COMPUTER GRAPHICS

Second Edition

DONALD HEARN
M. PAULINE BAKER

Align EX1045
Align v. 3Shape
IPR2022-00144

COMPUTER GRAPHICS

Second Edition

DONALD HEARN
M. PAULINE BAKER

The authors and the Prentice Hall staff gratefully acknowledge Jerry Farm for providing the cover image, ***Raytracers' Recess.***

This image was rendered using Rayshade, a freeware ray tracer written by Craig E. Kolb. Rayshade was adapted for the Macintosh and compiled using Think C. A separate program, Guygen, was created to translate lists of arm- and leg-joint angles into Rayshade-compatible input files to define the little characters in the scene. The final image was transformed to a 35mm slide using an Autographix film recorder.

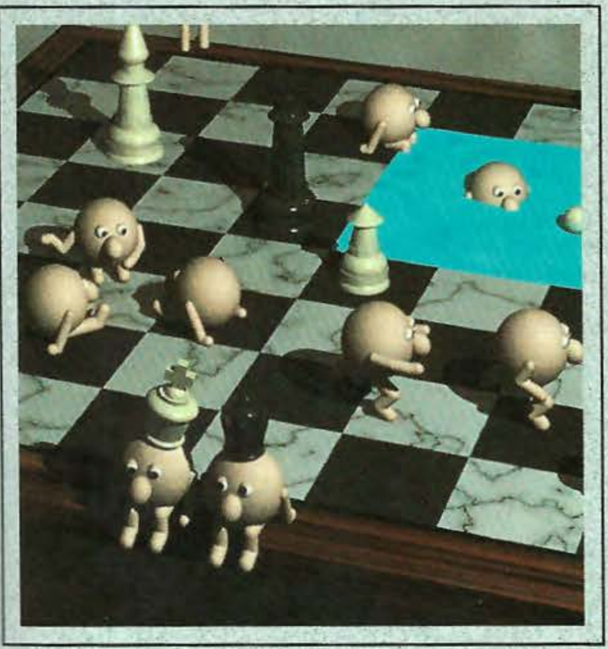
Including eye, reflected, and shadow rays, 67 million rays were traced to render the 1800 x 900 pixel image, taking 213 hours on a Macintosh II. Antialiasing was provided by subdividing each pixel into a 4 x 4 grid of subpixels, and soft-edged shadows were obtained by tracing multiple shadow rays to an area light source. Wood, marble, and other surface textures were rendered using the 3-D solid procedural textures available in Rayshade.

The author of ***Raytracers' Recess***, Jerry Farm, works in the aerospace industry and dabbles in computer-graphics topics such as ray tracing, 3-D stereo graphics, and real-time interactive graphics in his spare time.

ISBN 0-13-161530-0
90000
9 780131 615304

SECOND EDITION

Computer Graphics



Donald Hearn
Department of Computer Science and
National Center for Supercomputing Applications
University of Illinois

M. Pauline Baker
National Center for Supercomputing Applications
University of Illinois

PRENTICE HALL
Englewood Cliffs, New Jersey 07632

Library of Congress Cataloging-in-Publication Data

Hearn, Donald.
Computer graphics / Donald Hearn, M. Pauline Baker. — 2nd ed.
p. cm.
Includes bibliographical references and index.
ISBN 0-13-161530-0
1. Computer graphics. I. Baker, M. Pauline. II. Title.
T385.H38 1994
006.6—dc20 93-33698
CIP

Acquisitions Editor: *Bill Zobrist*
Editor-in-Chief: *Marcia Horton*
Production Editor: *Bayani Mendoza de Leon*
Copy Editor: *Peter Zurita*
Marketing Manager: *Tom McElwee*
Design Director: *Anne Bonanno Nieglos*
Designers: *Jules Perlmutter/Amy Rosen*
Cover Designer: *Amy Rosen*
Buyer: *Linda Behrens*
Editorial Assistant: *Phyllis Morgan*
Cover Computer Art: *Jerry Farm*

Published by Prentice Hall, Inc.
A Paramount Communications Company
Englewood Cliffs, New Jersey 07632

The author and publisher of this book have used their best efforts in preparing this book. These efforts include the development, research, and testing of the theories and programs to determine their effectiveness. The author and publisher make no warranty of any kind, expressed or implied, with regard to these programs or the documentation contained in this book. The author and publisher shall not be liable in any event for incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or use of these programs.

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-161530-0

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Simon & Schuster Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brașil, Ltda., *Rio de Janeiro*

Contents

3 Output Primitives 83

4 Attributes of Output Primitives 143

7 Structures and Hierarchical Modeling 250

8 Graphical User Interfaces and Interactive Input Methods 271

9 Three-Dimensional Concepts 296

10 Three-Dimensional Object Representations 304

13 Visible-Surface Detection Methods 469

14 Illumination Models and Surface-Rendering Methods 494

15 Color Models and Color Applications 564

16 Computer Animation 583

A Mathematics for Computer Graphics 599

CHAPTER

2 Overview of Graphics Systems

Due to the widespread recognition of the power and utility of computer graphics in virtually all fields, a broad range of graphics hardware and software systems is now available. Graphics capabilities for both two-dimensional and three-dimensional applications are now common on general-purpose computers, including many hand-held calculators. With personal computers, we can use a wide variety of interactive input devices and graphics software packages. For higher-quality applications, we can choose from a number of sophisticated special-purpose graphics hardware systems and technologies. In this chapter, we explore the basic features of graphics hardware components and graphics software packages.

2-1 VIDEO DISPLAY DEVICES

Typically, the primary output device in a graphics system is a video monitor (Fig. 2-1). The operation of most video monitors is based on the standard **cathode-ray tube (CRT)** design, but several other technologies exist and solid-state monitors may eventually predominate.

Figure 2-1
A computer graphics workstation. (*Courtesy of Tektronix, Inc.*)

Refresh Cathode-Ray Tubes

Figure 2-2 illustrates the basic operation of a CRT. A beam of electrons (*cathode rays*), emitted by an electron gun, passes through focusing and deflection systems that direct the beam toward specified positions on the phosphor-coated screen. The phosphor then emits a small spot of light at each position contacted by the electron beam. Because the light emitted by the phosphor fades very rapidly, some method is needed for maintaining the screen picture. One way to keep the phosphor glowing is to redraw the picture repeatedly by quickly directing the electron beam back over the same points. This type of display is called a **refresh CRT**.

The primary components of an electron gun in a CRT are the heated metal cathode and a control grid (Fig. 2-3). Heat is supplied to the cathode by directing a current through a coil of wire, called the filament, inside the cylindrical cathode structure. This causes electrons to be "boiled off" the hot cathode surface. In the vacuum inside the CRT envelope, the free, negatively charged electrons are then accelerated toward the phosphor coating by a high positive voltage. The acceler-

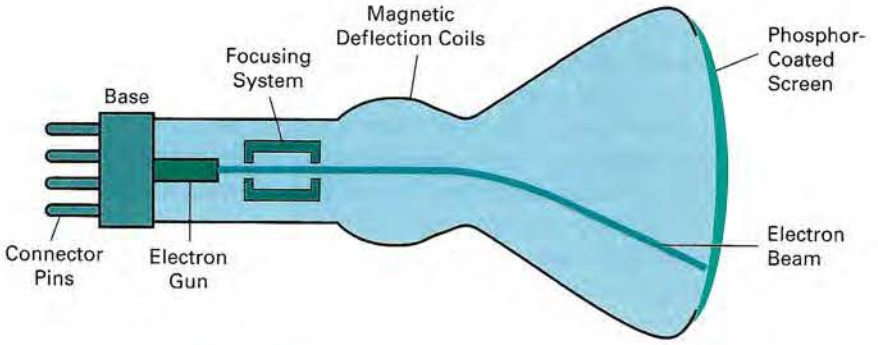


Figure 2-2
Basic design of a magnetic-deflection CRT.

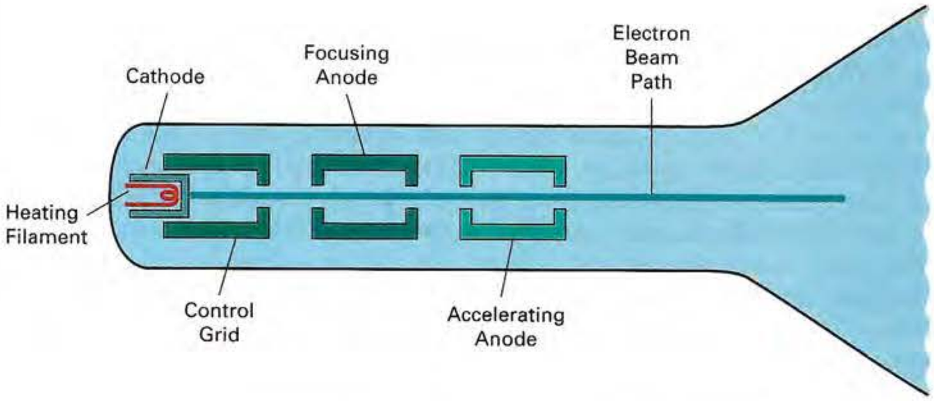


Figure 2-3
Operation of an electron gun with an accelerating anode.

ating voltage can be generated with a positively charged metal coating on the inside of the CRT envelope near the phosphor screen, or an accelerating anode can be used, as in Fig. 2-3. Sometimes the electron gun is built to contain the accelerating anode and focusing system within the same unit.

Intensity of the electron beam is controlled by setting voltage levels on the control grid, which is a metal cylinder that fits over the cathode. A high negative voltage applied to the control grid will shut off the beam by repelling electrons and stopping them from passing through the small hole at the end of the control grid structure. A smaller negative voltage on the control grid simply decreases the number of electrons passing through. Since the amount of light emitted by the phosphor coating depends on the number of electrons striking the screen, we control the brightness of a display by varying the voltage on the control grid. We specify the intensity level for individual screen positions with graphics software commands, as discussed in Chapter 3.

The focusing system in a CRT is needed to force the electron beam to converge into a small spot as it strikes the phosphor. Otherwise, the electrons would repel each other, and the beam would spread out as it approaches the screen. Focusing is accomplished with either electric or magnetic fields. Electrostatic focusing is commonly used in television and computer graphics monitors. With electrostatic focusing, the electron beam passes through a positively charged metal cylinder that forms an electrostatic lens, as shown in Fig. 2-3. The action of the electrostatic lens focuses the electron beam at the center of the screen, in exactly the same way that an optical lens focuses a beam of light at a particular focal distance. Similar lens focusing effects can be accomplished with a magnetic field set up by a coil mounted around the outside of the CRT envelope. Magnetic lens focusing produces the smallest spot size on the screen and is used in special-purpose devices.

Additional focusing hardware is used in high-precision systems to keep the beam in focus at all screen positions. The distance that the electron beam must travel to different points on the screen varies because the radius of curvature for most CRTs is greater than the distance from the focusing system to the screen center. Therefore, the electron beam will be focused properly only at the center of the screen. As the beam moves to the outer edges of the screen, displayed images become blurred. To compensate for this, the system can adjust the focusing according to the screen position of the beam.

As with focusing, deflection of the electron beam can be controlled either with electric fields or with magnetic fields. Cathode-ray tubes are now commonly constructed with magnetic deflection coils mounted on the outside of the CRT envelope, as illustrated in Fig. 2-2. Two pairs of coils are used, with the coils in each pair mounted on opposite sides of the neck of the CRT envelope. One pair is mounted on the top and bottom of the neck, and the other pair is mounted on opposite sides of the neck. The magnetic field produced by each pair of coils results in a transverse deflection force that is perpendicular both to the direction of the magnetic field and to the direction of travel of the electron beam. Horizontal deflection is accomplished with one pair of coils, and vertical deflection by the other pair. The proper deflection amounts are attained by adjusting the current through the coils. When electrostatic deflection is used, two pairs of parallel plates are mounted inside the CRT envelope. One pair of plates is mounted horizontally to control the vertical deflection, and the other pair is mounted vertically to control horizontal deflection (Fig. 2-4).

Spots of light are produced on the screen by the transfer of the CRT beam energy to the phosphor. When the electrons in the beam collide with the phos-

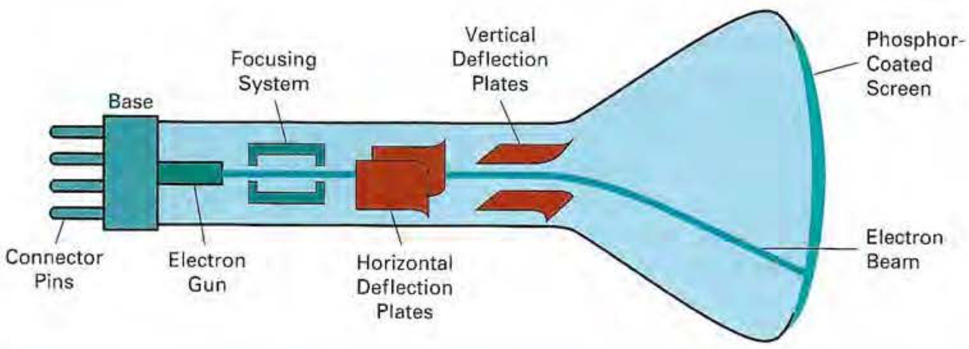


Figure 2-4
Electrostatic deflection of the electron beam in a CRT.

phor coating, they are stopped and their kinetic energy is absorbed by the phosphor. Part of the beam energy is converted by friction into heat energy, and the remainder causes electrons in the phosphor atoms to move up to higher quantum-energy levels. After a short time, the "excited" phosphor electrons begin dropping back to their stable ground state, giving up their extra energy as small quantums of light energy. What we see on the screen is the combined effect of all the electron light emissions: a glowing spot that quickly fades after all the excited phosphor electrons have returned to their ground energy level. The frequency (or color) of the light emitted by the phosphor is proportional to the energy difference between the excited quantum state and the ground state.

