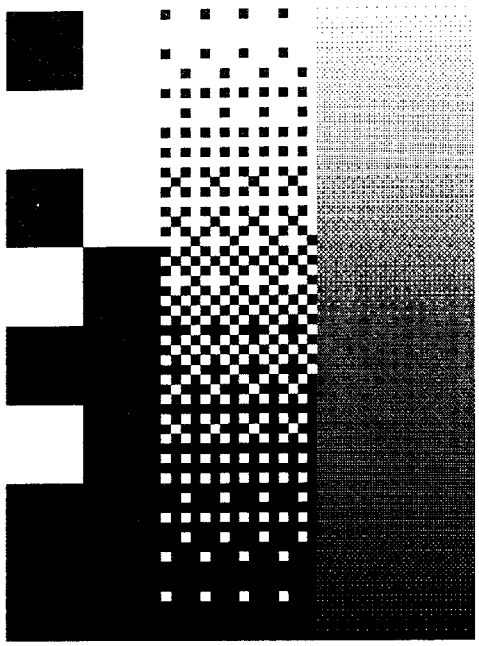
6



Colors and Gray Scales

Color is not one-dimensional or two-dimensional; it is three-dimensional. One way of visualizing the three dimensions is to separate color into *hue, saturation,* and *brightness.* Hue and saturation are the chromatic attributes of color, whereas brightness (similar to *lightness, value, intensity,* or *luminosity*) is achromatic. Hue is the spectral part of color; that is, hue determines whether the color is similar to red, yellow, green, blue, etc. Saturation determines the purity of the color. Colors of zero saturation are shades of gray, varying from black to white as brightness increases from zero to its maximum value.

The illustration at the top of this page shows how gray scales may be produced by ordered *dithering.* On the left side of the illustration, 4 nonwhite shades of gray are produced as black squares and are filled in a 2 × 2 matrix. In the middle part of the illustration, 16 nonwhite shades are produced in a 4 × 4 matrix. On the right, 64 shades result from an 8 × 8 matrix. Dithering is also useful for expanding the number of colors. For example, if your color graphics adapter is set to a mode that produces only 16 colors, then your rendering (on AutoCAD, AutoShade, AutoVision, or other software) will expand the number of colors to 256 by dithering a number of different hues.

Color is a complex subject because it involves human perception in addition to the physical characteristics of light. A given color does not require a unique formation; it can be constructed in a variety of ways by mixing three or more other colors. To quantify color mixing, the Commission Internationale de l'Eclairage (CIE) developed a chromaticity diagram in 1931. The CIE diagram was based on a large number of experiments in which subjects determined the amounts of three standard colors that combined to match any given color. New color-application tools relate color models for computers to color-perception models (CIE diagram) and to color characteristics of monitors and printers.

6.1 COLOR MODELS

The color models discussed in this section were developed for convenient visualization of the three-dimensional character of color. These models can be readily implemented into computer graphics systems. Transformations relate the three color variables of one system to the variables of any other system. Before the first color models (RGB and CMY color cubes) are considered, the concepts of additive and subtractive color mixing must be reviewed.

Color Mixing

Additive and subtractive color mixing are shown in figure 6.1 and color plate 1. Additive mixing corresponds to adding light sources, such as overlapping beams of colored lights. Also, additive mixing applies to the red, green, and blue dots (or lines) on a computer monitor. Although a magnifying glass will reveal individual colored dots on a computer screen, at normal viewing distances the human eye fuses the dots to form a full set of colors. With additive mixing, red and green produce yellow; green and blue yield cyan; blue and red combine to form magenta; and red, green, and blue produce white. Black corresponds to the absence of any light source.

Subtractive color mixing (filtering) applies to colors on paper illuminated by ambient white light. The colors may overlap, or they may be separate small dots. An example is the the small cyan, magenta, and yellow dots produced by an ink-jet printer. For subtractive mixing, cyan and magenta combine to form blue; magenta and yellow yield red; yellow and cyan form green; and cyan, magenta, and yellow produce black. The ink dyes subtract color from the ambient white light. For example, cyan filters out red in the ambient light, and magenta filters green; therefore cyan combined with magenta results in the

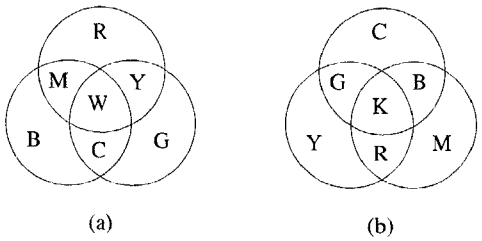


Figure 6.1 Diagrams illustrating (a) additive color mixing and (b) subtractive color mixing. The letters indicate the colors: R, red; G, green; B, blue, C, cyan; M, magenta; Y, yellow; W, white; and K, black.

color blue being reflected; see figure 6.2. Similarly, magenta together with yellow filters green and blue, with the result that white light reflects as red.

The RGB and CMY Color Cubes

The RGB color cube maps out color space by plotting red, green, and blue along perpendicular axes; see figure 6.3a. Colors are obtained by additive color mixing of the three primary colors (RGB). Accordingly, grays lie along the diagonal of the cube, from the origin (black) to the far corner (white). Along the cube face on which red = 0, green and blue combine, reaching saturated cyan at the corner corresponding to maximum amounts of green and blue. As illustrated in the figure, magenta and yellow lie on the corners of other faces.

The CMY cube in figure 6.3b is similar to the RGB cube, except that subtractive color mixing determines the colors in the cube. Grays occur along the cube diagonal, starting with white at the origin and ending with black at the far corner, corresponding to maximum amounts of cyan, magenta, and yellow.

The CMY values for a given color may be obtained from the RGB values by means of the following transformation, in which the CMY and the RGB colors are each normalized to a maximum value of 1.

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{6.1}$$

Equation 6.1 is given in terms of column vectors, presented in the form of 1×3 matrices; see appendix C for a review of matrix operations.

The above transformation may be readily interpreted for the cases shown in figures 6.2a–b. For example, figure 6.2a corresponds to $R = 0$, $G = 1$, and $B = 1$, which transform to $C = 1$, $M = 0$, and $Y = 0$. For figure 6.2b, $R = 0$, $G = 0$, and $B = 1$, which correspond to $C = 1$, $M = 1$, and $Y = 0$.

The columns in equation 6.1 may be transposed to provide the following transformation for normalized RGB values in terms of normalized CMY values.

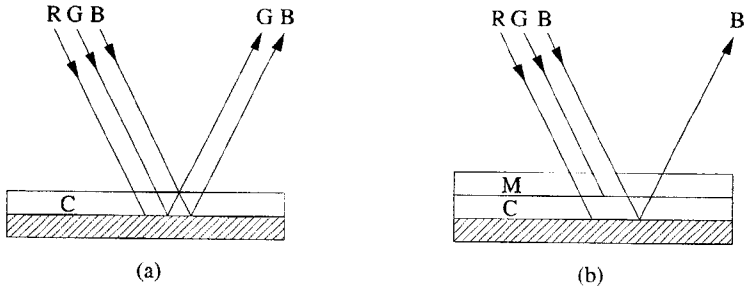


Figure 6.2 Diagrams illustrate the interaction of white light (W = R + G + B) incident on paper (cross-hatched layer) coated with the color cyan (C) in (a) and the colors cyan and magenta (M) in (b).

