *h += 360.0;                   /* Make sure hue is nonnegative */
   } /* Chromatic case */
} /* RGB_To_HSV */
```

Fig. 13.33 Algorithm for converting from RGB to HSV color space.

its main diagonal, is like the hexagon in Fig. 13.31, except smaller. Each plane of constant V in HSV space corresponds to such a view of a subcube in RGB space. The main diagonal of RGB space becomes the V axis of HSV space. Thus, we can see intuitively the correspondence between RGB and HSV. The algorithms of Figs. 13.33 and 13.34 define the correspondence precisely by providing conversions from one model to the other.

13.3.5 The HLS Color Model

The HLS (hue, lightness, saturation) color model is defined in the double-hexcone subset of a cylindrical space, as seen in Fig. 13.35. Hue is the angle around the vertical axis of the double hexcone, with red at 0° (some discussions of HLS have blue at 0°; we place red at 0° for consistency with the HSV model). The colors occur around the perimeter in the same order as in the CIE diagram when its boundary is traversed counterclockwise: red, yellow, green, cyan, blue, and magenta. This is also the same order as in the HSV single-hexcone model. In fact, we can think of HLS as a deformation of HSV, in which white is pulled

```
void HSV_To_RGB (double *r, double *g, double *b, double h, double s, double v)
/* Given: h in [0,360] or UNDEFINED, s and v in [0,1]. */
/* Desired: r, g, b, each in [0,1]. */
{
   if (s == 0.0) {          /* The color is on the black-and-white center line. */
      /* Achromatic color: There is no hue. */
      if (h == UNDEFINED) {
         *r = v;            /* This is the achromatic case. */
         *g = v;
         *b = v;
      } else
         Error();           /* By our convention, error if s = 0 and h has a value. */
   } else {
      double f,p,q,t;       /* Chromatic color: s != 0, so there is a hue. */
      int i;

      if (h == 360.0)       /* 360 degrees is equivalent to 0 degrees. */
         h = 0.0;
      h /= 60.0;            /* h is now in [0,6). */
      i = floor (h);        /* Floor returns the largest integer <= h */
      f = h - i;            /* f is the fractional part of h. */
      p = v * (1.0 - s);
      q = v * (1.0 - (s * f));
      t = v * (1.0 - (s * (1.0 - f)));
      switch(i) {
         case 0: *r = v;  *g = t;  *b = p; break;
         case 1: *r = q;  *g = v;  *b = p; break;
         case 2: *r = p;  *g = v;  *b = t; break;
         case 3: *r = p;  *g = q;  *b = v; break;
         case 4: *r = t;  *g = p;  *b = v; break;
         case 5: *r = v;  *g = p;  *b = q; break;
      }
   } /* Chromatic case */
} /* HSV_To_RGB */
```

Fig. 13.34 Algorithm for converting from HSV to RGB color space.

upward to form the upper hexcone from the $V = 1$ plane. As with the single-hexcone model, the complement of any hue is located 180° farther around the double hexcone, and saturation is measured radially from the vertical axis, from 0 on the axis to 1 on the surface. Lightness is 0 for black (at the lower tip of the double hexcone) to 1 for white (at the upper tip). Color Plate II.9 shows a view of the HLS model. Again, the terms hue, lightness, and saturation in this model are similar to, but are not exactly identical to, the same terms as they were introduced in an earlier section. Color Plate II.10 shows a different view of the space.

The procedures of Figs. 13.36 and 13.37 perform the conversions between HLS and RGB. They are modified from those given by Metrick [GSPC79] to leave H UNDEFINED when $S = 0$, and to have $H = 0$ for red rather than for blue.

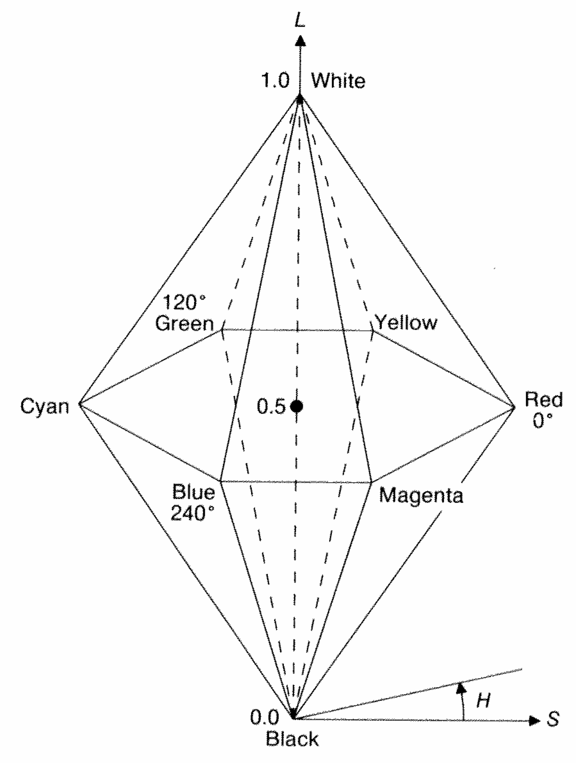

Fig. 13.35 Double-hexcone HLS color model.