Different kinds of phosphors are available for use in a CRT. Besides color, a major difference between phosphors is their **persistence**: how long they continue to emit light (that is, have excited electrons returning to the ground state) after the CRT beam is removed. Persistence is defined as the time it takes the emitted light from the screen to decay to one-tenth of its original intensity. Lower-persistence phosphors require higher refresh rates to maintain a picture on the screen without flicker. A phosphor with low persistence is useful for animation; a high-persistence phosphor is useful for displaying highly complex, static pictures. Although some phosphors have a persistence greater than 1 second, graphics monitors are usually constructed with a persistence in the range from 10 to 60 microseconds.

Figure 2-5 shows the intensity distribution of a spot on the screen. The intensity is greatest at the center of the spot, and decreases with a Gaussian distribution out to the edges of the spot. This distribution corresponds to the cross-sectional electron density distribution of the CRT beam.

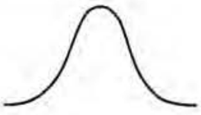


Figure 2-5
Intensity distribution of an illuminated phosphor spot on a CRT screen.

The maximum number of points that can be displayed without overlap on a CRT is referred to as the **resolution**. A more precise definition of resolution is the number of points per centimeter that can be plotted horizontally and vertically, although it is often simply stated as the total number of points in each direction. Spot intensity has a Gaussian distribution (Fig. 2-5), so two adjacent spots will appear distinct as long as their separation is greater than the diameter at which each spot has an intensity of about 60 percent of that at the center of the spot. This overlap position is illustrated in Fig. 2-6. Spot size also depends on intensity. As more electrons are accelerated toward the phospher per second, the CRT beam diameter and the illuminated spot increase. In addition, the increased excitation energy tends to spread to neighboring phosphor atoms not directly in the path of the beam, which further increases the spot diameter. Thus, resolution of a CRT is dependent on the type of phosphor, the intensity to be displayed, and the focusing and deflection systems. Typical resolution on high-quality systems is 1280 by 1024, with higher resolutions available on many systems. High-resolution systems are often referred to as *high-definition systems*. The physical size of a graphics monitor is given as the length of the screen diagonal, with sizes varying from about 12 inches to 27 inches or more. A CRT monitor can be attached to a variety of computer systems, so the number of screen points that can actually be plotted depends on the capabilities of the system to which it is attached.

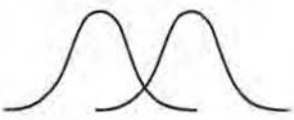


Figure 2-6
Two illuminated phosphor spots are distinguishable when their separation is greater than the diameter at which a spot intensity has fallen to 60 percent of maximum.

Another property of video monitors is **aspect ratio**. This number gives the ratio of vertical points to horizontal points necessary to produce equal-length lines in both directions on the screen. (Sometimes aspect ratio is stated in terms of the ratio of horizontal to vertical points.) An aspect ratio of 3/4 means that a vertical line plotted with three points has the same length as a horizontal line plotted with four points.

Raster-Scan Displays

The most common type of graphics monitor employing a CRT is the **raster-scan** display, based on television technology. In a raster-scan system, the electron beam is swept across the screen, one row at a time from top to bottom.. As the electron beam moves across each row, the beam intensity is turned on and off to create a pattern of illuminated spots. Picture definition is stored in a memory area called the **refresh buffer** or **frame buffer**. This memory area holds the set of intensity values for all the screen points. Stored intensity values are then retrieved from the refresh buffer and "painted" on the screen one row (**scan line**) at a time (Fig. 2-7). Each screen point is referred to as a **pixel** or **pel** (shortened forms of **picture element**). The capability of a raster-scan system to store intensity information for each screen point makes it well suited for the realistic display of scenes containing subtle shading and color patterns. Home television sets and printers are examples of other systems using raster-scan methods.

Intensity range for pixel positions depends on the capability of the raster system. In a simple black-and-white system, each screen point is either on or off, so only one bit per pixel is needed to control the intensity of screen positions. For a bilevel system, a bit value of 1 indicates that the electron beam is to be turned on at that position, and a value of 0 indicates that the beam intensity is to be off. Additional bits are needed when color and intensity variations can be displayed. Up to 24 bits per pixel are included in high-quality systems, which can require several megabytes of storage for the frame buffer, depending on the resolution of the system. A system with 24 bits per pixel and a screen resolution of 1024 by 1024 requires 3 megabytes of storage for the frame buffer. On a black-and-white system with one bit per pixel, the frame buffer is commonly called a **bitmap**. For systems with multiple bits per pixel, the frame buffer is often referred to as a **pixmap**.

Refreshing on raster-scan displays is carried out at the rate of 60 to 80 frames per second, although some systems are designed for higher refresh rates. Sometimes, refresh rates are described in units of cycles per second, or Hertz (Hz), where a cycle corresponds to one frame. Using these units, we would describe a refresh rate of 60 frames per second as simply 60 Hz. At the end of each scan line, the electron beam returns to the left side of the screen to begin displaying the next scan line. The return to the left of the screen, after refreshing each

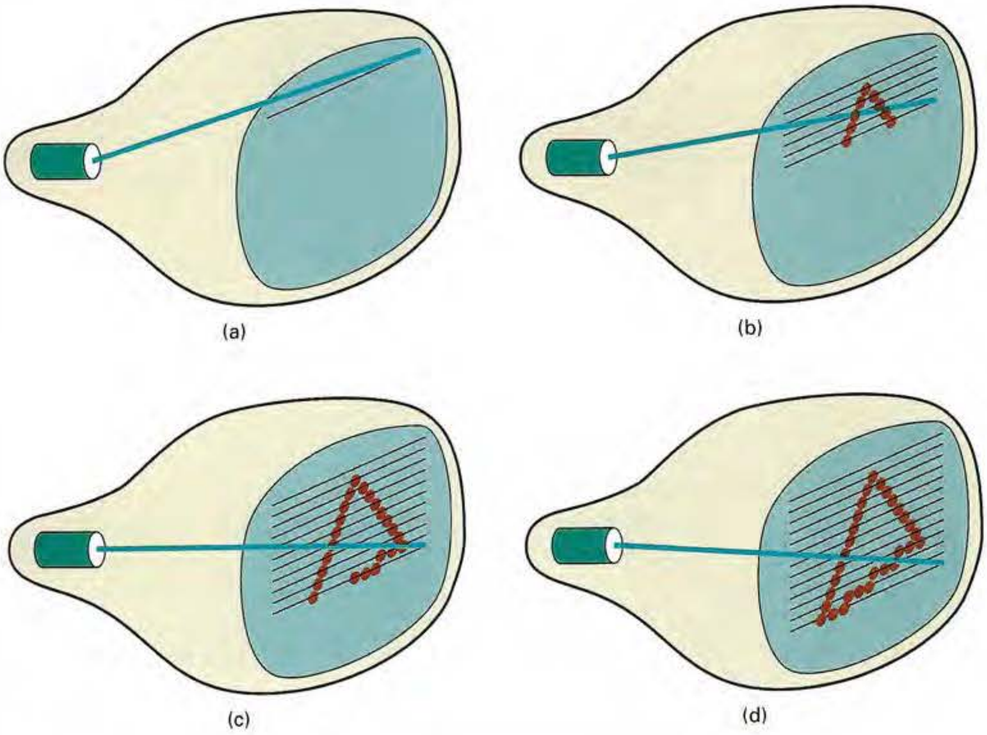


Figure 2-7
A raster-scan system displays an object as a set of discrete points across each scan line.

scan line, is called the **horizontal retrace** of the electron beam. And at the end of each frame (displayed in 1/80th to 1/60th of a second), the electron beam returns (**vertical retrace**) to the top left corner of the screen to begin the next frame.

On some raster-scan systems (and in TV sets), each frame is displayed in two passes using an *interlaced* refresh procedure. In the first pass, the beam sweeps across every other scan line from top to bottom. Then after the vertical retrace, the beam sweeps out the remaining scan lines (Fig. 2-8). Interlacing of the scan lines in this way allows us to see the entire screen displayed in one-half the time it would have taken to sweep across all the lines at once from top to bottom. Interlacing is primarily used with slower refreshing rates. On an older, 30 frame-per-second, noninterlaced display, for instance, some flicker is noticeable. But with interlacing, each of the two passes can be accomplished in 1/60th of a second, which brings the refresh rate nearer to 60 frames per second. This is an effective technique for avoiding flicker, providing that adjacent scan lines contain similar display information.

Random-Scan Displays

When operated as a **random-scan** display unit, a CRT has the electron beam directed only to the parts of the screen where a picture is to be drawn. Random-scan monitors draw a picture one line at a time and for this reason are also referred to as **vector** displays (or **stroke-writing** or **calligraphic** displays). The component lines of a picture can be drawn and refreshed by a random-scan sys-

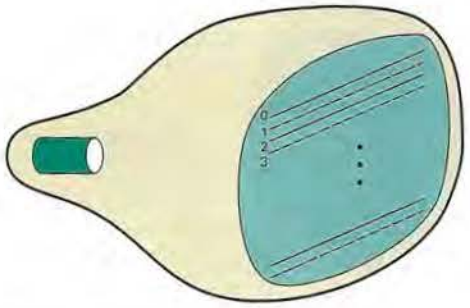


Figure 2-8
Interlacing scan lines on a raster-scan display. First, all points on the even-numbered (solid) scan lines are displayed; then all points along the odd-numbered (dashed) lines are displayed.

tem in any specified order (Fig. 2-9). A pen plotter operates in a similar way and is an example of a random-scan, hard-copy device.

Refresh rate on a random-scan system depends on the number of lines to be displayed. Picture definition is now stored as a set of line-drawing commands in an area of memory referred to as the **refresh display file**. Sometimes the refresh display file is called the **display list**, **display program**, or simply the **refresh buffer**. To display a specified picture, the system cycles through the set of commands in the display file, drawing each component line in turn. After all line-drawing commands have been processed, the system cycles back to the first line command in the list. Random-scan displays are designed to draw all the component lines of a picture 30 to 60 times each second. High-quality vector systems are capable of handling approximately 100,000 "short" lines at this refresh rate. When a small set of lines is to be displayed, each refresh cycle is delayed to avoid refresh rates greater than 60 frames per second. Otherwise, faster refreshing of the set of lines could burn out the phosphor.

Random-scan systems are designed for line-drawing applications and cannot display realistic shaded scenes. Since picture definition is stored as a set of line-drawing instructions and not as a set of intensity values for all screen points, vector displays generally have higher resolution than raster systems. Also, vector displays produce smooth line drawings because the CRT beam directly follows the line path. A raster system, in contrast, produces jagged lines that are plotted as discrete point sets.

Color CRT Monitors

A CRT monitor displays color pictures by using a combination of phosphors that emit different-colored light. By combining the emitted light from the different phosphors, a range of colors can be generated. The two basic techniques for producing color displays with a CRT are the beam-penetration method and the shadow-mask method.

The **beam-penetration** method for displaying color pictures has been used with random-scan monitors. Two layers of phosphor, usually red and green, are

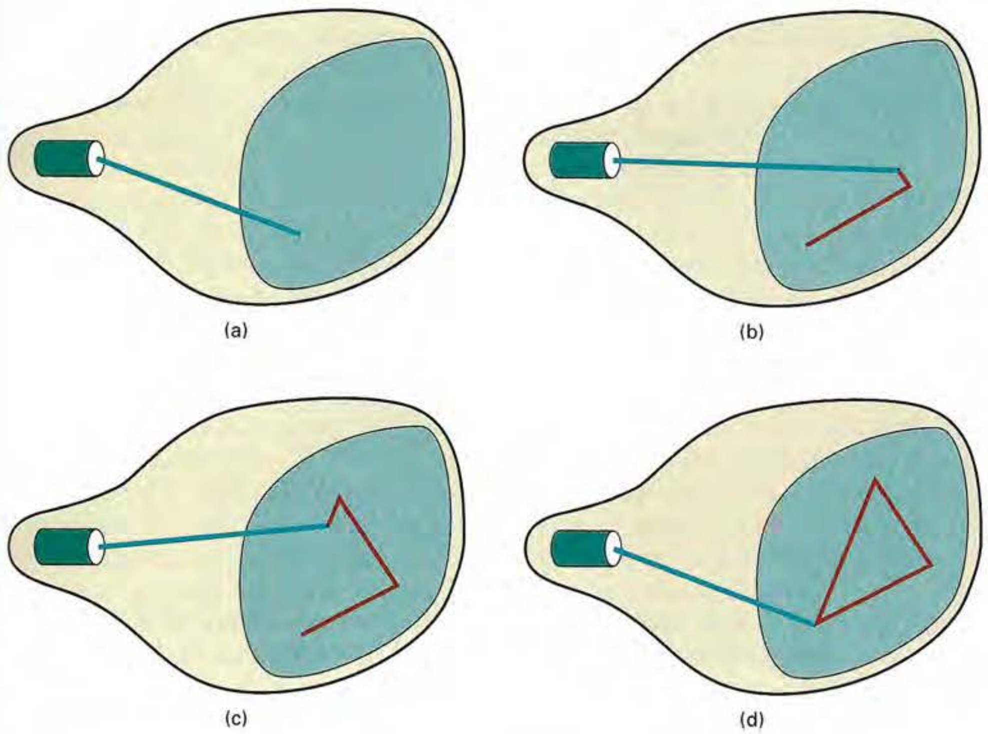


Figure 2-9
A random-scan system draws the component lines of an object in any order specified.

coated onto the inside of the CRT screen, and the displayed color depends on how far the electron beam penetrates into the phosphor layers. A beam of slow electrons excites only the outer red layer. A beam of very fast electrons penetrates through the red layer and excites the inner green layer. At intermediate beam speeds, combinations of red and green light are emitted to show two additional colors, orange and yellow. The speed of the electrons, and hence the screen color at any point, is controlled by the beam-acceleration voltage. Beam penetration has been an inexpensive way to produce color in random-scan monitors, but only four colors are possible, and the quality of pictures is not as good as with other methods.