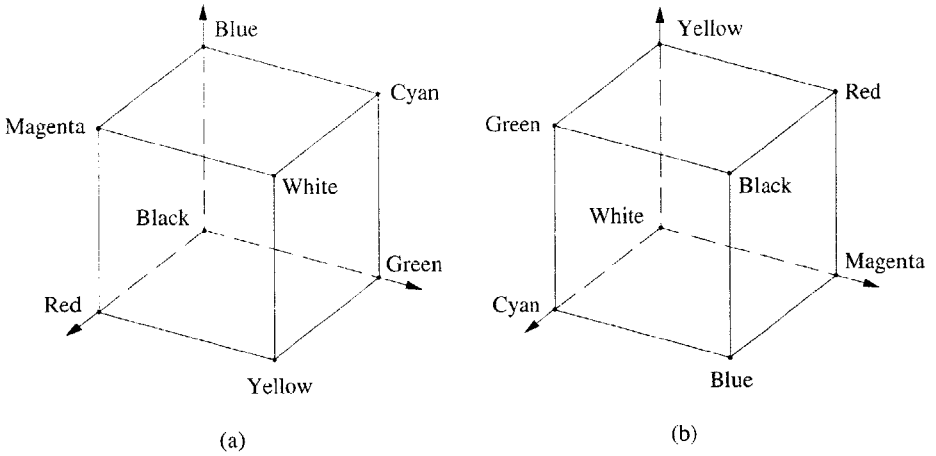


Figure 6.3 Color cubes: (a) the RGB cube for additive color mixing and (b) the CMY cube for subtractive mixing.

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} C \\ M \\ Y \end{bmatrix} \tag{6.2}$$

Most color printers have four colors: black *(K)* is added to CMY because controlling the quality of a black produced by mixing cyan, magenta, and yellow inks is difficult. Also, black is the most frequently used color, so that for ink jets, for example, it is common to have one black ink cartridge, and one color cartridge that has separate bladders for the cyan, magenta, and yellow inks. For the CMYK system the amount of K (black) in a given color will equal the minimum of C, M, and Y that would have been present if there were no black ink. Then the amount of the other components will be reduced by the amount of black. For example, if Y is the smallest component in a given color for the CMY system, then for CMYK, Y would be replaced by K; C would be replaced by $C - K$; and M would be replaced by $M - K$.

The HSV and HLS Color Models

As mentioned at the beginning of this section, the three-dimensionality of color space can be viewed in terms of hue *(H)*, saturation *(S)*, and value *(V)*, with value being similar to lightness or brightness. It is easier for most people to select a color based on *H, S,* and *V,* than on *R, G,* and *B*. That is, an HSV model is a more natural way for us to formulate colors. Although most programming languages require RGB values for setting a color on the computer monitor, a transformation from the HSV model to the RGB model can be used in software to select colors based on *H, S,* and *V.* For example, the Macintosh "color picker" illustrated in plate 2a permits the user to choose a color based on HSV values, as well as on RGB values.

The HSV color model is usually referenced to the hexcone shown in figure 6.4a. In the hexcone, hue is measured around the circumference of the hexagonal cross sections, with red at 0°, yellow at 60°, green at 120°, cyan at 180°, etc., as shown in the figure. Saturation varies from zero on the axis (center of the hexagonal cross sections) to a maximum value on the side surfaces of the hexcone. The value varies from zero (black) at the bottom vertex of the hexcone to a maximum (white) at the center of the top (hexagonal) face. The cylindrical representation illustrated in figure 6.4b corresponds to the Macintosh color picker, in which the circle showing the hue and saturation becomes black when the value (slider bar in plate 2a) is set equal to zero. The saturation and value variables are often given a maximum of 1, although the Macintosh picker displays maximum values of 65,553 for all color dimensions: H, S, V, R, G, and B.

Consider a normalized RGB system in which R, G, and B all vary between 0 and 1, and an HSV system with H varying from 0° to 360°, and with S and V ranging from 0 to 1. The value V can be identified with the maximum of R, G, and B.

$$V = \max(R,G,B) \tag{6.3}$$

Note that because R, G, and B are normalized to maximum values of 1, V has a maximum value of 1.

The saturation is given by the following relation.

$$S = \frac{\max(R,G,B) - \min(R,G,B)}{\max(R,G,B)} \tag{6.4}$$

Equation 6.4 reduces to appropriate limits: for gray shades, $S = 0$ occurs when $\max(R,G,B) = \min(R,G,B)$, which implies $R = G = B$. Also, S equals its maximum value of 1 when $\min(R,G,B) = 0$; that is, when one of the RGB colors is zero, a fully saturated color will occur somewhere on the side surfaces of the hexcone between the other two RGB values.

The relation between H and R, G, and B depends on whether the hue is nearest R (0°), G (120°), or B (240°). When H is near R, the hue is computed from the following equation.

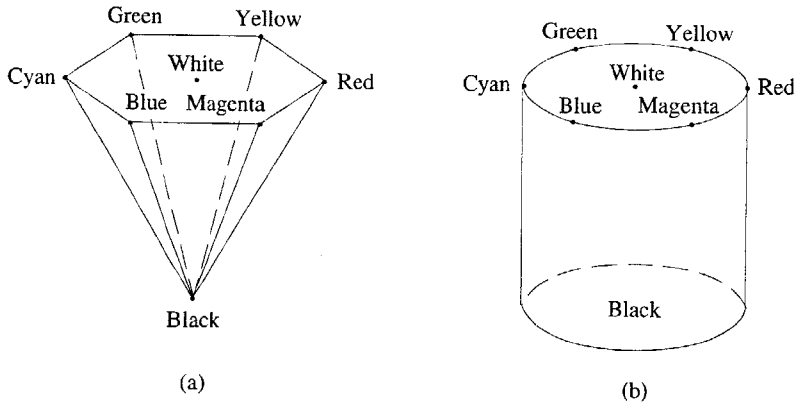


Figure 6.4 HSV color solids: (a) the hexcone, and (b) the circular cylinder.

$$H = \frac{60(G - B)}{\max(R,G,B) - \min(R,G,B)} \tag{6.5}$$

When $G = B$, equation 6.5 reduces to $H = 0°$, that is, to a red, as expected. On the other hand, when $G = R$, a yellow hue results and equation 6.5 gives $H = 60°$. If $B = R$, then equation 6.5 yields magenta at $H = -60°$, and 360° must be added to the result to put the hue in the proper degree range (0° to 360°).

When H is nearest G or B, the hue is given by the following relations.

$$H = 120 + \frac{60(B - R)}{\max(R,G,B) - \min(R,G,B)} \tag{6.6}$$

$$H = 240 + \frac{60(R - G)}{\max(R,G,B) - \min(R,G,B)} \tag{6.7}$$

It is evident that the above equations reduce to the proper angles when the hue is yellow, green, cyan, blue, or magenta.