The HLS model, like the HSV model, is easy to use. The grays all have $S = 0$, but the maximally saturated hues are at $S = 1$, $L = 0.5$. If potentiometers are used to specify the color-model parameters, the fact that L must be 0.5 to get the strongest possible colors is a disadvantage over the HSV model, in which $S = 1$ and $V = 1$ achieve the same effect. However, analogously to HSV, the colors of the $L = 0.5$ plane are *not* all of the same perceived brightness. Hence two different colors of equal perceived brightness will generally have different values of L. Furthermore, neither HLS nor any of the other models discussed thus far in this section are perceptually uniform, in the sense discussed in Section 13.2.

The recently developed Tektronix TekHVC (hue, value, chroma) color system, a modification of the CIE LUV perceptually uniform color space discussed in Section 13.2, does provide a color space in which measured and perceived distances between colors are approximately equal. This is an important advantage of both the CIE LUV and TekHVC models. Figure 13.38 shows one view of the HVC color model, and Color Plate II.11 shows another view. Details of the transformations from CIE to TekHVC have not been released. However, we see from Eq. (13.22) that the transformation from CIE XYZ to CIE LUV is computationally straightforward, with the cube root being the most computationally intense element. Thus, we expect that perceptually uniform color spaces will come to be more widely used in the future.

```
void RGB_To_HLS (double r, double g, double b, double *h, double *l, double *s)
/* Given: r, g, b each in [0,1]. */
/* Desired: h in [0,360), l and s in [0,1], except if s = 0, then h = UNDEFINED. */
{
  double max = Maximum (r, g, b);
  double min = Minimum (r, g, b);
  *l = (max + min) / 2.0;               /* This is the lightness. */
  /* Next calculate saturation */
  if (max == min) {                     /* Achromatic case, because r = g = b */
     *s = 0;
     *h = UNDEFINED;
  } else {                              /* Chromatic case */
     double delta = max - min;

     /* First calculate the saturation. */
     *s = (*l <= 0.5) ? (delta / (max + min)) : (delta / (2.0 - (max + min)));
     /* Next calculate the hue. */
     if (r == max)
        *h = (g - b) / delta;           /* Resulting color is between yellow and magenta */
     else if (g == max)
        *h = 2.0 + (b - r) / delta;     /* Resulting color is between cyan and yellow */
     else if (b == max)
        *h = 4.0 + (r - g) / delta;     /* Resulting color is between magenta and cyan */
     *h *= 60.0;                        /* Convert to degrees */
     if (h < 0.0)
        *h += 360.0;                    /* Make degrees be nonnegative */
  }  /* Chromatic case */
}  /* RGB_To_HLS */
```

Fig. 13.36 Algorithm for converting from RGB to HLS color space.

13.3.6 Interactive Specification of Color

Many application programs allow the user to specify colors of areas, lines, text, and so on. If only a small set of colors is provided, menu selection from samples of the available colors is appropriate. But what if the set of colors is larger than can reasonably be displayed in a menu?