Shadow-mask methods are commonly used in raster-scan systems (including color TV) because they produce a much wider range of colors than the beam-penetration method. A shadow-mask CRT has three phosphor color dots at each pixel position. One phosphor dot emits a red light, another emits a green light, and the third emits a blue light. This type of CRT has three electron guns, one for each color dot, and a shadow-mask grid just behind the phosphor-coated screen. Figure 2-10 illustrates the *delta-delta* shadow-mask method, commonly used in color CRT systems. The three electron beams are deflected and focused as a group onto the shadow mask, which contains a series of holes aligned with the phosphor-dot patterns. When the three beams pass through a hole in the shadow mask, they activate a dot triangle, which appears as a small color spot on the screen. The phosphor dots in the triangles are arranged so that each electron beam can activate only its corresponding color dot when it passes through the

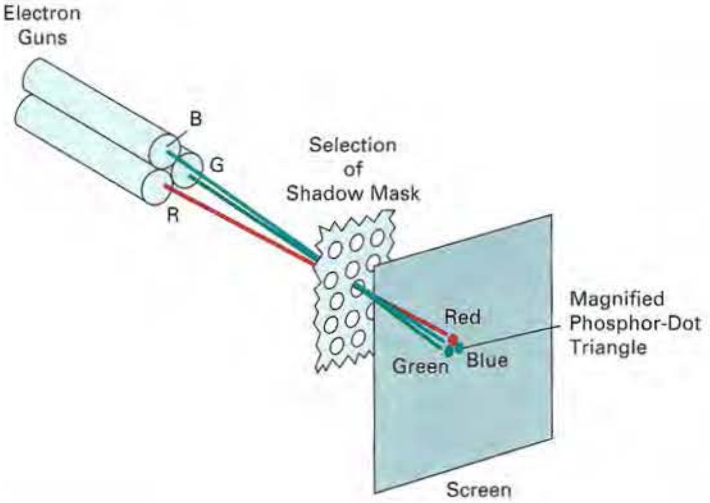


Figure 2-10
Operation of a delta-delta, shadow-mask CRT. Three electron guns, aligned with the triangular color-dot patterns on the screen, are directed to each dot triangle by a shadow mask.

shadow mask. Another configuration for the three electron guns is an *in-line* arrangement in which the three electron guns, and the corresponding red–green–blue color dots on the screen, are aligned along one scan line instead of in a triangular pattern. This in-line arrangement of electron guns is easier to keep in alignment and is commonly used in high-resolution color CRTs.

We obtain color variations in a shadow-mask CRT by varying the intensity levels of the three electron beams. By turning off the red and green guns, we get only the color coming from the blue phosphor. Other combinations of beam intensities produce a small light spot for each pixel position, since our eyes tend to merge the three colors into one composite. The color we see depends on the amount of excitation of the red, green, and blue phosphors. A white (or gray) area is the result of activating all three dots with equal intensity. Yellow is produced with the green and red dots only, magenta is produced with the blue and red dots, and cyan shows up when blue and green are activated equally. In some low-cost systems, the electron beam can only be set to on or off, limiting displays to eight colors. More sophisticated systems can set intermediate intensity levels for the electron beams, allowing several million different colors to be generated.

Color graphics systems can be designed to be used with several types of CRT display devices. Some inexpensive home-computer systems and video games are designed for use with a color TV set and an RF (radio-frequency) modulator. The purpose of the RF modulator is to simulate the signal from a broadcast TV station. This means that the color and intensity information of the picture must be combined and superimposed on the broadcast-frequency carrier signal that the TV needs to have as input. Then the circuitry in the TV takes this signal from the RF modulator, extracts the picture information, and paints it on the screen. As we might expect, this extra handling of the picture information by the RF modulator and TV circuitry decreases the quality of displayed images.

Composite monitors are adaptations of TV sets that allow bypass of the broadcast circuitry. These display devices still require that the picture information be combined, but no carrier signal is needed. Picture information is combined into a composite signal and then separated by the monitor, so the resulting picture quality is still not the best attainable.

Color CRTs in graphics systems are designed as **RGB monitors**. These monitors use shadow-mask methods and take the intensity level for each electron gun (red, green, and blue) directly from the computer system without any intermediate processing. High-quality raster-graphics systems have 24 bits per pixel in the frame buffer, allowing 256 voltage settings for each electron gun and nearly 17 million color choices for each pixel. An RGB color system with 24 bits of storage per pixel is generally referred to as a **full-color system** or a **true-color system**.

Direct-View Storage Tubes

An alternative method for maintaining a screen image is to store the picture information inside the CRT instead of refreshing the screen. A **direct-view storage tube (DVST)** stores the picture information as a charge distribution just behind the phosphor-coated screen. Two electron guns are used in a DVST. One, the primary gun, is used to store the picture pattern; the second, the flood gun, maintains the picture display.

A DVST monitor has both disadvantages and advantages compared to the refresh CRT. Because no refreshing is needed, very complex pictures can be displayed at very high resolutions without flicker. Disadvantages of DVST systems are that they ordinarily do not display color and that selected parts of a picture cannot be erased. To eliminate a picture section, the entire screen must be erased and the modified picture redrawn. The erasing and redrawing process can take several seconds for a complex picture. For these reasons, storage displays have been largely replaced by raster systems.

Flat-Panel Displays

Although most graphics monitors are still constructed with CRTs, other technologies are emerging that may soon replace CRT monitors. The term **flat-panel display** refers to a class of video devices that have reduced volume, weight, and power requirements compared to a CRT. A significant feature of flat-panel displays is that they are thinner than CRTs, and we can hang them on walls or wear them on our wrists. Since we can even write on some flat-panel displays, they will soon be available as pocket notepads. Current uses for flat-panel displays include small TV monitors, calculators, pocket video games, laptop computers, armrest viewing of movies on airlines, as advertisement boards in elevators, and as graphics displays in applications requiring rugged, portable monitors.

We can separate flat-panel displays into two categories: **emissive displays** and **nonemissive displays**. The emissive displays (or **emitters**) are devices that convert electrical energy into light. Plasma panels, thin-film electroluminescent displays, and light-emitting diodes are examples of emissive displays. Flat CRTs have also been devised, in which electron beams are accelerated parallel to the screen, then deflected 90° to the screen. But flat CRTs have not proved to be as successful as other emissive devices. Nonemmissive displays (or **nonemitters**) use optical effects to convert sunlight or light from some other source into graphics patterns. The most important example of a nonemissive flat-panel display is a liquid-crystal device.

Plasma panels, also called **gas-discharge displays**, are constructed by filling the region between two glass plates with a mixture of gases that usually includes neon. A series of vertical conducting ribbons is placed on one glass panel, and a set of horizontal ribbons is built into the other glass panel (Fig. 2-11). Firing voltages applied to a pair of horizontal and vertical conductors cause the gas at the intersection of the two conductors to break down into a glowing plasma of electrons and ions. Picture definition is stored in a refresh buffer, and the firing voltages are applied to refresh the pixel positions (at the intersections of the conductors) 60 times per second. Alternating-current methods are used to provide faster application of the firing voltages, and thus brighter displays. Separation between pixels is provided by the electric field of the conductors. Figure 2-12 shows a high-definition plasma panel. One disadvantage of plasma panels has been that they were strictly monochromatic devices, but systems have been developed that are now capable of displaying color and grayscale.

Thin-film electroluminescent displays are similar in construction to a plasma panel. The difference is that the region between the glass plates is filled with a phosphor, such as zinc sulfide doped with manganese, instead of a gas (Fig. 2-13). When a sufficiently high voltage is applied to a pair of crossing electrodes, the phosphor becomes a conductor in the area of the intersection of the two electrodes. Electrical energy is then absorbed by the manganese atoms, which then release the energy as a spot of light similar to the glowing plasma effect in a plasma panel. Electroluminescent displays require more power than plasma panels, and good color and gray scale displays are hard to achieve.

A third type of emissive device is the **light-emitting diode (LED)**. A matrix of diodes is arranged to form the pixel positions in the display, and picture definition is stored in a refresh buffer. As in scan-line refreshing of a CRT, information

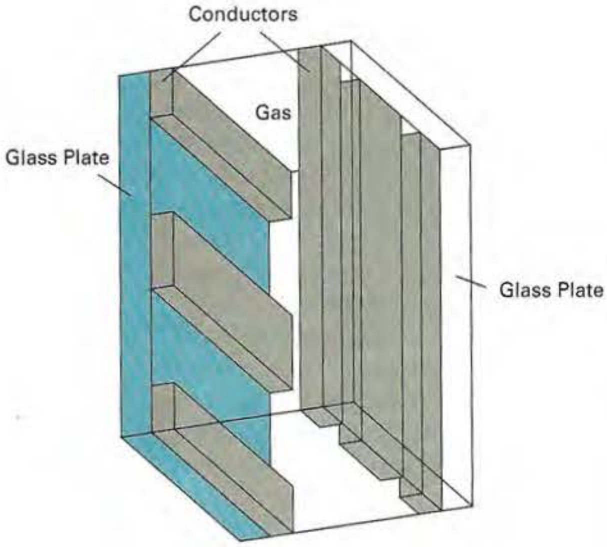


Figure 2-11
Basic design of a plasma-panel display device.

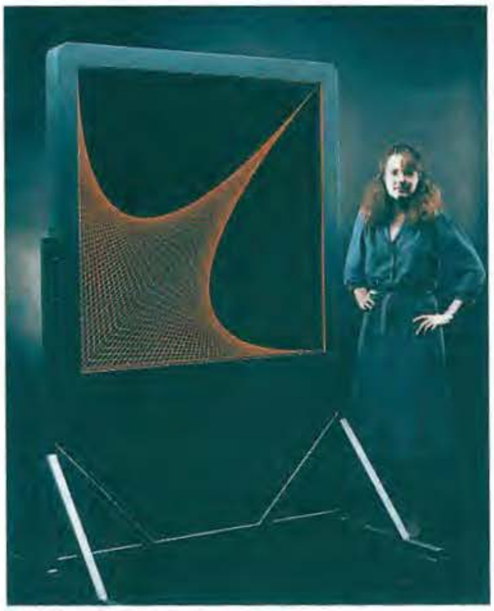


Figure 2-12
A plasma-panel display with a resolution of 2048 by 2048 and a screen diagonal of 1.5 meters.
(*Courtesy of Photonics Systems.*)

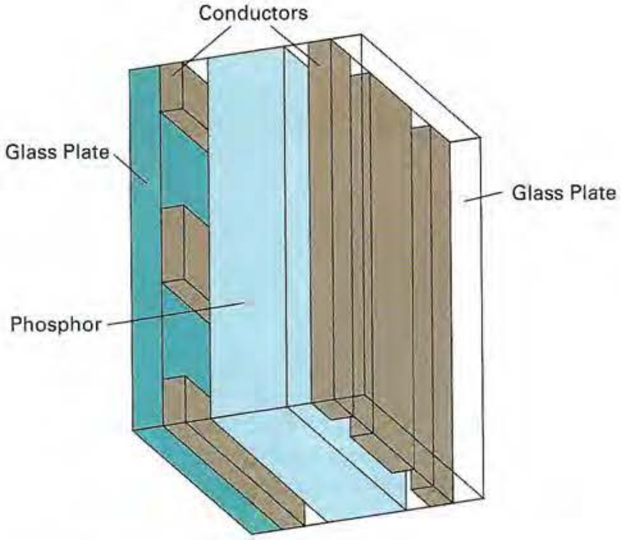


Figure 2-13
Basic design of a thin-film electroluminescent display device.

is read from the refresh buffer and converted to voltage levels that are applied to the diodes to produce the light patterns in the display.

Liquid-crystal displays (**LCD**s) are commonly used in small systems, such as calculators (Fig. 2-14) and portable, laptop computers (Fig. 2-15). These nonemissive devices produce a picture by passing polarized light from the surroundings or from an internal light source through a liquid-crystal material that can be aligned to either block or transmit the light.

The term *liquid crystal* refers to the fact that these compounds have a crystalline arrangement of molecules, yet they flow like a liquid. Flat-panel displays commonly use nematic (threadlike) liquid-crystal compounds that tend to keep the long axes of the rod-shaped molecules aligned. A flat-panel display can then be constructed with a nematic liquid crystal, as demonstrated in Fig. 2-16. Two glass plates, each containing a light polarizer at right angles to the other plate, sandwich the liquid-crystal material. Rows of horizontal transparent conductors are built into one glass plate, and columns of vertical conductors are put into the other plate. The intersection of two conductors defines a pixel position. Normally, the molecules are aligned as shown in the "on state" of Fig. 2-16. Polarized light passing through the material is twisted so that it will pass through the opposite polarizer. The light is then reflected back to the viewer. To turn off the pixel, we apply a voltage to the two intersecting conductors to align the molecules so that the light is not twisted. This type of flat-panel device is referred to as a **passive-matrix** LCD. Picture definitions are stored in a refresh buffer, and the screen is refreshed at the rate of 60 frames per second, as in the emissive devices. Back lighting is also commonly applied using solid-state electronic devices, so that the system is not completely dependent on outside light sources. Colors can be displayed by using different materials or dyes and by placing a triad of color pixels at each screen location. Another method for constructing LCDs is to place a transistor at each pixel location, using thin-film transistor technology. The transistors are used to control the voltage at pixel locations and to prevent charge from gradually leaking out of the liquid-crystal cells. These devices are called **active-matrix** displays.