Equations 6.3 through 6.7 are implemented in the subroutine below. When the saturation $S = 0$, the hue is undefined, but for specificity it is assigned the value $H = 0$, as in the operation of the Macintosh color picker. In the subroutine it has been assumed that the variables in the RGB system can have values up to a maximum RGBMAX (given, for example, by a *#define* assignment in the main part of the program). In the first part of the subroutine the values *maxrgb* and *minrgb* are computed, corresponding to the quantities max(R,G,B) and min(R,G,B), respectively.

```
void rgb2hsv(double r,double g,double b)
{
    double maxrgb,minrgb,h,s,v,cr,cg,cb;

    r = r/RGBMAX; g = g/RGBMAX; b = b/RGBMAX;

    maxrgb = r;
    if (g > maxrgb) maxrgb = g;
    if (b > maxrgb) maxrgb = b;
    minrgb = r;
    if (g < minrgb) minrgb = g;
    if (b < minrgb) minrgb = b;

    v = maxrgb;
    if (v == 0.)
        s = 0.;
    else
        s = (maxrgb - minrgb)/maxrgb;
    if (s == 0.)
        h = 0.; /* actually, h is undefined */
    else
    {
        if (r == v) h = 60.*(g-b)/(s*v);
```

```
        if (h < 0) h = h + 360.;
        if (g == v) h = 120.+ 60.*(b - r)/(s*v);
        if (b == v) h = 240.+ 60.*(r - g)/(s*v);
    }
    Hue = h*HMAX/360.; Saturation = s*SMAX; Value = v*VMAX;
}
```

In the above subroutine, the calculations are performed with H measured in degrees, and S and V each normalized to a maximum value of 1. Then the calculated values are transferred to global variables *Hue, Saturation,* and *Value,* which have maximum values HMAX, SMAX, and VMAX.

We will often wish to specify HSV values to be displayed on a computer that works with RGB values. Therefore, the inverse of the above transformation is required. For this HSV to RGB transformation, the six angular regions (of 60° each) of the hexagon are considered by introducing the variable *hexRegion,* which takes on six integer values. In the following subroutine, the values are scaled by dividing H by HMAX/6, S by SMAX, and V by VMAX. The transformation then computes normalized r, g, and b values that are multiplied by any desired scale RGBMAX at the end to determine values for the global variables *Red, Green,* and *Blue.*

```
void    hsv2rgb(double h,double s,double v)
{
    double r,g,b,c,d,e,f;
    int    hexRegion;

    h = h*6./HMAX; s = s/SMAX; v = v/VMAX;

    if (s == 0.)
    {
        r = v; g = v; b = v;
    }
    else
    {
        hexRegion = (int) floor(h);
        c = h - floor(h);
        d = v*(1.- s);
        e = v*(1.- s*c);
        f = v*(1.- s*(1.- c));
        switch(hexRegion)
        {
            case 0:
                r = v; g = f; b = d;
                break;
            case 1:
                r = e; g = v; b = d;
                break;
            case 2:
```

```
                r = d; g = v; b = f;
                break;
            case 3:
                r = d; g = e; b = v;
                break;
            case 4:
                r = f; g = d; b = v;
                break;
            case 5:
                r = v; g = d; b = e;
        }
    }
    Red = r*RGBMAX; Green = g*RGBMAX; Blue = b*RGBMAX;
}
```

The above transformations are included as part of the Macintosh color-picker package as the functions RGB2HSV() and HSV2RGB(). Other transformations included with the Macintosh software are CMY2RGB(), RGB2CMY(), HSL2RGB(), and RGB2HSL(). The transformations between CMY and RGB are given above in equations 6.1 and 6.2. The HSL model is often called the HLS model, and involves hue (H), lightness (L), and saturation (S).

Representations of the HLS model are shown in figure 6.5. The double hexcone is similar to the HSV hexcone, with hue being measured by the circumferential angle around hexagonal cross sections. Lightness is measured along the axis with black being at the bot-

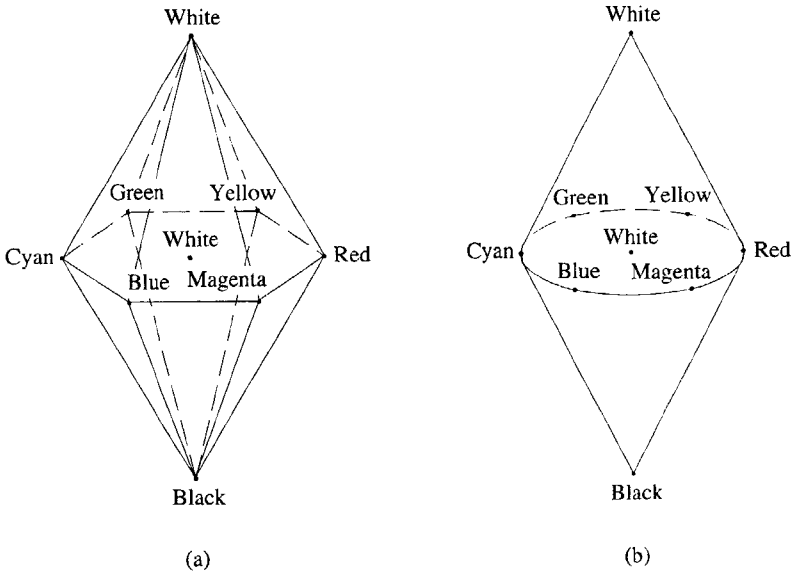


Figure 6.5 HLS color solids: (a) the double hexcone and (b) the double cone.

tom vertex and white at the top. Normally, black corresponds to $L = 0$, white to $L = 1$, and the hexagon that bisects the axis has a lightness $L = 0.5$. The double-cone representation of the HLS model corresponds to the Ostwald color space that was introduced in 1917. In the HLS model, saturation increases from the value 0 on the axis to the value 1 on the surface of the double hexcone or double cone.

Transformations for the HLS model are included in the books by Rogers (1985) and Watt (1989). Another important color model is the YIQ system adopted in 1953 by the National Television Standards Committee (NTSC) for color television broadcasting in the United States. Transformations between YIQ and RGB are given by Rogers (1985) and by Foley et al. (1990). In the next section, transformations will be studied that relate the RGB model to the tristimulus values XYZ and the CIE chromaticity diagram.

6.2 COLORIMETRY AND THE PERCEPTION OF COLOR

Colorimetry is the branch of color science that is concerned with the numerical specification of colors. Accurate representation of colors involves matching and mixing, which are very important in product design and manufacturing. Before we can describe colors by quantitative models, we must understand some basic facts about human visual perception.

The Perception of Color by the Human Eye

The human eye senses light by means of cone and rod cells. The rod cells are sensitive to low levels of light but cannot distinguish colors, only lightness or darkness. That is, when the ambient light is low at night or in the early morning, you cannot distinguish colors because only your rod cells, not your cone cells, can detect the low light levels—try picking out dark brown socks from black ones in ambient light at 6 a.m.

There are three types of cone cells in the human eye, with sensitivities peaking at long, medium, and short wavelengths within the visible spectrum. These cones are commonly called the red, green, and blue cones, although the red and green cones really have peak sensitivities in different parts of the yellow range of color. The fact that the cone cells have peak sensitivities in different parts of the spectrum enables us to distinguish colors. Furthermore, the three-dimensionality of color arises from the fact that there is visual stimulation from three different types of cone cells.

Our perception of a color, however, does not simply depend on the light spectrum from an object, but also on surrounding colors and conditions. Photographers know that different types of film are required for daylight and artificial light. The human eye compensates for these illumination differences to give us the impression that we are seeing the true colors of objects. Accordingly, our perception of colors depends on the physiology and psychology of vision, as well as on the physics of light.

To quantify color mixing, researchers performed a large number of experiments in which they asked people to determine the amounts of three standard colors that combined to match any given color. Each person viewed a projected spot of a given color and compared it with a color produced by mixing three standard colors (the three colors are pro-

jected on top of each other). Although visual perception varies among individuals, experiments with a large number of people have established results for a "standard observer." These experiments led in 1931 to the development of the widely used chromaticity diagram by the CIE.