The basic choices are to use English-language names, to specify the numeric coordinates of the color in a color space (either by typing or with slider dials), or to interact directly with a visual representation of the color space. Naming is in general unsatisfactory because it is ambiguous and subjective (''a light navy blue with a touch of green''), and it is also the antithesis of graphic interaction. On the other hand, [BERK82] describes CNS, a fairly well-defined color-naming scheme that uses terms such as ''greenish-yellow,'' ''green-yellow,'' and ''yellowish-green'' to distinguish three hues between green and

```
void HLS_To_RGB (double *r, double *g, double *b, double h, double l, double s)
/* Given: h in [0,360] or UNDEFINED, l and s in [0,1]. */
/* Desired: r, g, b each in [0,1] */
{
    double m1, m2;

    m2 = (l <= 0.5) ? (l * (l + s)) : (l + s - l * s);
    m1 = 2.0 * l - m2;
    if (s == 0.0) {                  /* Achromatic: There is no hue. */
        if (h == UNDEFINED)
            *r = *g = *b = l;        /* This is the achromatic case. */
        else Error ();               /* Error if s = 0 and h has a value */
    } else {        /* Chromatic case, so there is a hue */
        *r = Value (m1, m2, h + 120.0);
        *g = Value (m1, m2, h);
        *b = Value (m1, m2, h - 120.0);
    }
}   /* HLS_To_RGB */

static double Value (double n1, double n2, double hue)
{
    if (hue > 360.0)
        hue -= 360.0;
    else if (hue < 0.0)
        hue += 360.0;
    if (hue < 60.0)
        return n1 + (n2 - n1) * hue / 60.0;
    else if (hue < 180.0)
        return n2;
    else if (hue < 240.0)
        return n1 + (n2 - n1) * (240.0 - hue) / 60.0;
    else
        return n1;
}   /* Value */
```

Fig. 13.37 Algorithm for converting from HLS to RGB color space.

yellow. In an experiment, users of CNS were able to specify colors more precisely than were users who entered numeric coordinates in either RGB or HSV space.

Coordinate specification can be done with slider dials, as in Fig. 13.39, using any of the color models. If the user understands how each dimension affects the color, this technique works well. Probably the best interactive specification method is to let the user interact directly with a representation of the color space, as shown in Fig. 13.40. The line on the circle (representing the $V = 1$ plane) can be dragged around to determine which slice of the HSV volume is displayed in the triangle. The cursor on the triangle can be moved around to specify saturation and value. As the line or the cursor is moved, the numeric readouts change value. When the user types new values directly into the numeric readouts, the line and cursor are repositioned. The color sample box shows the currently selected

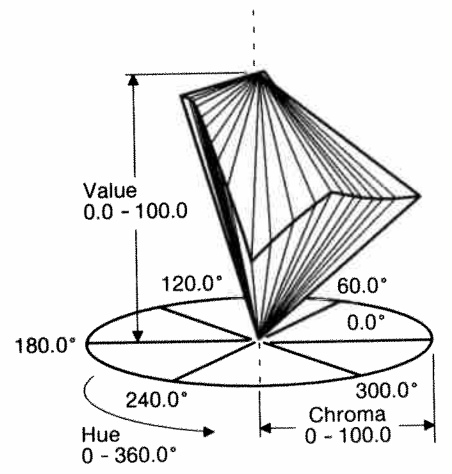

Fig. 13.38 The TekHVC color model. (Courtesy of Tektronix, Inc.)

color. Color Plate II.12 shows a similar method used for the HSV model.

[SCHW87] describes color-matching experiments in which subjects used a data tablet to specify colors in several models including RGB, YIQ, LAB [WYSZ82], and HSV. HSV was found to be slow but accurate, whereas RGB was faster but less accurate. There is a widespread belief that the HSV model is especially tractable, usually making it the model of choice.

Many interactive systems that permit user specification of color show the user the current color settings as a series of adjacent patches of color, as in part (a) of Fig. 13.39, or as the color of the single pixel value currently being set, as in part (b). As the user

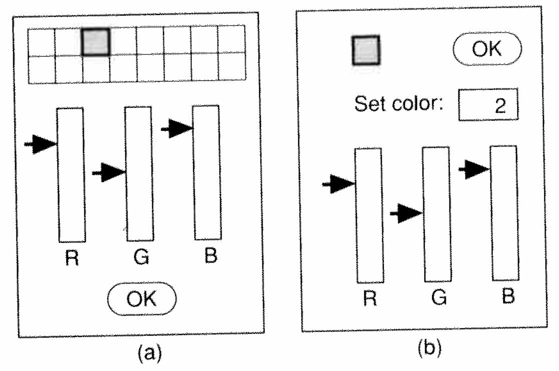

Fig. 13.39 Two common ways of setting colors. In (a), the user selects one of 16 colors to set; the selected color is designated with the thick border. The RGB slider dials control the color; OK is selected to dismiss the color control panel. In (b), the number of the color to be set is typed, and the current color is displayed in the gray-toned box. In both cases, the user must simultaneously see the actual display in order to understand the effects of the color change.