Figure 2-14
A hand calculator with an LCD screen. (*Courtesy of Texas Instruments.*)

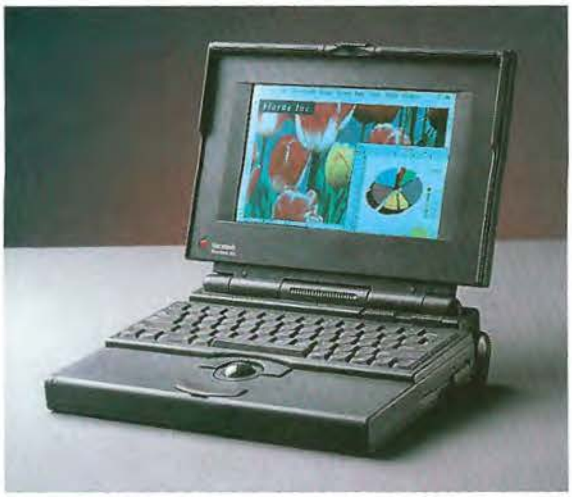


Figure 2-15
A backlit, passive-matrix, liquid-crystal display in a laptop computer, featuring 256 colors, a screen resolution of 640 by 400, and a screen diagonal of 9 inches.
(Courtesy of Apple Computer, Inc.)

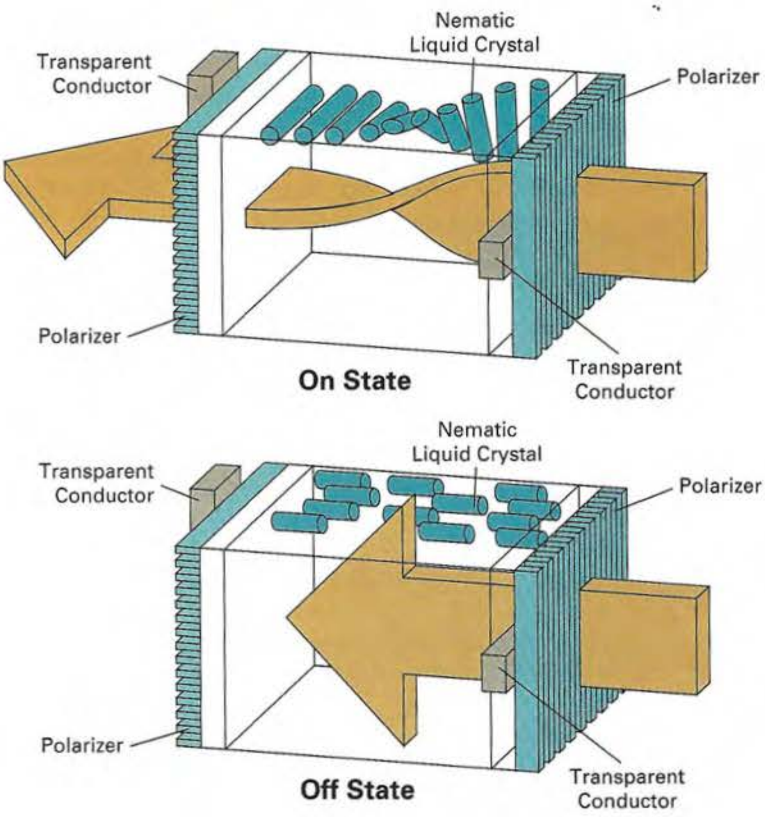


Figure 2-16
The light-twisting, shutter effect used in the design of most liquid-crystal display devices.

Three-Dimensional Viewing Devices

Graphics monitors for the display of three-dimensional scenes have been devised using a technique that reflects a CRT image from a vibrating, flexible mirror. The operation of such a system is demonstrated in Fig. 2-17. As the varifocal mirror vibrates, it changes focal length. These vibrations are synchronized with the display of an object on a CRT so that each point on the object is reflected from the mirror into a spatial position corresponding to the distance of that point from a specified viewing position. This allows us to walk around an object or scene and view it from different sides.

Figure 2-18 shows the Genisco SpaceGraph system, which uses a vibrating mirror to project three-dimensional objects into a 25-cm by 25-cm by 25-cm volume. This system is also capable of displaying two-dimensional cross-sectional "slices" of objects selected at different depths. Such systems have been used in medical applications to analyze data from ultrasonography and CAT scan devices, in geological applications to analyze topological and seismic data, in design applications involving solid objects, and in three-dimensional simulations of systems, such as molecules and terrain.

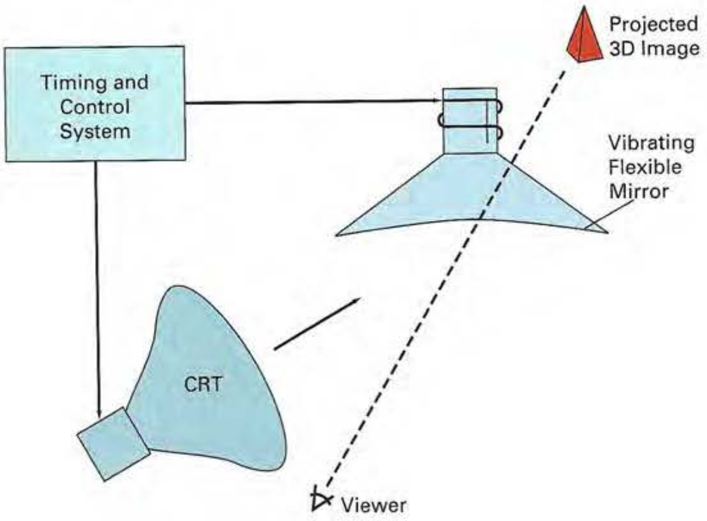


Figure 2-17
Operation of a three-dimensional display system using a vibrating mirror that changes focal length to match the depth of points in a scene.

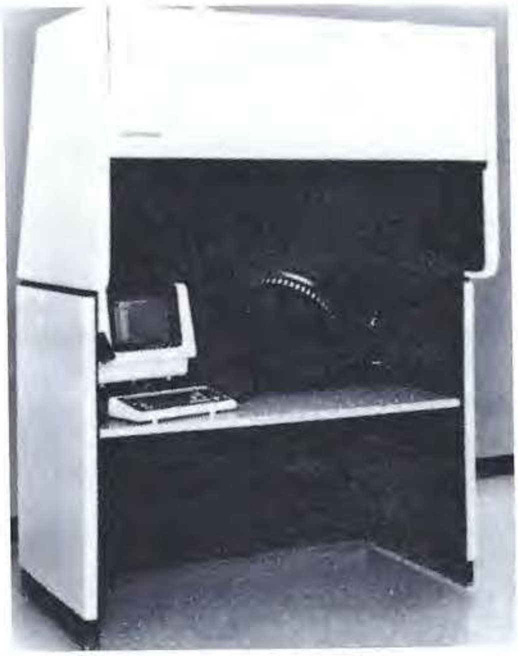


Figure 2-18
The SpaceGraph interactive graphics system displays objects in three dimensions using a vibrating, flexible mirror. (*Courtesy of Genisco Computers Corporation.*)

Stereoscopic and Virtual-Reality Systems

Another technique for representing three-dimensional objects is displaying stereoscopic views. This method does not produce true three-dimensional images, but it does provide a three-dimensional effect by presenting a different view to each eye of an observer so that scenes do appear to have depth (Fig. 2-19).

To obtain a stereoscopic projection, we first need to obtain two views of a scene generated from a viewing direction corresponding to each eye (left and right). We can construct the two views as computer-generated scenes with different viewing positions, or we can use a stereo camera pair to photograph some object or scene. When we simultaneous look at the left view with the left eye and the right view with the right eye, the two views merge into a single image and we perceive a scene with depth. Figure 2-20 shows two views of a computer-generated scene for stereographic projection. To increase viewing comfort, the areas at the left and right edges of this scene that are visible to only one eye have been eliminated.

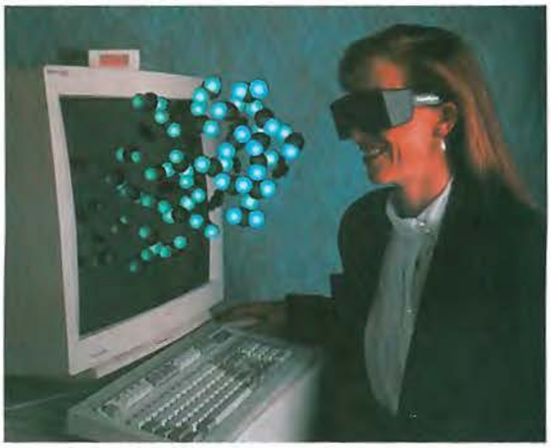
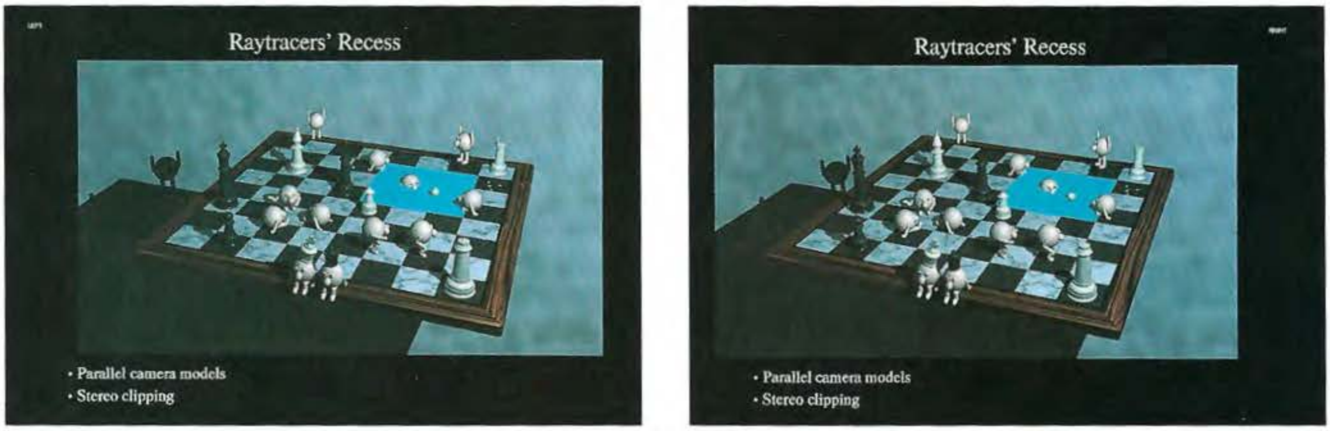


Figure 2-19
Viewing a stereoscopic projection.
(*Courtesy of StereoGraphics Corporation.*)



Left

Right

Figure 2-20
A stereoscopic viewing pair. (*Courtesy of Jerry Farm.*)

One way to produce a stereoscopic effect is to display each of the two views with a raster system on alternate refresh cycles. The screen is viewed through glasses, with each lens designed to act as a rapidly alternating shutter that is synchronized to block out one of the views. Figure 2-21 shows a pair of stereoscopic glasses constructed with liquid-crystal shutters and an infrared emitter that synchronizes the glasses with the views on the screen.

Stereoscopic viewing is also a component in **virtual-reality** systems, where users can step into a scene and interact with the environment. A headset (Fig. 2-22) containing an optical system to generate the stereoscopic views is commonly used in conjuction with interactive input devices to locate and manipulate objects in the scene. A sensing system in the headset keeps track of the viewer's position, so that the front and back of objects can be seen as the viewer

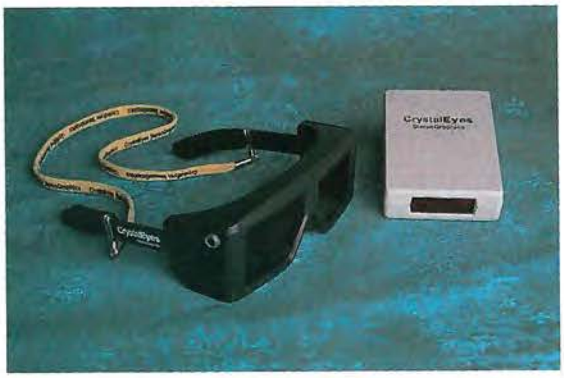


Figure 2-21
Glasses for viewing a stereoscopic scene and an infrared synchronizing emitter. *(Courtesy of StereoGraphics Corporation.)*

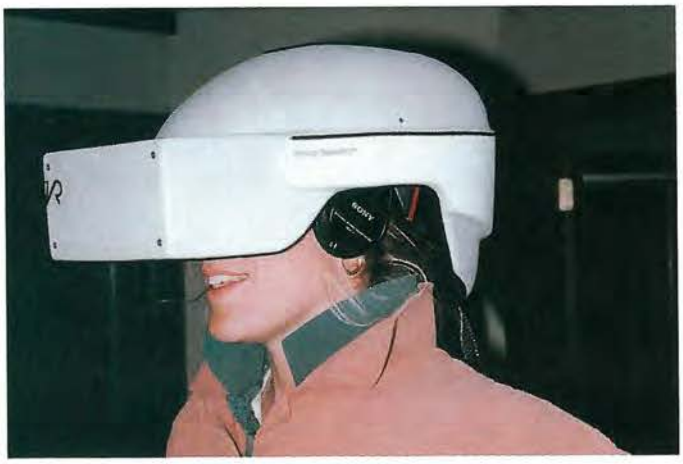


Figure 2-22
A headset used in virtual-reality systems. *(Courtesy of Virtual Research.)*

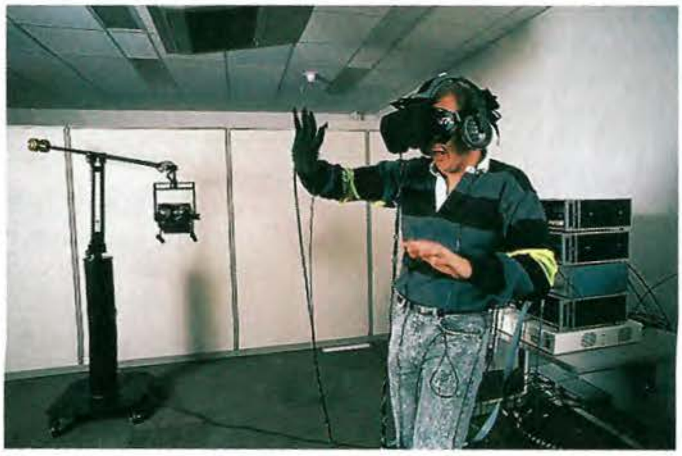


Figure 2-23
Interacting with a virtual-reality environment. (*Courtesy of the National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign.*)

"walks through" and interacts with the display. Figure 2-23 illustrates interaction with a virtual scene, using a headset and a data glove worn on the right hand (Section 2-5).