Figure 6.6 illustrates the experiment used to establish color-matching functions $\bar{r}(\lambda)$, $\bar{g}(\lambda)$, and $\bar{b}(\lambda)$. At each wavelength λ, a monochromatic test color $E(\lambda)$ is matched by amount $\bar{r}(\lambda)$ of the red primary R, amount $\bar{g}(\lambda)$ of the green primary G, and amount $\bar{b}(\lambda)$ of the blue primary B. That is, the following linear relation holds between the monochromatic test color and the three primaries.

$$E(\lambda) = \bar{r}(\lambda)R + \bar{g}(\lambda)G + \bar{b}(\lambda)B \tag{6.8}$$

In the above equation R, G, and B are primary stimuli of *unit* amounts, and $E(\lambda)$ is a monochromatic test color of *unit* radiant power.

For primaries R, G, and B consisting of monochromatic sources at wavelengths 700.0, 546.1, and 435.8 nm, respectively, the color-matching functions (also called tristimulus values) are plotted in figure 6.7. Although the linear relation (6.8) holds at every wavelength, in some wavelength regions one of the color-matching functions takes on negative values—primarily $\bar{r}(\lambda)$ between the wavelength of the blue and green primaries, 435.8 to 546.1 nm, although $\bar{g}(\lambda)$ and $\bar{b}(\lambda)$ take on small negative values in other wavelength regions. This result means that in the experiment, matching was achieved in the 435.8 to 546.1-nm wavelength region by adding the red primary to the monochromatic source, rather than to the green and blue primaries.

Equation 6.8 holds at each wavelength λ; therefore, a color with a spectral power distribution $P(\lambda)$ may be integrated over wavelength to obtain its total radiant power C. Integration of equation 6.8 over wavelength gives the following relation for C.

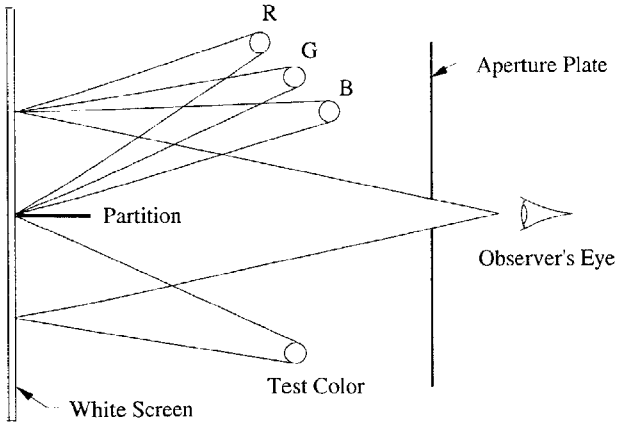


Figure 6.6 Experiment to determine color-matching functions. An observer views the superposition of three standard lights (R, G, and B) on one side of a partition and compares the resulting color with a test color projected on the other side of the partition.

$$C = rR + gG + bB \tag{6.9}$$

where

$$C = \int P(\lambda)\mathrm{d}\lambda, \qquad r = \int P(\lambda)\bar{r}(\lambda)\mathrm{d}\lambda, \qquad g = \int P(\lambda)\bar{g}(\lambda)\mathrm{d}\lambda, \qquad b = \int P(\lambda)\bar{b}(\lambda)\mathrm{d}\lambda$$

Color-matching functions for a different set of primaries can be obtained by a set of linear transformations from the above functions. Accordingly, a new set of functions was generated from the experimental data shown in figure 6.7. The new functions were the 1931 standard observer tristimulus values $\bar{x}(\lambda)$, $\bar{y}(\lambda)$, and $\bar{z}(\lambda)$. These new functions have no negative values, as shown in figure 6.8.

The three primaries corresponding to the CIE color-matching functions are designated as X, Y, and Z. The amounts of X, Y, and Z primaries needed to match a color light source having a spectral radiant power distribution $P(\lambda)$ are given by

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

where

$$k = \frac{1}{\int P_w(\lambda)\bar{y}(\lambda)\mathrm{d}\lambda} \tag{6.10}$$

In the above equations the tristimulus values X, Y, and Z have been normalized by dividing by the integral of the spectral radiant power $P_w(\lambda)$ for a standard white light source multiplied by $\bar{y}(\lambda)$. Accordingly, a bright white light has a luminance Y of 1. (Sometimes a fac-

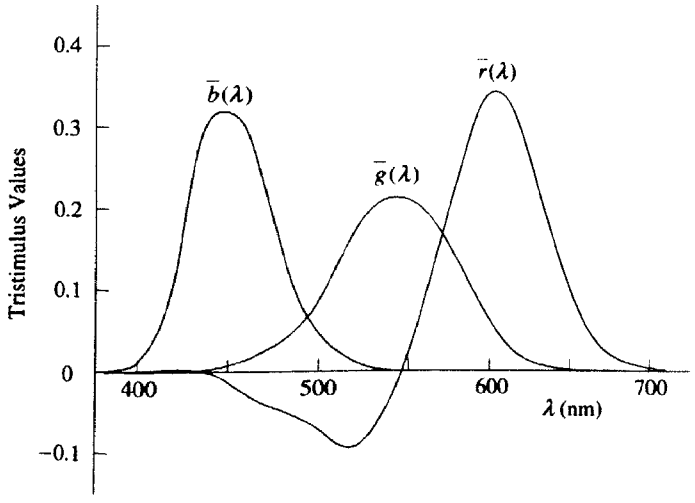


Figure 6.7 Color-matching functions (tristimulus values) for the experiment illustrated in figure 6.6. The R, G, and B primaries were monochromatic light sources at 700.0, 546.1, and 435.8 nm, respectively.

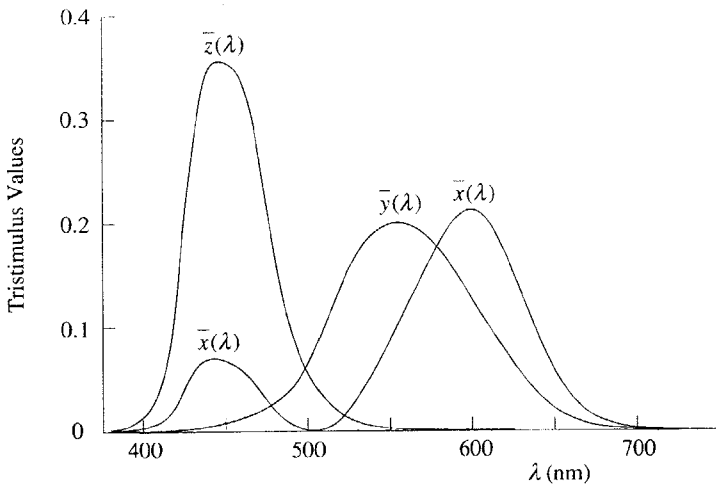


Figure 6.8 The CIE standard color-matching functions (tristimulus values).

tor of 100 is introduced into the equation for k to normalize a bright white light to a luminance of 100.) The primary Y has been set up to represent luminance (lightness, brightness, or value), and as shown below, the X and Y values will be further normalized to construct a chromaticity diagram that represents the chromatic aspects of color.

The CIE Chromaticity Diagram

The tristimulus values are divided by the sum $X + Y + Z$ to form the chromaticity coordinates x, y, and z.

$$x = \frac{X}{X + Y + Z}, \qquad y = \frac{Y}{X + Y + Z}, \qquad z = \frac{Z}{X + Y + Z} \tag{6.11}$$

From the above equations, it may be readily shown that $z = 1 - x - y$. Therefore, z is not an independent parameter, and the luminance Y is used with the chromaticity coordinates x and y to specify a color.