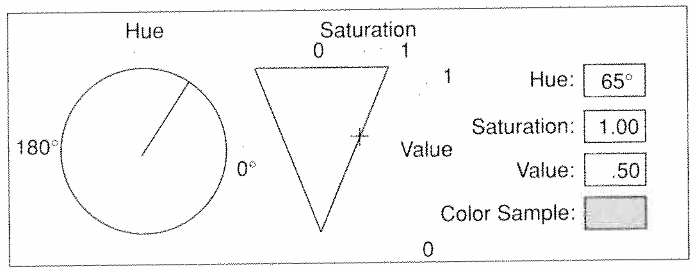

Fig. 13.40 A convenient way to specify colors in HSV space. Saturation and value are shown by the cursor in the triangular area, and hue by the line in the circular area. The user can move the line and cursor indicators on the diagrams, causing the numeric readouts to be updated. Alternatively, the user can type new values, causing the indicators to change. Slider dials for *H*, *S*, and *V* could also be added, giving the user accurate control over a single dimension at a time, without the need to type values.

manipulates the slider dials, the color sample changes. However, a person's perception of color is affected by surrounding colors and the sizes of colored areas, as shown in Color Plate II.13; hence, the color as perceived in the feedback area will probably differ from the color as perceived in the actual display. It is thus important that the user also see the actual display while the colors are being set.

13.3.7 Interpolating in Color Space

Color interpolation is necessary in at least three situations: for Gouraud shading (Section 16.2.4), for antialiasing (Section 3.17), and in blending two images together as for a fade-in, fade-out sequence. The results of the interpolation depend on the color model in which the colors are interpolated; thus, care must be taken to select an appropriate model.

If the conversion from one color model to another transforms a straight line (representing the interpolation path) in one color model into a straight line in the other color model, then the results of linear interpolation in both models will be the same. This is the case for the RGB, CMY, YIQ, and CIE color models, all of which are related by simple affine transformations. However, a straight line in the RGB model does *not* in general transform into a straight line in either the HSV or HLS models. Color Plate II.14 shows the results of linearly interpolating between the same two colors in the HSV, HSL, RGB, and YIQ color spaces. Consider the interpolation between red and green. In RGB, red = (1, 0, 0) and green = (0, 1, 0). Their interpolation (with both weights equal to 0.5 for convenience) is (0.5, 0.5, 0). Applying algorithm RGB_To_HSV (Fig. 13.33) to this result, we have (60°, 1, 0.5). Now, representing red and green in HSV, we have (0°, 1, 1) and (120°, 1, 1). But interpolating with equal weights in HSV, we have (60°, 1, 1); thus, the value differs by 0.5 from the same interpolation in RGB.

As a second example, consider interpolating red and cyan in the RGB and HSV models. In RGB, we start with (1, 0, 0) and (0, 1, 1), respectively, and interpolate to (0.5, 0.5, 0.5), which in HSV is represented as (UNDEFINED, 0, 0.5). In HSV, red and cyan are (0°, 1, 1) and (180°, 1, 1). Interpolating, we have (90°, 1, 1); a new hue at maximum value and saturation has been introduced, whereas the "right" result of combining equal amounts

of complementary colors is a gray value. Here, again, interpolating and then transforming gives different results from transforming and then interpolating.

For Gouraud shading, any of the models can be used, because the two interpolants are generally so close together that the interpolation paths between the colors are close together as well. When two images are blended—as in a fade-in, fade-out sequence or for antialiasing—the colors may be quite distant, and an additive model, such as RGB, is appropriate. If, on the other hand, the objective is to interpolate between two colors of fixed hue (or saturation) and to maintain the fixed hue (or saturation) for all interpolated colors, then HSV or HLS is preferable. But note that a fixed-saturation interpolation in HSV or HLS is *not* seen as having exactly fixed saturation by the viewer [WARE87].

13.4 REPRODUCING COLOR

Color images are reproduced in print in a way similar to that used for monochrome images, but four sets of halftone dots are printed, one for each of the subtractive primaries, and another for black. In a process called *undercolor removal*, black replaces equal amounts of cyan, magenta, and yellow. This creates a darker black than is possible by mixing the three primaries, and hastens drying by decreasing the amounts of cyan, magenta, and yellow ink needed. The orientation of each of the grids of dots is different, so that interference patterns are not created. Color Plate II.15 shows an enlarged halftone color pattern. Our eyes spatially integrate the light reflected from adjacent dots, so that we see the color defined by the proportions of primaries in adjacent dots. This spatial integration of different colors is the same phenomenon we experience when viewing the triads of red, green, and blue dots on a color monitor.