An interactive virtual-reality environment can also be viewed with stereoscopic glasses and a video monitor, instead of a headset. This provides a means for obtaining a lower-cost virtual-reality system. As an example, Fig. 2-24 shows an ultrasound tracking device with six degrees of freedom. The tracking device is placed on top of the video display and is used to monitor head movements so that the viewing position for a scene can be changed as head position changes.

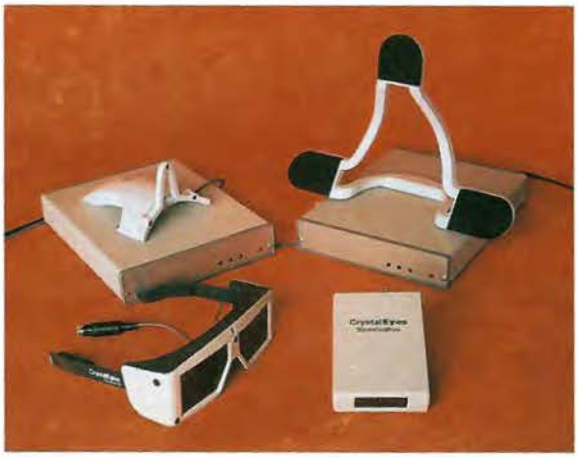


Figure 2-24
An ultrasound tracking device used with stereoscopic glasses to track head position. (*Courtesy of StereoGraphics Corporation.*)

2-2 RASTER-SCAN SYSTEMS

Interactive raster graphics systems typically employ several processing units. In addition to the central processing unit, or CPU, a special-purpose processor, called the **video controller** or **display controller**, is used to control the operation of the display device. Organization of a simple raster system is shown in Fig. 2-25. Here, the frame buffer can be anywhere in the system memory, and the video controller accesses the frame buffer to refresh the screen. In addition to the video controller, more sophisticated raster systems employ other processors as coprocessors and accelerators to implement various graphics operations.

Video Controller

Figure 2-26 shows a commonly used organization for raster systems. A fixed area of the system memory is reserved for the frame buffer, and the video controller is given direct access to the frame-buffer memory.

Frame-buffer locations, and the corresponding screen positions, are referenced in Cartesian coordinates. For many graphics monitors, the coordinate ori-

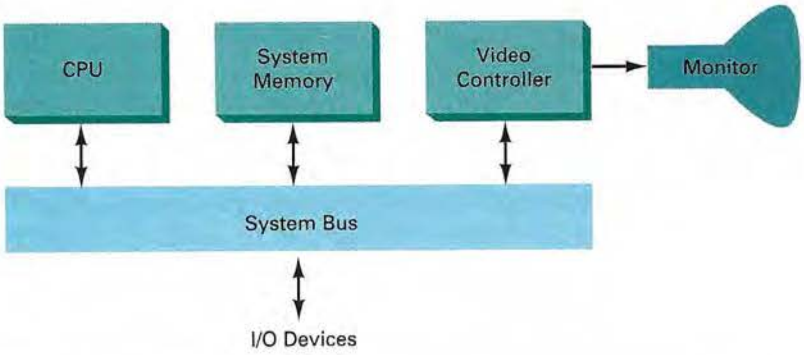


Figure 2-25
Architecture of a simple raster graphics system.

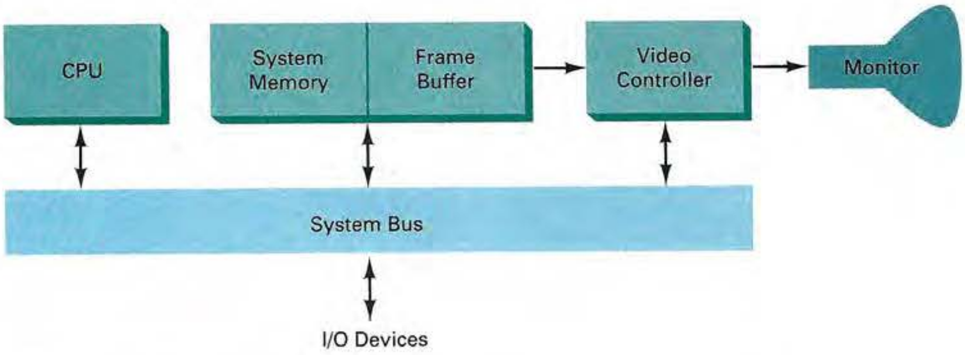


Figure 2-26
Architecture of a raster system with a fixed portion of the system memory reserved for the frame buffer.

gin is defined at the lower left screen corner (Fig. 2-27). The screen surface is then represented as the first quadrant of a two-dimensional system, with positive x values increasing to the right and positive y values increasing from bottom to top. (On some personal computers, the coordinate origin is referenced at the upper left corner of the screen, so the y values are inverted.) Scan lines are then labeled from y_{max} at the top of the screen to 0 at the bottom. Along each scan line, screen pixel positions are labeled from 0 to x_{max}.

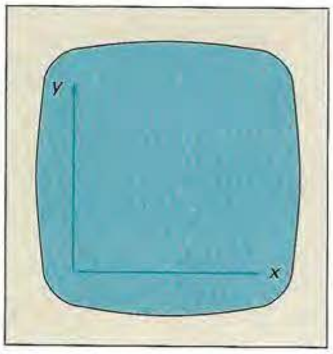


Figure 2-27
The origin of the coordinate system for identifying screen positions is usually specified in the lower-left corner.

In Fig. 2-28, the basic refresh operations of the video controller are diagrammed. Two registers are used to store the coordinates of the screen pixels. Initially, the x register is set to 0 and the y register is set to y_{max}. The value stored in the frame buffer for this pixel position is then retrieved and used to set the intensity of the CRT beam. Then the x register is incremented by 1, and the process repeated for the next pixel on the top scan line. This procedure is repeated for each pixel along the scan line. After the last pixel on the top scan line has been processed, the x register is reset to 0 and the y register is decremented by 1. Pixels along this scan line are then processed in turn, and the procedure is repeated for each successive scan line. After cycling through all pixels along the bottom scan line ($y = 0$), the video controller resets the registers to the first pixel position on the top scan line and the refresh process starts over.

Since the screen must be refreshed at the rate of 60 frames per second, the simple procedure illustrated in Fig. 2-28 cannot be accommodated by typical RAM chips. The cycle time is too slow. To speed up pixel processing, video controllers can retrieve multiple pixel values from the refresh buffer on each pass. The multiple pixel intensities are then stored in a separate register and used to control the CRT beam intensity for a group of adjacent pixels. When that group of pixels has been processed, the next block of pixel values is retrieved from the frame buffer.

A number of other operations can be performed by the video controller, besides the basic refreshing operations. For various applications, the video con-

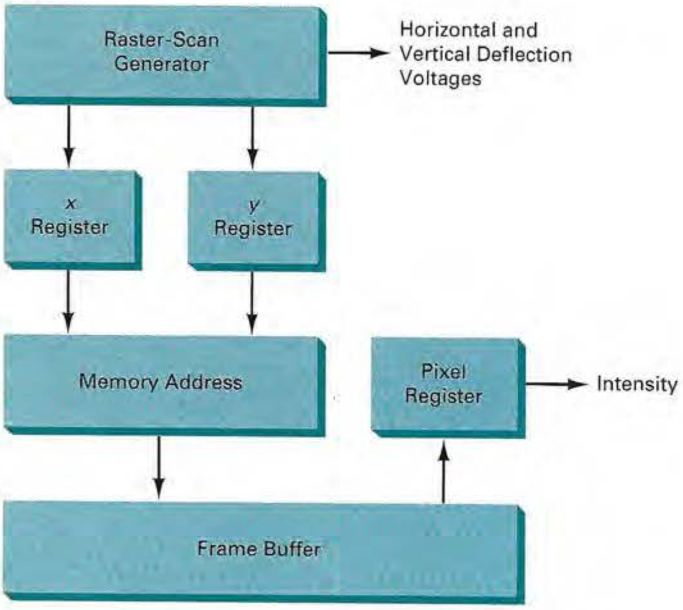


Figure 2-28
Basic video-controller refresh operations.

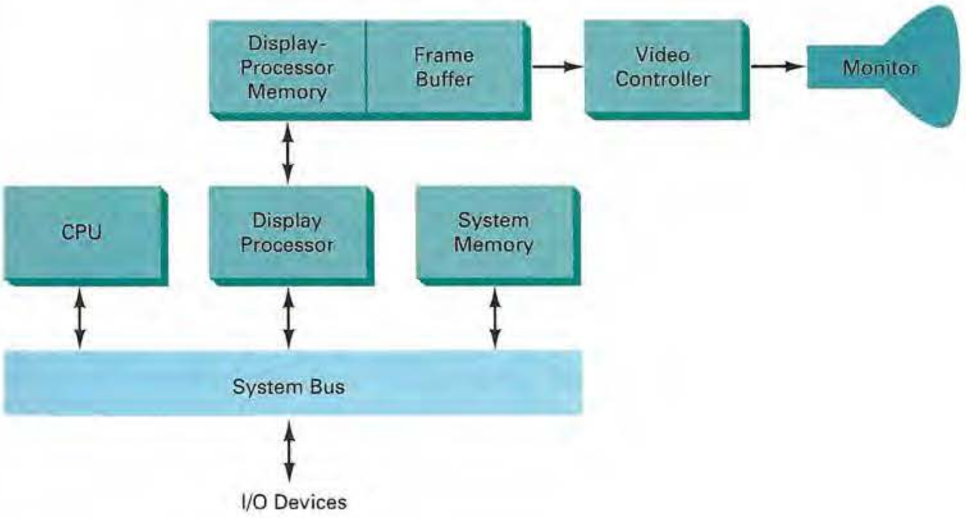


Figure 2-29
Architecture of a raster-graphics system with a display processor.

troller can retrieve pixel intensities from different memory areas on different refresh cycles. In high-quality systems, for example, two frame buffers are often provided so that one buffer can be used for refreshing while the other is being filled with intensity values. Then the two buffers can switch roles. This provides a fast mechanism for generating real-time animations, since different views of moving objects can be successively loaded into the refresh buffers. Also, some transformations can be accomplished by the video controller. Areas of the screen can be enlarged, reduced, or moved from one location to another during the refresh cycles. In addition, the video controller often contains a lookup table, so that pixel values in the frame buffer are used to access the lookup table instead of controlling the CRT beam intensity directly. This provides a fast method for changing screen intensity values, and we discuss lookup tables in more detail in Chapter 4. Finally, some systems are designed to allow the video controller to mix the frame-buffer image with an input image from a television camera or other input device.

Raster-Scan Display Processor

Figure 2-29 shows one way to set up the organization of a raster system containing a separate **display processor**, sometimes referred to as a **graphics controller** or a **display coprocessor.** The purpose of the display processor is to free the CPU from the graphics chores. In addition to the system memory, a separate display-processor memory area can also be provided.

A major task of the display processor is digitizing a picture definition given in an application program into a set of pixel-intensity values for storage in the frame buffer. This digitization process is called **scan conversion**. Graphics commands specifying straight lines and other geometric objects are scan converted into a set of discrete intensity points. Scan converting a straight-line segment, for example, means that we have to locate the pixel positions closest to the line path and store the intensity for each position in the frame buffer. Similar methods are used for scan converting curved lines and polygon outlines. Characters can be defined with rectangular grids, as in Fig. 2-30, or they can be defined with curved

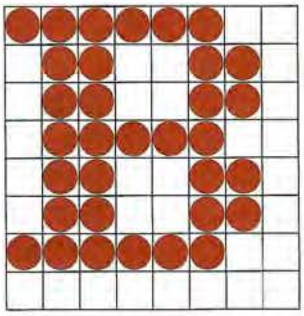


Figure 2-30
A character defined as a rectangular grid of pixel positions.

outlines, as in Fig. 2-31. The array size for character grids can vary from about 5 by 7 to 9 by 12 or more for higher-quality displays. A character grid is displayed by superimposing the rectangular grid pattern into the frame buffer at a specified coordinate position. With characters that are defined as curve outlines, character shapes are scan converted into the frame buffer.

Display processors are also designed to perform a number of additional operations. These functions include generating various line styles (dashed, dotted, or solid), displaying color areas, and performing certain transformations and manipulations on displayed objects. Also, display processors are typically designed to interface with interactive input devices, such as a mouse.

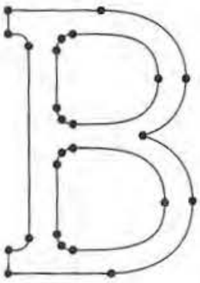


Figure 2-31
A character defined as a curve outline.

In an effort to reduce memory requirements in raster systems, methods have been devised for organizing the frame buffer as a linked list and encoding the intensity information. One way to do this is to store each scan line as a set of integer pairs. One number of each pair indicates an intensity value, and the second number specifies the number of adjacent pixels on the scan line that are to have that intensity. This technique, called **run-length encoding,** can result in a considerable saving in storage space if a picture is to be constructed mostly with long runs of a single color each. A similar approach can be taken when pixel intensities change linearly. Another approach is to encode the raster as a set of rectangular areas (**cell encoding**). The disadvantages of encoding runs are that intensity changes are difficult to make and storage requirements actually increase as the length of the runs decreases. In addition, it is difficult for the display controller to process the raster when many short runs are involved.