The chromaticity diagram is shown in figure 6.9. Wavelengths are identified on the horseshoe-shaped curve that represents the bounds of visible colors, with the numbers from 400 to 700 representing the wavelengths in nanometers (nm) of the pure spectral colors. The straight line at the bottom connecting the limiting blue-violet (shortest visible wavelength) with the limiting red (longest wavelength) is called the line of purples. The colors along this straight line do not occur in the natural spectrum, but do occur as mixtures of red and blue-violet.

All points within the horseshoe-shaped curve represent visible colors, but only those within a triangle can be produced on a standard computer monitor. The red, green, blue, and white coordinates for the solid triangle are (Rogers 1985): $x_r = 0.628$, $y_r = 0.346$, $x_g =$

0.268, $y_g = 0.588$, $x_b = 0.150$, $y_b = 0.070$, $x_w = 0.313$, and $y_w = 0.329$. Color plate 3 illustrates the gamut of colors within the triangular region.

The triangular region of available colors depends on the particular monitor. The dashed triangle in figure 6.9 was determined from measurements on a Silicon Graphics monitor. The smaller, six-sided dashed polygon in figure 6.9 shows the color gamut for a color printer. Unlike colors on a monitor, the colors on a printer do not lie within a triangular region because they are determined by subtractive color mixing rather than by additive mixing. The dashed polygon for the printer was plotted by connecting measured coordinates for the six colors R, Y, G, C, B, and M. In general, a color hardcopy device has a much smaller gamut of colors than a color monitor.

Figure 6.10 illustrates some of the features of the chromaticity diagram. A straight line passing through the standard white source (W in the figure) connects complementary

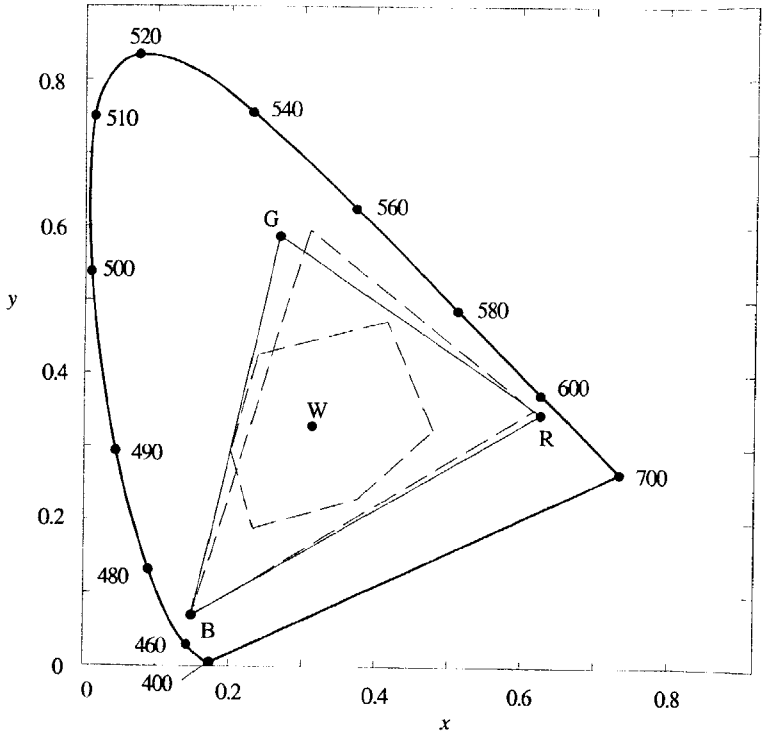


Figure 6.9 The CIE chromaticity diagram, with interior polygons showing color gamuts for two monitors and a printer. The triangle with the solid outline was plotted from monitor data given by Rogers (1985). The dashed polygons were plotted from data measured with a photometer by Michael Bailey (San Diego Supercomputer Center); the dashed triangle was measured on a Silicon Graphics 4D/320 VGX monitor; and the dashed, six-sided polygon was measured on hard copy from a Canon CLC-500 color copier/printer.

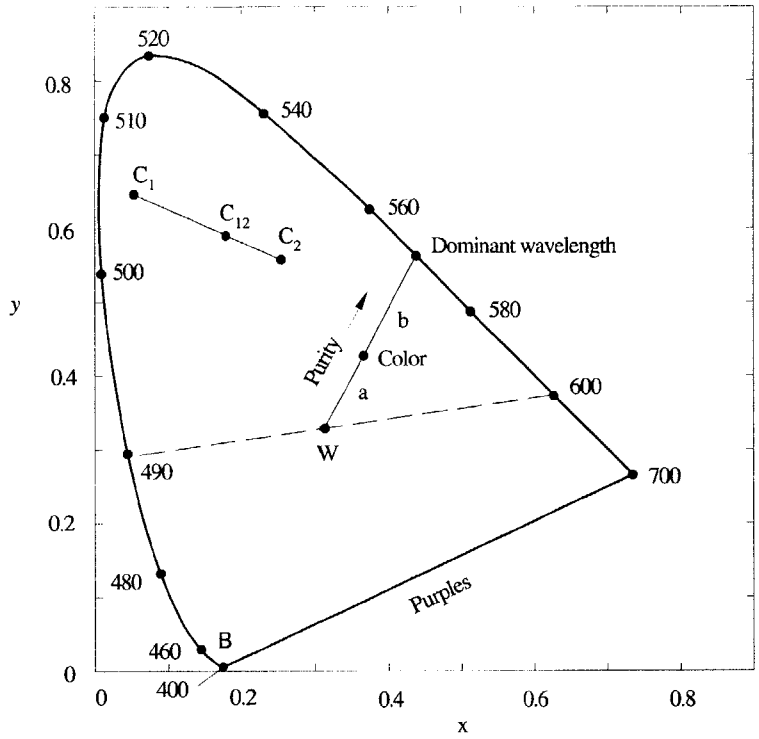


Figure 6.10 The CIE chromaticity diagram with straight lines that illustrate color characteristics and color mixing.

colors on opposite sides of the diagram. For example, the complement of the color corresponding to a 600-nm monochromatic light source is the color corresponding to a wavelength of approximately 490 nm (dashed line in the figure).

The chromatic character of a color may be described by the purity (similar to saturation) of the color and its dominant wavelength (similar to hue). In figure 6.10, a straight line is drawn from the white point W through a dot representing a given color to the dominant wavelength of the color at the boundary of the horseshoe curve. The purity of the color is defined as the distance a from W to the color divided by the distance $a + b$ between W and the dominant wavelength.

A straight line drawn between two colors (C_1 and C_2 in figure 6.10) contains all colors that may be obtained by mixing these two. From Grassman's laws for additive color mixture, two colors are mixed by summing their X, Y, and Z values. From the definition (6.11) of the chromaticity coordinates, the following relations are readily derived.

$$X + Y + Z = \frac{Y}{y}, \qquad X = x\frac{Y}{y}, \qquad Z = (1 - x - y)\frac{Y}{y} \tag{6.12}$$

For the two colors C_1 and C_2, define $T_1 = Y_1/y_1$ and $T_2 = Y_2/y_2$. Then from equation 6.12 the sum of the XYZ primaries for the two colors is $T_1 + T_2$. After forming the sum $X_{12} = X_1 + X_2$, x_{12} is calculated from equation 6.11, with y_{12} obtained in a similar manner.

$$x_{12} = \frac{x_1T_1 + x_2T_2}{T_1 + T_2}, \qquad y_{12} = \frac{y_1T_1 + y_2T_2}{T_1 + T_2}, \qquad Y_{12} = Y_1 + Y_2 \tag{6.13}$$

Note that T_1 is proportional to the luminance Y_1, and T_2 is proportional to Y_2. Therefore, when $Y_2 = 0$, the chromaticity coordinates reduce to those of color 1 ($x_{12} = x_1$ and $y_{12} = y_1$); similarly, if $Y_1 = 0$, then $x_{12} = x_2$ and $y_{12} = y_2$. When both Y_1 and Y_2 are nonzero, the chromaticity coordinates of the mixture lie on a line joining C_1 and C_2, as given by equations 6.13.