We infer, then, that color reproduction in print and on CRTs depends on the same spatial integration used in monochrome reproduction. The monochrome dithering techniques discussed in Section 13.1.2 can also be used with color to extend the number of available colors, again at the expense of resolution. Consider a color display with 3 bits per pixel, one each for red, green, and blue. We can use a 2×2 pixel pattern area to obtain 125 different colors: each pattern can display five intensities for each of red, green, and blue, by using the halftone patterns in Fig. 13.8. This results in $5 \times 5 \times 5 = 125$ color combinations.

Not all color reproduction depends exclusively on spatial integration. For instance, xerographic color copiers, ink-jet plotters, and thermal color printers actually mix subtractive pigments on the paper's surface to obtain a small set of different colors. In xerography, the colored pigments are first deposited in three successive steps, then are heated and melted together. The inks sprayed by the plotter mix before drying. Spatial integration may be used to expand the color range further.

A related quantization problem occurs when a color image with n bits per pixel is to be displayed on a display of $m < n$ bits per pixel with no loss of spatial resolution. Here, color resolution *must* be sacrificed. In this situation, there are two key questions: Which 2^m colors should be displayed? What is the mapping from the set of 2^n colors in the image to the smaller set of 2^m colors being displayed?

The simple answers are to use a predefined set of display colors and a fixed mapping from image colors to display colors. For instance, with $m = 8$, a typical assignment is 3 bits to red and green and 2 to blue (because of the eye's lower sensitivity to blue). Thus, the 256 displayable colors are all the combinations of eight reds, eight greens, and four blues. The specific values for the red, green, and blue colors would be spaced across the range of 0.0 to 1.0 range on a ratio scale, as discussed in Section 13.1.1. For an image with 6 bits per color and hence 64 levels for each color, the 64 red colors are mapped into one of the eight displayable reds, and similarly for the greens. The 64 blue colors are mapped into just four blues.

With this solution, however, if all the blue image colors are clustered together in color space, they might all be displayed as the same blue, whereas the other three displayable blues might go unused. An adaptive scheme would take this possibility into account and would divide up the blue-value range on the basis of the distribution of values in the range. Heckbert [HECK82] describes two approaches of this type: the popularity algorithm and the median-cut algorithm.

The *popularity algorithm* creates a histogram of the image's colors, and uses the 2^m most frequent colors in the mapping. The *median-cut algorithm* recursively fits a box around the colors used in the image, splitting the box along its longer dimension at the median in that dimension. The recursion ends when 2^m boxes have been created; the centroid of each box is used as the display color for all image colors in the box. The median-cut algorithm is slower than is the popularity algorithm, but produces better results. Another way of splitting the boxes is described in [WAN88].

Creating an *accurate* color reproduction is much more difficult than is approximating colors. Two display monitors can be calibrated to create the same tristimulus values; the multistep process is described in detail in [COWA83; MEYE83]. The steps are to measure the chromaticity coordinates of the monitor phosphors, then to adjust the brightness and contrast controls of each of the monitor guns so that the same white chromaticity is produced whenever $R = G = B$, and to determine the appropriate gamma correction for each gun.

Making slides or movie film that look exactly like the image on a display is difficult, because many variables are involved. They include the gamma correction of the display and of the CRT used in the film recorder; the color of light emitted by the CRT in the film recorder; the filters used in the film recorder; the type of film used; the quality and temperature of the developing chemicals, the length of time the film is in the chemicals, and the color of light emitted by the bulb in the slide or film projector. Fortunately, all these variables can be quantified and controlled, albeit with considerable difficulty.

Controlling the color match on printed materials is also difficult; the printing process, with its cyan, magenta, yellow, and black primaries, requires careful quality control to maintain registration and ink flow. The paper texture, absorbancy, and gloss also affect the result. Complicating matters further, the simple subtractive CMY color model discussed in Section 13.3.2 cannot be used directly, because it does not take into account these complications of the printing process. More detail can be found in [STON88].

Even if extreme care is taken in color reproduction, the results may not seem to match the original. Lighting conditions and reflections from the display can cause colors with the same measured chromaticity coordinates to appear to be different. Fortunately, the purpose of the reproduction is usually (although not always) to maintain color relationships between different parts of the image, rather than to make an exact copy.