2-3 RANDOM-SCAN SYSTEMS

The organization of a simple random-scan (vector) system is shown in Fig. 2-32. An application program is input and stored in the system memory along with a graphics package. Graphics commands in the application program are translated by the graphics package into a display file stored in the system memory. This display file is then accessed by the display processor to refresh the screen. The display processor cycles through each command in the display file program once during every refresh cycle. Sometimes the display processor in a random-scan system is referred to as a **display processing unit** or a **graphics controller**.

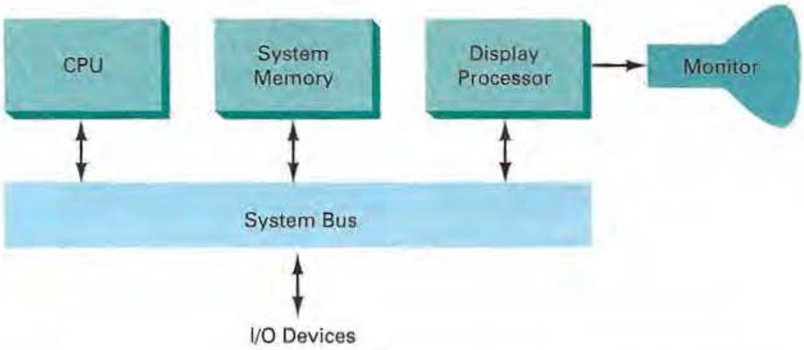


Figure 2-32
Architecture of a simple random-scan system.

Graphics patterns are drawn on a random-scan system by directing the electron beam along the component lines of the picture. Lines are defined by the values for their coordinate endpoints, and these input coordinate values are converted to x and y deflection voltages. A scene is then drawn one line at a time by positioning the beam to fill in the line between specified endpoints.

2-4 GRAPHICS MONITORS AND WORKSTATIONS

Most graphics monitors today operate as raster-scan displays, and here we survey a few of the many graphics hardware configurations available. Graphics systems range from small general-purpose computer systems with graphics capabilities (Fig. 2-33) to sophisticated full-color systems that are designed specifically for graphics applications (Fig. 2-34). A typical screen resolution for personal com-

Figure 2-33
A desktop general-purpose computer system that can be used for graphics applications. (*Courtesy of Apple Computer, Inc.*)

Figure 2-34
Computer graphics workstations with keyboard and mouse input devices. (a) The Iris Indigo. (*Courtesy of Silicon Graphics Corporation.*) (b) SPARCstation 10. (*Courtesy of Sun Microsystems.*)

puter systems, such as the Apple Quadra shown in Fig. 2-33, is 640 by 480, although screen resolution and other system capabilities vary depending on the size and cost of the system. Diagonal screen dimensions for general-purpose personal computer systems can range from 12 to 21 inches, and allowable color selections range from 16 to over 32,000. For workstations specifically designed for graphics applications, such as the systems shown in Fig. 2-34, typical screen resolution is 1280 by 1024, with a screen diagonal of 16 inches or more. Graphics workstations can be configured with from 8 to 24 bits per pixel (full-color systems), with higher screen resolutions, faster processors, and other options available in high-end systems.

Figure 2-35 shows a high-definition graphics monitor used in applications such as air traffic control, simulation, medical imaging, and CAD. This system has a diagonal screen size of 27 inches, resolutions ranging from 2048 by 1536 to 2560 by 2048, with refresh rates of 80 Hz or 60 Hz noninterlaced.

A multiscreen system called the MediaWall, shown in Fig. 2-36, provides a large "wall-sized" display area. This system is designed for applications that require large area displays in brightly lighted environments, such as at trade shows, conventions, retail stores, museums, or passenger terminals. MediaWall operates by splitting images into a number of sections and distributing the sections over an array of monitors or projectors using a graphics adapter and satellite control units. An array of up to 5 by 5 monitors, each with a resolution of 640 by 480, can be used in the MediaWall to provide an overall resolution of 3200 by 2400 for either static scenes or animations. Scenes can be displayed behind mullions, as in Fig. 2-36, or the mullions can be eliminated to display a continuous picture with no breaks between the various sections.

Many graphics workstations, such as some of those shown in Fig. 2-37, are configured with two monitors. One monitor can be used to show all features of an object or scene, while the second monitor displays the detail in some part of the picture. Another use for dual-monitor systems is to view a picture on one monitor and display graphics options (menus) for manipulating the picture components on the other monitor.

Figure 2-35
A very high-resolution (2560 by 2048) color monitor. *(Courtesy of BARCO Chromatics.)*

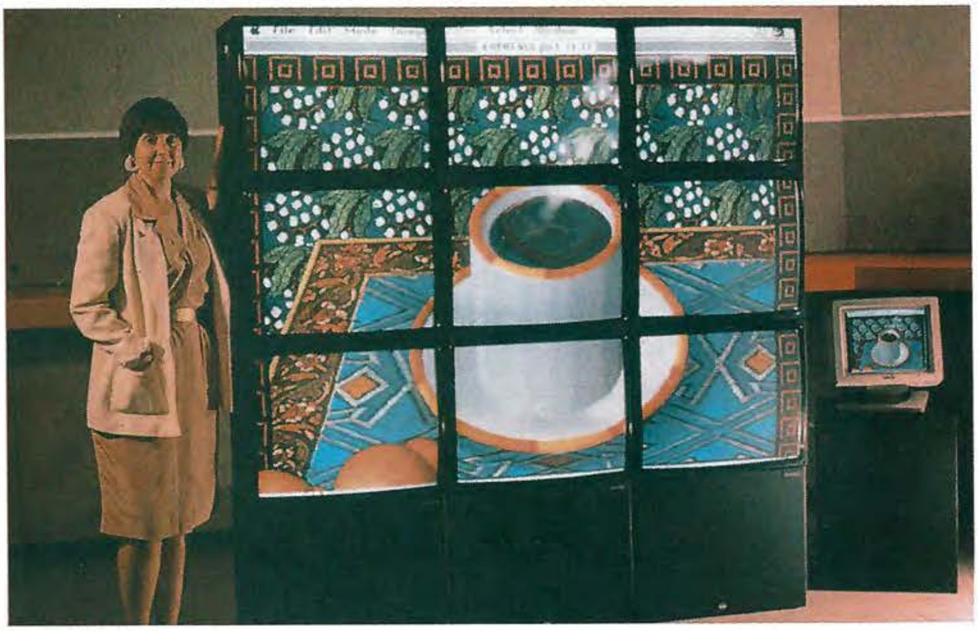


Figure 2-36
The MediaWall: A multiscreen display system. The image displayed on this 3-by-3 array of monitors was created by Deneba Software. (*Courtesy of RGB Spectrum.*)

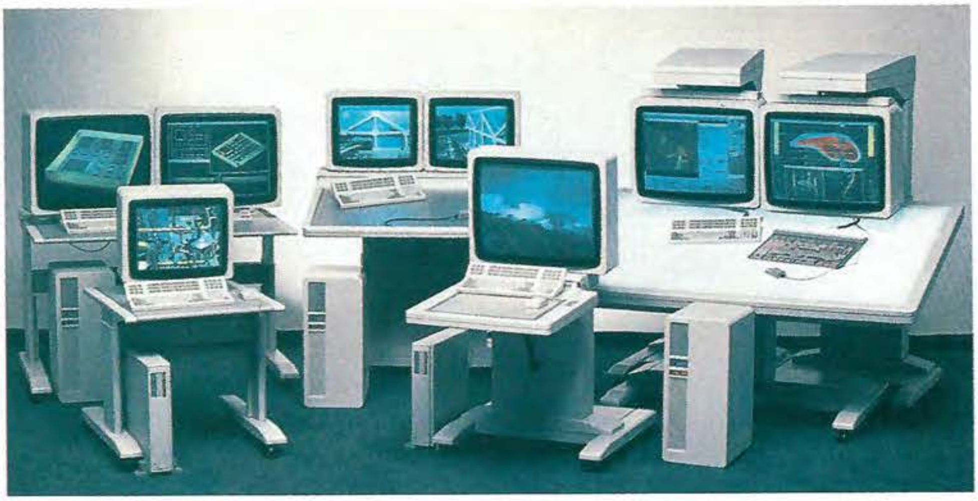


Figure 2-37
Single- and dual-monitor graphics workstations. (*Courtesy of Intergraph Corporation.*)

Figures 2-38 and 2-39 illustrate examples of interactive graphics workstations containing multiple input and other devices. A typical setup for CAD applications is shown in Fig. 2-38. Various keyboards, button boxes, tablets, and mice are attached to the video monitors for use in the design process. Figure 2-39 shows features of some types of artist's workstations.

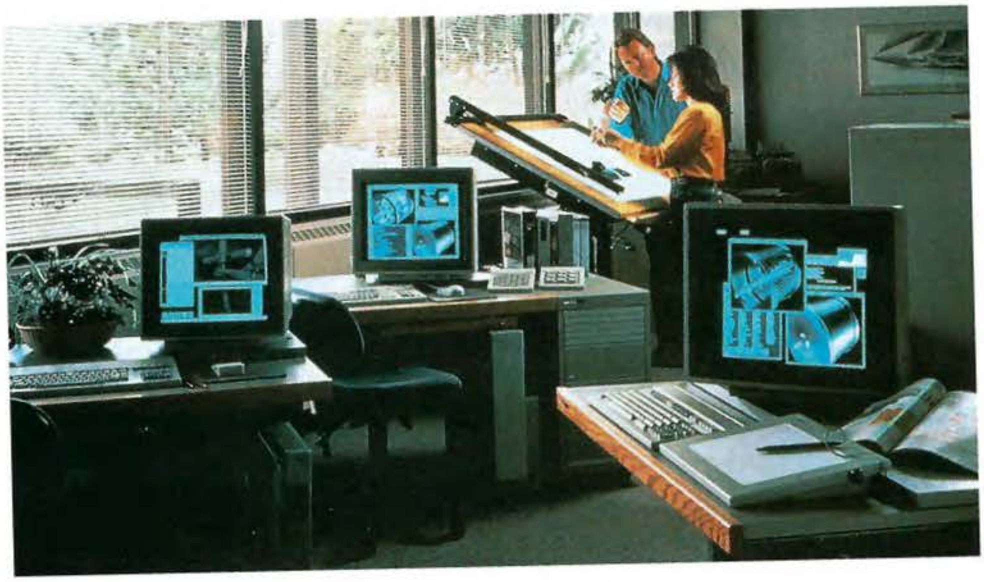


Figure 2-38
Multiple workstations for a CAD group. *(Courtesy of Hewlett-Packard Company.)*

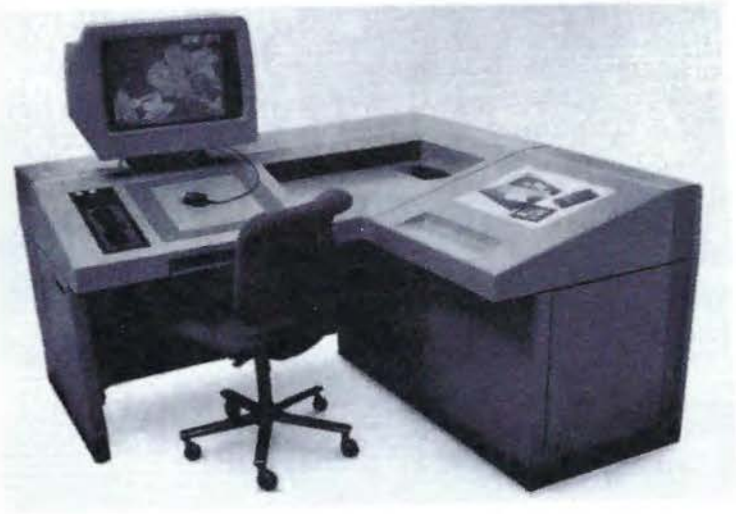


Figure 2-39
An artist's workstation, featuring a color raster monitor, keyboard, graphics tablet with hand cursor, and a light table, in addition to data storage and telecommunications devices. *(Courtesy of DICOMED Corporation.)*

2-5 INPUT DEVICES

Various devices are available for data input on graphics workstations. Most systems have a keyboard and one or more additional devices specially designed for interactive input. These include a mouse, trackball, spaceball, joystick, digitizers, dials, and button boxes. Some other input devices used in particular applications are data gloves, touch panels, image scanners, and voice systems.

Keyboards

An alphanumeric keyboard on a graphics system is used primarily as a device for entering text strings. The keyboard is an efficient device for inputting such nongraphic data as picture labels associated with a graphics display. Keyboards can also be provided with features to facilitate entry of screen coordinates, menu selections, or graphics functions.

Cursor-control keys and function keys are common features on general-purpose keyboards. Function keys allow users to enter frequently used operations in a single keystroke, and cursor-control keys can be used to select displayed objects or coordinate positions by positioning the screen cursor. Other types of cursor-positioning devices, such as a trackball or joystick, are included on some keyboards. Additionally, a numeric keypad is often included on the keyboard for fast entry of numeric data. Typical examples of general-purpose keyboards are given in Figs. 2-1, 2-33, and 2-34. Fig. 2-40 shows an ergonomic keyboard design.

For specialized applications, input to a graphics application may come from a set of buttons, dials, or switches that select data values or customized graphics operations. Figure 2-41 gives an example of a button box and a set of input dials. Buttons and switches are often used to input predefined functions, and dials are common devices for entering scalar values. Real numbers within some defined range are selected for input with dial rotations. Potentiometers are used to measure dial rotations, which are then converted to deflection voltages for cursor movement.