A mixture of three colors will be bounded by the triangle formed by connecting the colors with straight lines in the chromaticity diagram. The solid triangle shown in figure 6.9 and color plate 3 shows the colors (for a given luminance) that can be made by mixing the R, G, and B colors on a computer monitor.

The RGB color space of a monitor can be transformed to the CIE XYZ color space. The transformation may be written in the following form.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} X_r & X_g & X_b \\ Y_r & Y_g & Y_b \\ Z_r & Z_g & Z_b \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{6.14}$$

The tristimulus values X_r, Y_r, and Z_r are normalized for unit values of the RGB primaries; for example, $R = 1$, $G = 0$, and $B = 0$ yield $X = X_r$, $Y = Y_r$, and $Z = Z_r$.

Equations 6.11 may be used to relate the X_r, Y_r, and Z_r terms to x_r and y_r, and to the quantity $C_r = X_r + Y_r + Z_r$. Along with similar relations for the other terms, equation 6.14 becomes

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} x_rC_r & x_gC_g & x_bC_b \\ y_rC_r & y_gC_g & y_bC_b \\ (1 - x_r - y_r)C_r & (1 - x_g - y_g)C_g & (1 - x_b - y_b)C_b \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{6.15}$$

The quantities C_r, C_g, and C_b can be evaluated by using the tristimulus values for white (X_w, Y_w, and Z_w, obtained from the chromaticity coordinates x_w and y_w, plus unit luminance $Y_w = 1$).

For the color monitor x- and y-values listed above for the colors red, green, blue, and white, the RGB to XYZ transformation (6.15) reduces to the following matrix equation (see Rogers 1985).

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.478 & 0.299 & 0.175 \\ 0.263 & 0.655 & 0.081 \\ 0.020 & 0.160 & 0.908 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{6.16}$$

After the above relation is used to obtain values for X, Y, and Z, the normalized chromaticity coordinates x and y may be obtained from equations 6.11.

The matrix equation 6.16 may be inverted to obtain the following relation (see appendix C for a matrix-inversion procedure).

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 2.739 & -1.145 & -0.424 \\ -1.119 & 2.029 & 0.033 \\ 0.138 & -0.333 & 1.105 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{6.17}$$

The above matrix equations may be easily incorporated into subroutines, as shown below. In the subroutine rgb2cie(), *x, y,* and *ycap* are assumed to be global variables; and in the subroutine cie2rgb(), *Red, Green,* and *Blue* are global variables.

```
void rgb2cie(double r,double g,double b)
{
    double  xcap,zcap;

    xcap = 0.478*r + 0.299*g + 0.175*b;
    ycap = 0.263*r + 0.655*g + 0.081*b;
    zcap = 0.020*r + 0.160*g + 0.908*b;

    x = xcap/(xcap + ycap + zcap);
    y = ycap/(xcap + ycap + zcap);
}

void cie2rgb(double x,double y,double ycap)
{
    double  xcap,zcap;

    xcap = x*ycap/y;
    zcap = (1.- x - y)*ycap/y;

    Red =  2.739*xcap - 1.145*ycap - 0.424*zcap;
    Green = -1.119*xcap + 2.029*ycap + 0.033*zcap;
    Blue =  0.138*xcap - 0.333*ycap + 1.105*zcap;
}
```

6.3 USE OF COLOR PALETTES

Workstations with 24-bit color are becoming more common, but many designers still are working with systems having only 4-bit or 8-bit color. On a 24-bit system, $2^{24} = 16{,}777{,}216$ colors can be displayed simultaneously on the computer screen. At the low end, only $2^4 = 16$ colors or $2^8 = 256$ colors can be displayed simultaneously. In between these color capabilities, graphics cards having 1M byte of memory can display $2^{15} = 32{,}768$ colors simultaneously on a screen of 800×600 pixels.

For line drawings, 16 or fewer colors may be enough to distinguish different parts or layers on the drawing. For scientific visualization, as discussed in the next section, approximately 256 colors are required to produce a fairly smooth variation in hue for color plots of a variable throughout a region. For photorealistic rendering, however, all three dimensions of color must be used to show appropriate variations in lighting, saturations, and

hues. Accordingly, the best output for rendering can require up to 24 bits of color. When your system does not have the number of colors needed for rendering, the effective number of colors can be expanded by the use of dithering (at the expense of decreased resolution); see section 6.5.

As discussed in section 2.1, a color look-up table may be used to expand the number of colors available to your system. Although the number of colors that may be displayed simultaneously is not increased by a look-up table, the simultaneous colors can be selected from a much larger number of colors. That is, a look-up table makes it possible to change the *palette* of colors used for drawing on the screen.

The particular procedures for changing the color palette depend on your system. For Microsoft C on IBM PC compatibles, the functions that change the palette in VGA mode are _remappalette() and _remapallpalette(). In the VGA mode, 6 bits are used to assign the intensity of each of the RGB colors. Therefore, R, G, and B can each have $2^6 = 64$ different values, for a total number of $64 \times 64 \times 64 = 262{,}144$ colors. The standard VGA mode permits only 16 of these colors to appear on the screen at a time for 640×480 pixel resolution, or 256 simultaneous colors for 320×200 pixel resolution.

To illustrate the procedure for setting a palette color, the following Microsoft C subroutine uses the function _remappalette() to reset the color for one of the colors on the current palette (in contrast, _remapallpalette() resets all of the palette colors). In the subroutine, the integer *clrIndex* denotes one of the color numbers 0 to 15 in the 640×480 pixel mode, or a number 0 to 255 in the 320×200 pixel mode. The long integers *red*, *green*, and *blue* have values from 0 to 63.

```
void setPal(int clrIndex,long int red,
                long int green,long int blue)
{
     long int clr;

     clr = (blue<<16) | (green<<8) | red;
     _remappalette(clrIndex,clr);
}
```

In the above subroutine, the color red (0–63) is put into the first byte of the 4-byte long integer *clr*, the green into the second byte (shifted by 8 bits from the 6-bit red color), blue into the third byte, with the fourth byte being empty; see figure 6.11.

Color pickers programmed with Microsoft C are shown in color plates 2b–f. These programs use the hsv2rgb() subroutine given in section 6.1 along with the _remappalette()

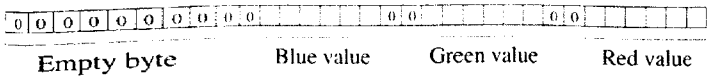


Figure 6.11 Long-integer storage of a color value in the VGA mode for IBM PC compatibles.

function to continuously change a color in response to the user's mouse input. All of these color pickers are limited to a display of only 16 simultaneous colors, but a given color in the palette can be changed to any of 262,144 colors.

The color picker shown in plate 2b is similar to the Macintosh picker. It features a color wheel in which a dot can be dragged by the mouse cursor to change the hue and saturation. The Macintosh dot is less visible because it lies on top of the colors in the wheel—the color picker in plate 2b is limited by the 16-color display, so colors are placed only on the rim of the color wheel. In this color picker, slider (scroll) bars are used to change the color value (brightness) and also to directly change the amounts of red, green, and blue. A slider bar subroutine is discussed in chapter 9, section 9.5.