13.5 USING COLOR IN COMPUTER GRAPHICS

We use color for aesthetics, to establish a tone or mood, for realism, as a highlight, to identify associated areas as being associated, and for coding. With care, color can be used effectively for these purposes. In addition, users tend to like color, even when there is no quantitative evidence that it helps their performance. Although cost-conscious buyers may scoff at color monitors, we believe that anything that encourages people to use computers is important!

Careless use of color can make the display less useful or less attractive than a corresponding monochrome presentation. In one experiment, introduction of meaningless color reduced user performance to about one-third of what it was without color [KREB79]. Color should be employed conservatively. Any decorative use of color should be subservient to the functional use, so that the color cannot be misinterpreted as having some underlying meaning. Thus, the use of color, like all other aspects of a user–computer interface, must be tested with real users to identify and remedy problems. Of course, some individuals may have other preferences, so it is common practice to provide defaults chosen on the basis of color usage rules, with some means for the user to change the defaults. A conservative approach to color selection is to design first for a monochrome display, to ensure that color use is purely redundant. This avoids creating problems for color-deficient users and also means that the application can be used on a monochrome display. Additional information on color deficiencies is given in [MEYE88]. The color choices used in the window managers shown in Color Plates I.26 through I.29 are quite conservative. Color is not used as a unique code for button status, selected menu item, and so forth.

Many books have been written on the use of color for aesthetic purposes, including [BIRR61]; we state here just a few of the simpler rules that help to produce color harmony. The most fundamental rule of color aesthetics is to select colors according to some method, typically by traversing a smooth path in a color model or by restricting the colors to planes or hexcones in a color space. This might mean using colors of constant lightness or value. Furthermore, colors are best spaced at equal *perceptual* distances (this is not the same as being at equally spaced increments of a coordinate, and can be difficult to implement). Recall too that linear interpolation (as in Gouraud shading) between two colors produces different results in different color spaces (see Exercise 13.10 and Color Plate II.14).

A random selection of different hues and saturations is usually quite garish. Alvy Ray Smith performed an informal experiment in which a 16 × 16 grid was filled with randomly generated colors. Not unexpectedly, the grid was unattractive. Sorting the 256 colors according to their H, S, and V values and redisplaying them on the grid in their new order improved the appearance of the grid remarkably.

More specific instances of these rules suggest that, if a chart contains just a few colors, the complement of one of the colors should be used as the background. A neutral (gray) background should be used for an image containing many different colors, since it is both harmonious and inconspicuous. If two adjoining colors are not particularly harmonious, a thin black border can be used to set them apart. This use of borders is also more effective for the achromatic (black/white) visual channel, since shape detection is facilitated by the black outline. Some of these rules are encoded in ACE (A Color Expert), an expert system for selecting user-interface colors [MEIE88]. In general, it is good to minimize the number of different colors being used (except for shading of realistic images).

Color can be used for coding, as discussed in Chapter 9 and illustrated by Color Plate II.16. However, several cautions are in order. First, color codes can easily carry unintended meanings. Displaying the earnings of company A as red and those of company B as green might well suggest that company A is in financial trouble, because of our learned associations of colors with meanings. Bright, saturated colors stand out more strongly than do dimmer, paler colors, and may give unintended emphasis. Two elements of a display that have the same color may be seen as related by the same color code, even if they are not.

This problem often arises when color is used both to group menu items and to distinguish display elements, such as different layers of a printed circuit board or VLSI chip; for example, green display elements tend to be associated with menu items of the same color. This is one of the reasons that use of color in user-interface elements, such as menus, dialogue boxes, and window borders, should be restrained. (Another reason is to leave as many colors as possible free for the application program itself.)

A number of color usage rules are based on physiological rather than aesthetic considerations. For example, because the eye is more sensitive to spatial variation in intensity than it is to variation in chromaticity, lines, text, and other fine detail should vary from the background not just in chromaticity, but in brightness (perceived intensity) as well—especially for colors containing blue, since relatively few cones are sensitive to blue. Thus, the edge between two equal-brightness colored areas that differ only in the amount of blue will be fuzzy. On the other hand, blue-sensitive cones spread out farther on the retina than do red- and green-sensitive ones, so our peripheral color vision is better for blue (this is why many police-car flashers are now blue instead of red).

Blue and black differ very little in brightness, and are thus a particularly bad combination. Similarly, yellow on white is relatively hard to distinguish, because both colors are quite bright (see Exercise 13.11). Color plates I.28 and I.29 show a very effective use of yellow to highlight black text on a white background. The yellow contrasts very well with the black text and also stands out. In addition, the yellow highlight is not as overpowering as a black highlight with reversed text (that is, with the highlighted text in white on a black highlight), as is common on monochrome displays.