Mouse

A **mouse** is small hand-held box used to position the screen cursor. Wheels or rollers on the bottom of the mouse can be used to record the amount and direc-

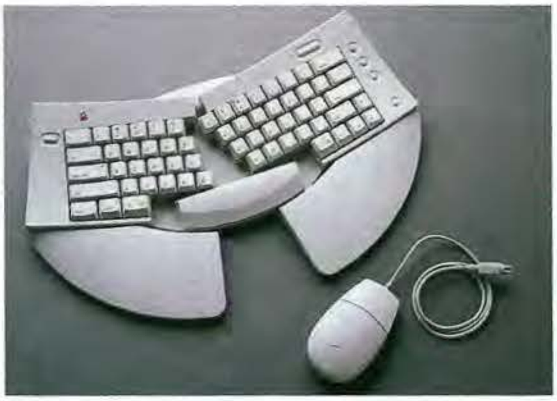


Figure 2-40
Ergonomically designed keyboard with removable palm rests. The slope of each half of the keyboard can be adjusted separately. (*Courtesy of Apple Computer, Inc.*)

tion of movement. Another method for detecting mouse motion is with an optical sensor. For these systems, the mouse is moved over a special mouse pad that has a grid of horizontal and vertical lines. The optical sensor detects movement across the lines in the grid.

Since a mouse can be picked up and put down at another position without change in cursor movement, it is used for making relative changes in the position of the screen cursor. One, two, or three buttons are usually included on the top of the mouse for signaling the execution of some operation, such as recording cursor position or invoking a function. Most general-purpose graphics systems now include a mouse and a keyboard as the major input devices, as in Figs. 2-1, 2-33, and 2-34.

Additional devices can be included in the basic mouse design to increase the number of allowable input parameters. The Z mouse in Fig. 2-42 includes

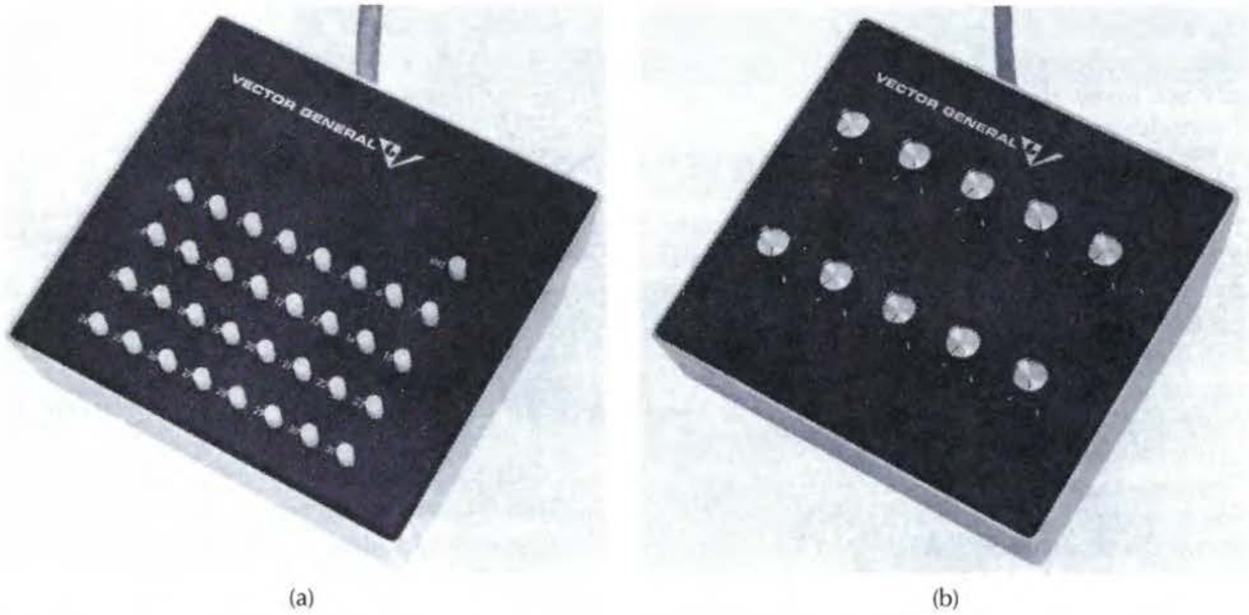


Figure 2-41
A button box (a) and a set of input dials (b). *(Courtesy of Vector General.)*

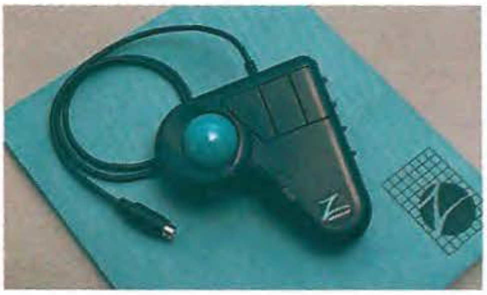


Figure 2-42
The Z mouse features three buttons, a mouse ball underneath, a thumbwheel on the side, and a trackball on top. *(Courtesy of Multipoint Technology Corporation.)*

three buttons, a thumbwheel on the side, a trackball on the top, and a standard mouse ball underneath. This design provides six degrees of freedom to select spatial positions, rotations, and other parameters. With the Z mouse, we can pick up an object, rotate it, and move it in any direction, or we can navigate our viewing position and orientation through a three-dimensional scene. Applications of the Z mouse include virtual reality, CAD, and animation.

Trackball and Spaceball

As the name implies, a **trackball** is a ball that can be rotated with the fingers or palm of the hand, as in Fig. 2-43, to produce screen-cursor movement. Potentiometers, attached to the ball, measure the amount and direction of rotation. Trackballs are often mounted on keyboards (Fig. 2-15) or other devices such as the Z mouse (Fig. 2-42).

While a trackball is a two-dimensional positioning device, a **spaceball** (Fig. 2-45) provides six degrees of freedom. Unlike the trackball, a spaceball does not actually move. Strain gauges measure the amount of pressure applied to the spaceball to provide input for spatial positioning and orientation as the ball is pushed or pulled in various directions. Spaceballs are used for three-dimensional positioning and selection operations in virtual-reality systems, modeling, animation, CAD, and other applications.

Joysticks

A **joystick** consists of a small, vertical lever (called the stick) mounted on a base that is used to steer the screen cursor around. Most joysticks select screen positions with actual stick movement; others respond to pressure on the stick. Figure 2-44 shows a movable joystick. Some joysticks are mounted on a keyboard; others function as stand-alone units.

The distance that the stick is moved in any direction from its center position corresponds to screen-cursor movement in that direction. Potentiometers mounted at the base of the joystick measure the amount of movement, and springs return the stick to the center position when it is released. One or more buttons can be programmed to act as input switchs to signal certain actions once a screen position has been selected.

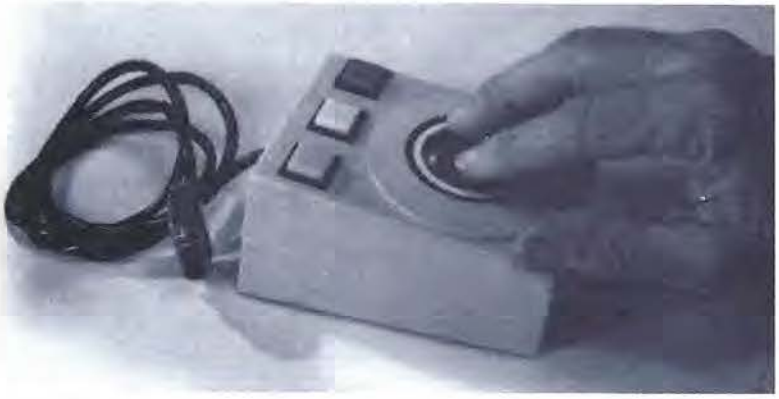


Figure 2-43
A three-button track ball. (*Courtesy of Measurement Systems Inc., Norwalk, Connecticut.*)

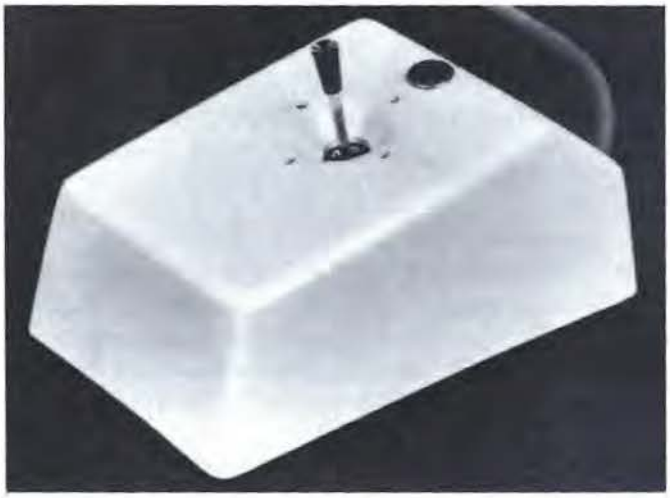


Figure 2-44
A moveable joystick. (*Courtesy of CalComp Group; Sanders Associates, Inc.*)

In another type of movable joystick, the stick is used to activate switches that cause the screen cursor to move at a constant rate in the direction selected. Eight switches, arranged in a circle, are sometimes provided, so that the stick can select any one of eight directions for cursor movement. Pressure-sensitive joysticks, also called isometric joysticks, have a nonmovable stick. Pressure on the stick is measured with strain gauges and converted to movement of the cursor in the direction specified.

Data Glove

Figure 2-45 shows a **data glove** that can be used to grasp a "virtual" object. The glove is constructed with a series of sensors that detect hand and finger motions. Electromagnetic coupling between transmitting antennas and receiving antennas is used to provide information about the position and orientation of the hand. The transmitting and receiving antennas can each be structured as a set of three mutually perpendicular coils, forming a three-dimensional Cartesian coordinate system. Input from the glove can be used to position or manipulate objects in a virtual scene. A two-dimensional projection of the scene can be viewed on a video monitor, or a three-dimensional projection can be viewed with a headset.

Digitizers

A common device for drawing, painting, or interactively selecting coordinate positions on an object is a **digitizer**. These devices can be used to input coordinate values in either a two-dimensional or a three-dimensional space. Typically, a digitizer is used to scan over a drawing or object and to input a set of discrete coordinate positions, which can be joined with straight-line segments to approximate the curve or surface shapes.

One type of digitizer is the **graphics tablet** (also referred to as a data tablet), which is used to input two-dimensional coordinates by activating a hand cursor or stylus at selected positions on a flat surface. A hand cursor contains cross hairs for sighting positions, while a stylus is a pencil-shaped device that is pointed at

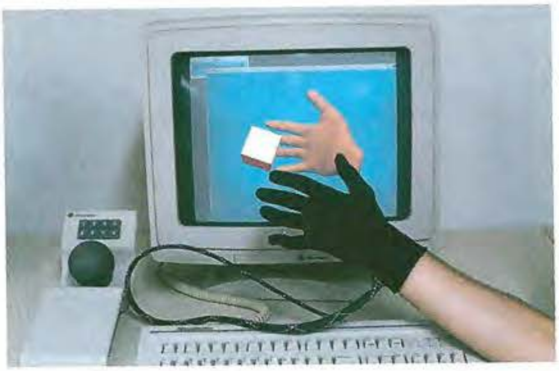


Figure 2-45
A virtual-reality scene, displayed on a two-dimensional video monitor, with input from a data glove and a spaceball. (*Courtesy of The Computer Graphics Center, Darmstadt, Germany.*)

positions on the tablet. Figures 2-46 and 2-47 show examples of desktop and floor-model tablets, using hand cursors that are available with 2, 4, or 16 buttons. Examples of stylus input with a tablet are shown in Figs. 2-48 and 2-49. The artist's digitizing system in Fig. 2-49 uses electromagnetic resonance to detect the three-dimensional position of the stylus. This allows an artist to produce different brush strokes with different pressures on the tablet surface. Tablet size varies from 12 by 12 inches for desktop models to 44 by 60 inches or larger for floor models. Graphics tablets provide a highly accurate method for selecting coordinate positions, with an accuracy that varies from about 0.2 mm on desktop models to about 0.05 mm or less on larger models.

Many graphics tablets are constructed with a rectangular grid of wires embedded in the tablet surface. Electromagnetic pulses are generated in sequence

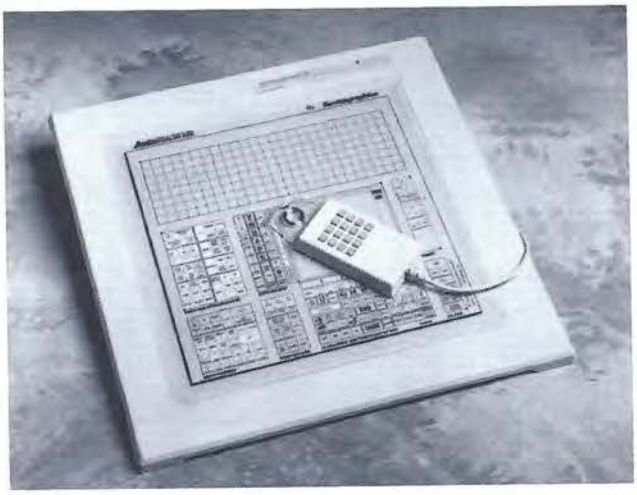


Figure 2-46
The SummaSketch III desktop tablet with a 16-button hand cursor. (*Courtesy of Summagraphics Corporation.*)

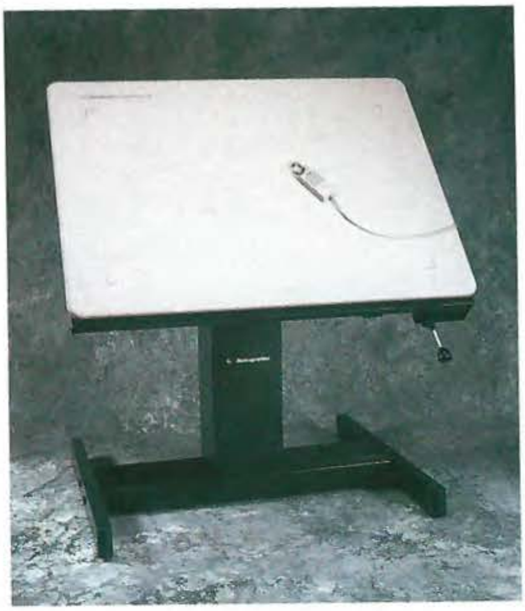


Figure 2-47
The Microgrid III tablet with a 16-button hand cursor, designed for digitizing larger drawings. *(Courtesy of Summagraphics Corporation.)*

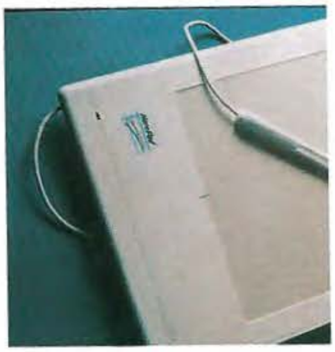


Figure 2-48
The NotePad desktop tablet with stylus. *(Courtesy of CalComp Digitizer Division, a part of CalComp, Inc.)*

along the wires, and an electric signal is induced in a wire coil in an activated stylus or hand cursor to record a tablet position. Depending on the technology, either signal strength, coded pulses, or phase shifts can be used to determine the position on the tablet.