In plate 2c, hue and saturation levels are displayed by the dot location in the hexagonal region on the left, while the color is displayed around the rim of the color wheel on the right (this permits the color to be compared with many adjacent colors inside the wheel). The hue, saturation, and value levels are changed with the arrow buttons at the lower left.

To facilitate rapid selection of a color, the picker in plate 2d displays many colors close in hue to the one being changed (these colors can be selected at any time with the mouse). The center of this picker also displays a color wheel; the yellow radius can be dragged around the circle to change hue (which may also be changed by clicking on the + and − boxes). The angle between the cyan radii in the color wheel shows the range of hues for the colors displayed at the top (this range can be changed with the + and − boxes above the view range label). Value and saturation levels can be changed with the arrow buttons at the left; the horizontal and vertical yellow lines display the current levels (the saturation level is also displayed by the size of the inner yellow circle in the color wheel). Red, green, and blue levels can be changed directly with the buttons on the right.

The color pickers in plates 2e and f work effectively without color wheels. The color picker in 2e has slider bars that graphically display hue, saturation, and value. The cursor cross used for selection changes to a downward arrow when it is slid along one of these bars to change a color variable.

The color picker in plate 2f displays a bar chart of red, green, and blue amounts. By changing hue, saturation, and value levels with the + and − buttons on the left, the user can observe the rise and fall of the red, green, and blue bars. This display readily shows that one of the primary colors is reduced to zero as the saturation is increased to 100% (in this picker the maximum amounts of hue, saturation, and value are set at 360). Also, all bars move up when the value level is increased, and move down when the value is decreased.

6.4 COLOR FOR SCIENTIFIC VISUALIZATION

Color provides another dimension for the presentation of scientific and engineering data. A variable to be plotted can be color coded; that is, a spectrum of colors can represent a range of values. Plate 4 shows a color plot of the pressure coefficient C_p throughout the flow field surrounding an airfoil. A color key at the bottom of the plot identifies colors with C_p values.

The pressure coefficient plotted in plate 4 is defined by the equation

$$C_p = \frac{p - p_\infty}{\frac{1}{2}\rho_\infty V_\infty^2} = \frac{V_\infty^2 - V^2}{V_\infty^2} \tag{6.18}$$

in which p is pressure, ρ density, V velocity, and the subscript ∞ denotes freestream values far from the airfoil. Accordingly, C_p is a dimensionless parameter that is proportional to the difference between the local pressure p and the freestream pressure p_∞; alternatively, C_p may be expressed in terms of the difference between the square of the freestream velocity V_∞ and the square of the local velocity V. As observed from plate 4, the pressure coefficient is negative ($p < p_\infty$ and $V > V_\infty$) above the airfoil and positive over most of the region below the airfoil.

The color key at the bottom of plate 4 was selected by setting both the saturation S and value V of all the colors equal to their maximum amounts. Then the color spectrum was established by varying the hue H from red through violet (the range was extended to only about 82% of the maximum amount for hue, so that there would be sufficient difference between the colors of minimum and maximum hues, corresponding to minimum and maximum C_p values).

The color plot was then constructed by calculating C_p at the position of each pixel and setting the pixel color according to the color key. The calculation of C_p follows from the analytical solution for a Joukowski airfoil. With this solution, the airfoil thickness, camber (curvature), and angle of attack can be varied by the program user. Without color, individual line plots of C_p on the upper and lower airfoil surfaces can be drawn; on the other hand, color permits C_p to be plotted throughout the flow field near the airfoil, not just on the surface of the airfoil.

5 DITHERING

The printing industry uses a halftone process in which different gray scales or shades are produced by dots of different sizes. Although the halftone process can be simulated on a computer, most computer graphics software employs a dither pattern. Dither patterns are based on square cells composed of a matrix of $n \times n$ pixels, where the cell size n determines the number of gray shades that can be produced. (Dithering can also be used to increase the number of colors available with monitors and printers.)

The shade of gray produced by a dither pattern depends on the number of black pixels relative to the number of white ones in a cell. Gray values between white and black can be obtained by randomly distributing black pixels on a white background, as illustrated at the top of figure 6.12. Although this procedure can produce appropriate gray values, the overall effect of a *random* dither is to produce a "dirty" or "foggy" effect on the scene being rendered. A much better procedure is to use an *ordered* dither, which produces cleaner shading, as shown at the bottom of figure 6.12. In an ordered dither, pixels are turned on in a prescribed pattern to produce desired gray values.

The total number of required gray values determines the size of the cell needed. For example, a 2×2 pixel cell can produce five values of gray when 0, 1, 2, 3, or 4 of the pixels are filled; see figure 6.13. The order in which the pixels are filled is important; for ex-

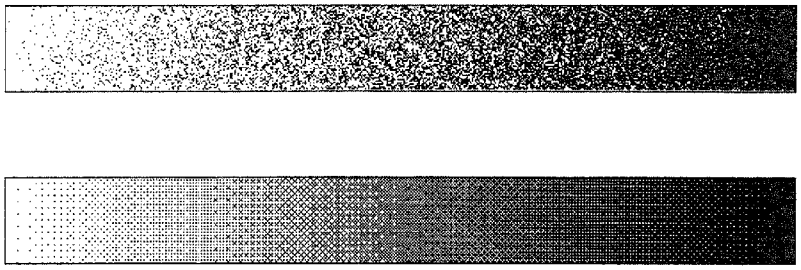


Figure 6.12 Gray scales produced by a random dither (top) and by an ordered dither (bottom).

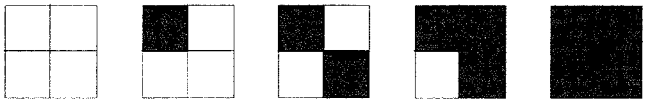


Figure 6.13 The five gray shades available with a 2 × 2-pixel matrix.

ample, if we first fill the top pixels in a 2 × 2 cell, then at a 50% gray value, horizontal lines will be evident. Accordingly, a suitable procedure for filling the pixels is given by the dither matrix

$$[D_2] = \begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix} \tag{6.19}$$

The numbers in the above equation provide the prescription for filling a matrix: first fill the upper left pixel, then the lower right pixel, followed by the upper right pixel, and finally the lower left pixel. The resulting five gray values (counting white with no filled pixels) are shown in figure 6.13; see also the left part of the figure at the beginning of this chapter.