White text on a blue background provides a good contrast that is less harsh than white on black. It is good to avoid reds and greens with low saturation and luminance, as these are the colors confused by those of us who are red–green color blind, the most common form of color-perception deficiency. Meyer and Greenberg describe effective ways to choose colors for color-blind viewers [MEYE88].

The eye cannot distinguish the color of very small objects, as already remarked in connection with the YIQ NTSC color model, so color coding should not be applied to small objects. In particular, judging the color of objects subtending less than 20 to 40 minutes of arc is error-prone [BISH60, HAEU76]. An object 0.1 inches high, viewed from 24 inches (a typical viewing distance) subtends this much arc, which corresponds to about 7 pixels of height on a 1024-line display with a vertical height of 15 inches. It is clear that the color of a single pixel is quite difficult to discern (see Exercise 13.18).

The perceived color of a colored area is affected by the color of the surrounding area, as is very evident in Color Plate II.13; this effect is particularly problematic if colors are used to encode information. The effect is minimized when the surrounding areas are some shade of gray or are relatively unsaturated colors.

The color of an area can actually affect its perceived size. Cleveland and McGill discovered that a red square is perceived as larger than is a green square of equal size [CLEV83]. This effect could well cause the viewer to attach more importance to the red square than to the green ones.

If a user stares at a large area of highly saturated color for several seconds and then looks elsewhere, an afterimage of the large area will appear. This effect is disconcerting, and causes eye strain. Use of large areas of saturated colors is hence unwise. Also, large areas of different colors can appear to be at different distances from the viewer, because the index of refraction of light depends on wavelength. The eye changes its focus as the viewer's gaze moves from one colored area to another, and this change in focus gives the impression of differing depths. Red and blue, which are at opposite ends of the spectrum, have the strongest depth-disparity effect, with red appearing closer and blue more distant. Hence, simultaneously using blue for foreground objects and red for the background is unwise; the converse is fine.

With all these perils and pitfalls of color usage, is it surprising that one of our first-stated rules was to apply color conservatively?

13.6 SUMMARY

The importance of color in computer graphics will continue to increase as color monitors and color hardcopy devices become the norm in many applications. In this chapter, we have introduced those color concepts most relevant to computer graphics; for more information, see the vast literature on color, such as [BILL81; BOYN79; GREG66; HUNT87; JUDD75; WYSZ82]. More background on artistic and aesthetic issues in the use of color in computer graphics can be found in [FROM84; MARC82; MEIE88; MURC85]. The difficult problems of precisely calibrating monitors and matching the colors appearing on monitors with printed colors are discussed in [COWA83; STON88].

EXERCISES

13.1 Derive an equation for the number of intensities that can be represented by $m \times m$ pixel patterns, where each pixel has w bits.

13.2 Write the programs needed to gamma-correct a black-and-white display through a look-up table. Input parameters are γ, I_0, m, the number of intensities desired, and K, the constant in Eq. (13.5).

13.3 Write an algorithm to display a pixel array on a bilevel output device. The inputs to the algorithm are an $m \times m$ array of pixel intensities, with w bits per pixel, and an $n \times n$ growth sequence matrix. Assume that the output device has resolution of $m \cdot n \times m \cdot n$.

13.4 Repeat Exercise 13.3 by using ordered dither. Now the output device has resolution $m \times m$, the same as the input array of pixel intensities.

13.5 Write an algorithm to display a filled polygon on a bilevel device by using an $n \times n$ filling pattern.

13.6 When certain patterns are used to fill a polygon being displayed on an interlaced raster display, all of the "on" bits fall on either the odd or the even scan lines, introducing a slight amount of flicker. Revise the algorithm from Exercise 13.5 to permute rows of the $n \times n$ pattern so that alternate

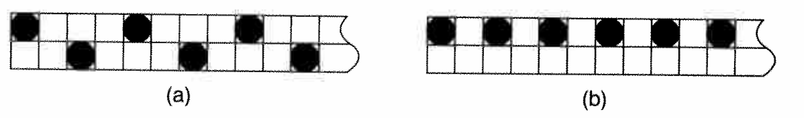

Fig. 13.41 Results obtained by using intensity level 1 from Fig. 13.8 in two ways: (a) with alternation (intensified pixels are on both scan lines), and (b) without alternation (all intensified pixels are on the same scan line).