A 3 × 3 pixel cell is filled by the prescription

$$[D_3] = \begin{bmatrix} 2 & 6 & 4 \\ 5 & 0 & 1 \\ 8 & 3 & 7 \end{bmatrix} \tag{6.20}$$

Dither matrices of higher order can be constructed from the formula

$$[D_n] = \begin{bmatrix} 4D_{n/2} & 4D_{n/2} + 2U_{n/2} \\ 4D_{n/2} + 3U_{n/2} & 4D_{n/2} + U_{n/2} \end{bmatrix} \tag{6.21}$$

in which $U_{n/2}$ stands for the unit matrix, which has the value 1 for every element. Therefore, the following $[D_4]$ matrix arises from equation 6.21 when equation 6.19 is used for $[D_2]$.

$$[D_4] = \begin{bmatrix} 0 & 8 & 2 & 10 \\ 12 & 4 & 14 & 6 \\ 3 & 11 & 1 & 9 \\ 15 & 7 & 13 & 5 \end{bmatrix} \tag{6.22}$$

The following matrix is used for the 5 × 5 pixel case:

$$[D_5] = \begin{bmatrix} 21 & 10 & 17 & 14 & 23 \\ 15 & 2 & 6 & 4 & 9 \\ 20 & 5 & 0 & 1 & 18 \\ 12 & 8 & 3 & 7 & 13 \\ 24 & 16 & 19 & 11 & 22 \end{bmatrix} \tag{6.23}$$

With the above lower-order matrices, equation 6.19 can be used to generate dither matrices $[D_6]$, $[D_8]$, $[D_{10}]$, $[D_{12}]$, $[D_{16}]$, etc. The matrix $[D_8]$ is of particular interest because graphics libraries often feature convenient procedures for storing 8 × 8 pixel matrices. The Macintosh has a pixel editor for interactive use in creating patterns of 8 × 8 pixels, and Microsoft C for IBM PC compatibles has the function _setfillmask(*userArray*), in which *userArray* is the current fill pattern set up by the user as an 8-byte array. The bottom strip in figure 6.12 illustrates the range of gray scales that can be obtained by an 8 × 8 dither array (see also the right side of the figure at the beginning of the chapter).

Ordered dithering is very useful for increasing the range of colors in rendered images. Because rendering attempts to produce photorealistic images, a large range of colors is required for good results. For a low-end system that can display only 16 colors, dithering can expand the number of colors to 256. This procedure is used in renderings by AutoCAD and AutoShade with systems that can display only 16 colors in VGA (640 × 480 pixel) resolution. The disadvantage of using dithering is that the effective resolution is reduced, because shades are produced by pixel matrices rather than by intensity variations in a single pixel. Accordingly, for rendering it is usually better to set up your system for 320 × 200 pixel resolution with 256 colors at each pixel, rather than for 640 × 480 pixel resolution with only 16 colors at each pixel.

Dithering of colored pixels is important for printers, as well as for monitors. The printed quality of a rendering is enhanced by using the printer driver developed for your software (for example, the Hewlett-Packard PaintJet driver supplied with AutoShade). That is, the printer driver optimizes the dithering of the cyan, magenta, yellow, and black ink-jet dots on paper, so that the result is much better than a screen dump of the monitor pixels.

EXERCISES

6.1 If your system can display only 256 or fewer simultaneous colors, write a program to display its default palette as an array of filled rectangles.

6.2 Change the palette of your computer system's colors to gray scales and display the palette from white to black as thin rectangles from left to right across the screen. The result should resemble a shaded cylinder.

6.3 Write a program that displays the colors in a color cube as a series of cross sections along the red, green, or blue axis. If your system displays 256 colors, then retain the black color for the background and display 15 × 15 = 225 squares of color for each cross section. A total of 15 cross sections should be displayed while progressing along a given color axis (let the program user select the red, green, or blue color axis).

6.4 Design an RGB color picker. That is, write a program that changes the color palette of your system in response to values entered for red, green, and blue color components. The resulting color should be displayed and compared with other colors. Write your program so that colors can be changed quickly and efficiently.

6.5 Write a program for an HSV color picker; that is, have the user enter numbers for hue, saturation, and value. Display RGB values and HSV values along with the new colors and comparison colors.

6.6 Develop a generalized color picker with which the user can change any of the quantities H, S, V, R, G, or B. Values should be changed with a convenient user interface, such as slider bars. Examples are shown in plate 2, but your color picker should be of your own design.

6.7 For CIE diagram coordinates consider the statement "roses are $x = 0.441$, $y = 0.305$; violets are $x = 0.282$, $y = 0.218$." Check out these values by transforming to RGB values for a luminance $Y = 0.32 \times$ RGBMAX, where RGBMAX is the maximum value for RGB colors on your system. Display the resulting colors on your computer monitor.

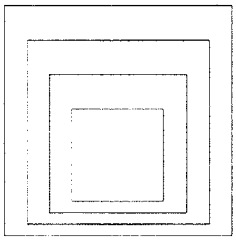
6.8 Write a program that constructs a color plot for a solution to the heat-conduction equation in a rectangular region.

6.9 Write out (a) the 6×6 ordered-dither matrix $[D_6]$, and (b) the matrix $[D_8]$.

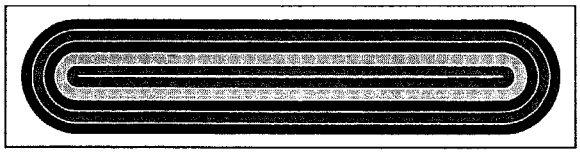
6.10 Set up an 8×8 ordered-dither matrix with your programming language. Check your results by printing out the 65 gray shades from white to black.

6.11 Write a program to produce a random dither of gray shades from white to black, similar to that shown in figure 6.12.

6.12 Look up three paintings of squares by Josef Albers and try to reproduce them on your computer screen (see, for example, *Albers*, by W. Spies, New York: Harry N. Abrams, Inc., 1970; *Josef Albers*, by J. Wissmann. Recklinghausen: Verlag Aurel Bongers, 1971). If you have a limited number of colors that can be displayed simultaneously on your monitor, change the palette to achieve a good match of colors with the painting. As illustrated below, Albers squares have sides of the following dimensions (arbitrary units): 8, 12, 16, and 20. The squares are separated by 2 units at their sides, 3 units at their tops, and 1 unit on the bottom.



6.13 By referring to a book on the paintings of Frank Stella (for example, *The Prints of Frank Stella*, by R. H. Axsom, New York: Hudson Hills Press, 1983), produce on your computer screen an image that approximates a painting from his racetrack series (black-and-white example shown below). Try to match the colors in the painting.



IOGRAPHY

BURGER, D., AND D. GILLIES, *Interactive Computer Graphics: Functional, Procedural and Device-Level Methods.* Reading, Mass.: Addison-Wesley Publishing Co., 1989.

FOLEY, J. D., A. VAN DAM, S. K. FEINER, AND J. F. HUGHES, *Computer Graphics: Principles and Practice,* 2nd ed. Reading, Mass.: Addison-Wesley Publishing Co., 1990.

HALL, R., *Illumination and Color in Computer Generated Imagery.* New York: Springer-Verlag, 1989.

MACADAM, D. L., *Color Measurement: Theme and Variations.* Berlin: Springer-Verlag, 1981.

MARK, D., *Macintosh C Programming: Volume II, Mastering the Toolbox Using THINK C.* Reading, Mass.: Addison-Wesley, 1990.

ROGERS, D. F., *Procedural Elements for Computer Graphics.* New York: McGraw-Hill Publishing Co., 1985.

ROSSOTTI, H., *Colour: Why the World Isn't Grey.* Princeton: Princeton University Press, 1983.

STONE, M. C., "Color Printing for Computer Graphics," pp. 79–127 in *Computer Graphics Techniques: Theory and Practice,* edited by D. F. Rogers and R. A. Earnshaw. New York: Springer-Verlag, 1990.

THE WAITE GROUP, *Microsoft C Programming for the PC,* 2nd ed. Indianapolis: Howard Sams, 1990.

WAITE, M., S. PRATA, B. COSTALES, AND H. HENDERSON, *Microsoft QuickC Programming*, 2nd ed. Redmond, Wash.: Microsoft Press, 1990.

WATT, A., *Fundamentals of Three-Dimensional Computer Graphics.* Reading, Mass.: Addison-Wesley Publishing Co., 1989.

WYSZECKI, G., *Color Science: Concepts and Methods, Quantitative Data and Formulae,* 2nd ed. New York: John Wiley & Sons, 1982.