replications of the pattern will alternate use of the odd and even scan lines. Figure 13.41 shows the results obtained by using intensity level 1 from Fig. 13.8, with and without this alternation.

13.7 Given a spectral energy distribution, how would you find the dominant wavelength, excitation purity, and luminance of the color it represents?

13.8 Plot the locus of points of the constant luminance values 0.25, 0.50, and 0.75, defined by $Y = 0.30R + 0.59G + 0.11B$, on the RGB cube, the HLS double hexcone, and the HSV hexcone.

13.9 Why are the opposite ends of the spectrum in the CIE diagram connected by a *straight* line?

13.10 Express, in terms of R, G, and B: the I of YIQ, the V of HSV, and the L of HSL. Note that I, V, and L are not the same.

13.11 Calculate in YIQ color space the luminances of the additive and subtractive primaries. Rank the primaries by luminance. This ranking gives their relative intensities, both as displayed on a black-and-white television and as perceived by our eyes.

13.12 Discuss the design of a raster display that uses HSV or HLS, instead of RGB, as its color specification.

13.13 In which color models are the rules of color harmony most easily applied?

13.14 Verify that Eq. (13.27) can be rewritten as Eq. (13.29) when $R = G = B = 1$.

13.15 If the white color used to calculate C_r, C_g, and C_b in Eq. (13.29) is given by x_w, y_w, and Y_w rather than by X_w, Y_w, and Z_w, what are the algebraic expressions for C_r, C_g, and C_b?

13.16 Rewrite the HSV-to-RGB conversion algorithm to make it more efficient. Replace the assignment statements for p, q, and t with: $vs = v * s$; $vsf = vs * f$; $p = v - vs$; $q = v - vsf$; $t = p + vsf$. Also assume that R, G, and B are in the interval [0, 255], and see how many of the computations can be converted to integer.

13.17 Write a program that displays, side by side, two 16×16 grids. Fill each grid with colors. The left grid will have 256 colors randomly selected from HSV color space (created by using a random-number generator to choose one out of 10 equally spaced values for each of H, S, and V). The right grid contains the same 256 colors, sorted on H, S, and V. Experiment with the results obtained by varying which of H, S, and V is used as the primary sort key.

13.18 Write a program to display on a gray background small squares colored orange, red, green, blue, cyan, magenta, and yellow. Each square is separated from the others and is of size $n \times n$ pixels, where n is an input variable. How large must n be so that the colors of each square can be unambiguously judged from distances of 24 and of 48 inches? What should be the relation between the two values of n? What effect, if any, do different background colors have on this result?

13.19 Calculate the number of bits of look-up-table accuracy needed to store 256 different intensity levels given dynamic intensity ranges of 50, 100, and 200.

13.20 Write a program to interpolate linearly between two colors in RGB, HSV, and HSL. Accept the two colors as input, allowing them to be specified in any of these three models.

14 The Quest for Visual Realism

In previous chapters, we discussed graphics techniques involving simple 2D and 3D primitives. The pictures that we produced, such as the wireframe houses of Chapter 6, represent objects that in real life are significantly more complex in both structure and appearance. In this chapter, we introduce an increasingly important application of computer graphics: creating realistic images of 3D scenes.

What is a *realistic* image? In what sense a picture, whether painted, photographed, or computer-generated, can be said to be "realistic" is a subject of much scholarly debate [HAGE86]. We use the term rather broadly to refer to a picture that captures many of the effects of light interacting with real physical objects. Thus, we treat realistic images as a continuum and speak freely of pictures, and of the techniques used to create them, as being "more" or "less" realistic. At one end of the continuum are examples of what is often called *photographic realism* (or *photorealism*). These pictures attempt to synthesize the field of light intensities that would be focused on the film plane of a camera aimed at the objects depicted. As we approach the other end of the continuum, we find images that provide successively fewer of the visual cues we shall discuss.

You should bear in mind that a more realistic picture is not necessarily a more desirable or useful one. If the ultimate goal of a picture is to convey information, then a picture that is free of the complications of shadows and reflections may well be more successful than a *tour de force* of photographic realism. In addition, in many applications of the techniques outlined in the following chapters, reality is intentionally altered for aesthetic effect or to fulfill a naive viewer's expectations. This is done for the same reasons that science-fiction films feature the sounds of weapon blasts in outer space—an impossibility in a vacuum. For example, in depicting Uranus in Color Plate II.20, Blinn shined an extra light on the