Acoustic (or sonic) tablets use sound waves to detect a stylus position. Either strip microphones or point microphones can be used to detect the sound emitted by an electrical spark from a stylus tip. The position of the stylus is calcu-

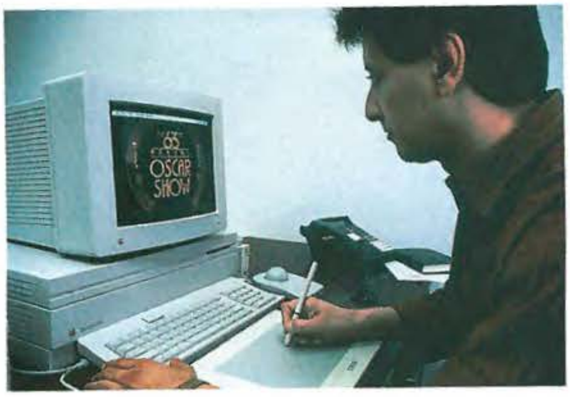


Figure 2-49
An artist's digitizer system, with a pressure-sensitive, cordless stylus. *(Courtesy of Wacom Technology Corporation.)*

lated by timing the arrival of the generated sound at the different microphone positions. An advantage of two-dimensional accoustic tablets is that the microphones can be placed on any surface to form the "tablet" work area. This can be convenient for various applications, such as digitizing drawings in a book.

Three-dimensional digitizers use sonic or electromagnetic transmissions to record positions. One electromagnetic transmission method is similar to that used in the data glove: A coupling between the transmitter and receiver is used to compute the location of a stylus as it moves over the surface of an object. Figure 2-50 shows a three-dimensional digitizer designed for Apple Macintosh computers. As the points are selected on a nonmetallic object, a wireframe outline of the surface is displayed on the computer screen. Once the surface outline is constructed, it can be shaded with lighting effects to produce a realistic display of the object. Resolution of this system is from 0.8 mm to 0.08 mm, depending on the model.

Image Scanners

Drawings, graphs, color and black-and-white photos, or text can be stored for computer processing with an **image scanner** by passing an optical scanning mechanism over the information to be stored. The gradations of gray scale or color are then recorded and stored in an array. Once we have the internal representation of a picture, we can apply transformations to rotate, scale, or crop the picture to a particular screen area. We can also apply various image-processing methods to modify the array representation of the picture. For scanned text input, various editing operations can be performed on the stored documents. Some scanners are able to scan either graphical representations or text, and they come in a variety of sizes and capabilities. A small hand-model scanner is shown in Fig. 2-51, while Figs 2-52 and 2-53 show larger models.

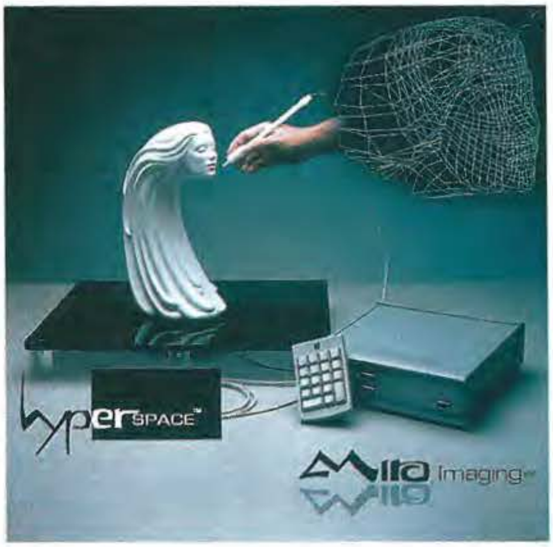


Figure 2-50
A three-dimensional digitizing system for use with Apple Macintosh computers. (*Courtesy of Mira Imaging.*)

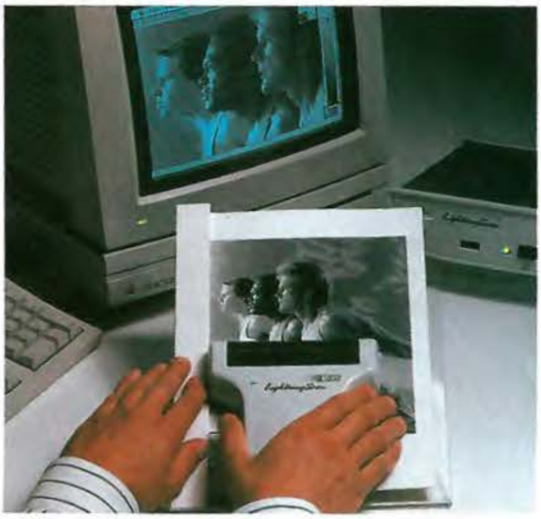


Figure 2-51
A hand-held scanner that can be used to input either text or graphics images. (*Courtesy of Thunderware, Inc.*)

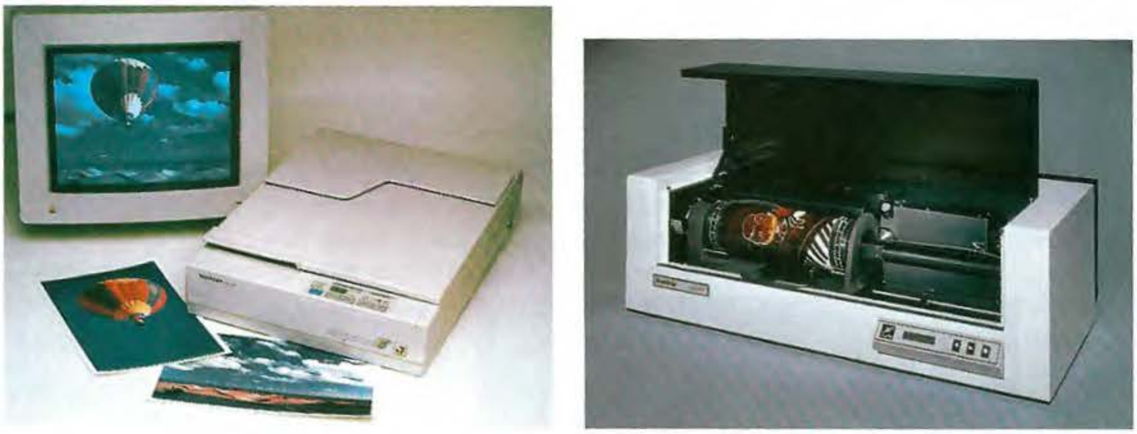


Figure 2-52
Desktop full-color scanners: (a) Flatbed scanner with a resolution of 600 dots per inch. (*Courtesy of Sharp Electronics Corporation.*) (b) Drum scanner with a selectable resolution from 50 to 4000 dots per inch. (*Courtesy of Howtek, Inc.*)

Touch Panels

As the name implies, **touch panels** allow displayed objects or screen positions to be selected with the touch of a finger. A typical application of touch panels is for the selection of processing options that are represented with graphical icons. Some systems, such as the plasma panels shown in Fig. 2-54, are designed with touch screens. Other systems can be adapted for touch input by fitting a transparent device with a touch-sensing mechanism over the video monitor screen. Touch input can be recorded using optical, electrical, or acoustical methods.

Optical touch panels employ a line of infrared light-emitting diodes (LEDs) along one vertical edge and along one horizontal edge of the frame. The opposite vertical and horizontal edges contain light detectors. These detectors are used to record which beams are interrupted when the panel is touched. The two crossing

Figure 2-53
A large floor-model scanner used to scan architectural and engineering drawings up to 40 inches wide and 100 feet long. (*Courtesy of Summagraphics Corporation.*)

beams that are interrupted identify the horizontal and vertical coordinates of the screen position selected. Positions can be selected with an accuracy of about 1/4 inch. With closely spaced LEDs, it is possible to break two horizontal or two vertical beams simultaneously. In this case, an average position between the two interrupted beams is recorded. The LEDs operate at infrared frequencies, so that the light is not visible to a user. Figure 2-55 illustrates the arrangement of LEDs in an optical touch panel that is designed to match the color and contours of the system to which it is to be fitted.

An electrical touch panel is constructed with two transparent plates separated by a small distance. One of the plates is coated with a conducting material, and the other plate is coated with a resistive material. When the outer plate is touched, it is forced into contact with the inner plate. This contact creates a voltage drop across the resistive plate that is converted to the coordinate values of the selected screen position.

In acoustical touch panels, high-frequency sound waves are generated in the horizontal and vertical directions across a glass plate. Touching the screen causes part of each wave to be reflected from the finger to the emitters. The screen position at the point of contact is calculated from a measurement of the time interval between the transmission of each wave and its reflection to the emitter.

(a) (b)

Figure 2-54
Plasma panels with touch screens. (*Courtesy of Photonics Systems.*)

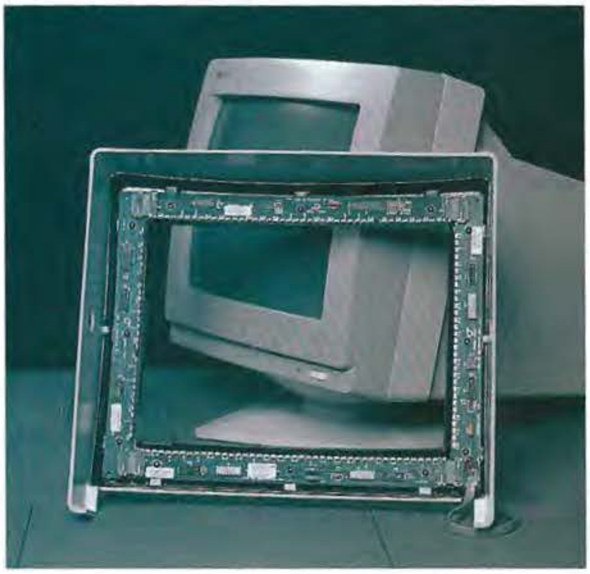


Figure 2-55
An optical touch panel, showing the arrangement of infrared LED units and detectors around the edges of the frame. (*Courtesy of Carroll Touch, Inc.*)

Light Pens

Figure 2-56 shows the design of one type of **light pen**. Such pencil-shaped devices are used to select screen positions by detecting the light coming from points on the CRT screen. They are sensitive to the short burst of light emitted from the phosphor coating at the instant the electron beam strikes a particular point. Other light sources, such as the background light in the room, are usually not detected by a light pen. An activated light pen, pointed at a spot on the screen as the electron beam lights up that spot, generates an electrical pulse that causes the coordinate position of the electron beam to be recorded. As with cursor-positioning devices, recorded light-pen coordinates can be used to position an object or to select a processing option.

Although light pens are still with us, they are not as popular as they once were since they have several disadvantages compared to other input devices that have been developed. For one, when a light pen is pointed at the screen, part of the screen image is obscured by the hand and pen. And prolonged use of the light pen can cause arm fatigue. Also, light pens require special implementations for some applications because they cannot detect positions within black areas. To be able to select positions in any screen area with a light pen, we must have some nonzero intensity assigned to each screen pixel. In addition, light pens sometimes give false readings due to background lighting in a room.

Voice Systems

Speech recognizers are used in some graphics workstations as input devices to accept voice commands. The **voice-system** input can be used to initiate graphics

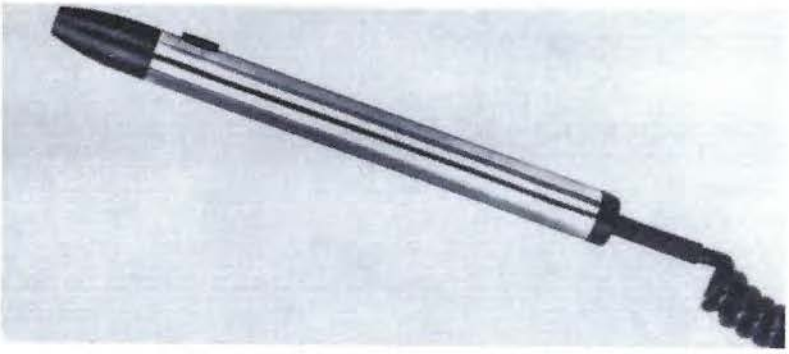


Figure 2-56
A light pen activated with a button switch. *(Courtesy of Interactive Computer Products.)*

operations or to enter data. These systems operate by matching an input against a predefined dictionary of words and phrases.

A dictionary is set up for a particular operator by having the operator speak the command words to be used into the system. Each word is spoken several times, and the system analyzes the word and establishes a frequency pattern for that word in the dictionary along with the corresponding function to be performed. Later, when a voice command is given, the system searches the dictionary for a frequency-pattern match. Voice input is typically spoken into a microphone mounted on a headset, as in Fig. 2-57. The microphone is designed to minimize input of other background sounds. If a different operator is to use the system, the dictionary must be reestablished with that operator's voice patterns. Voice systems have some advantage over other input devices, since the attention of the operator does not have to be switched from one device to another to enter a command.

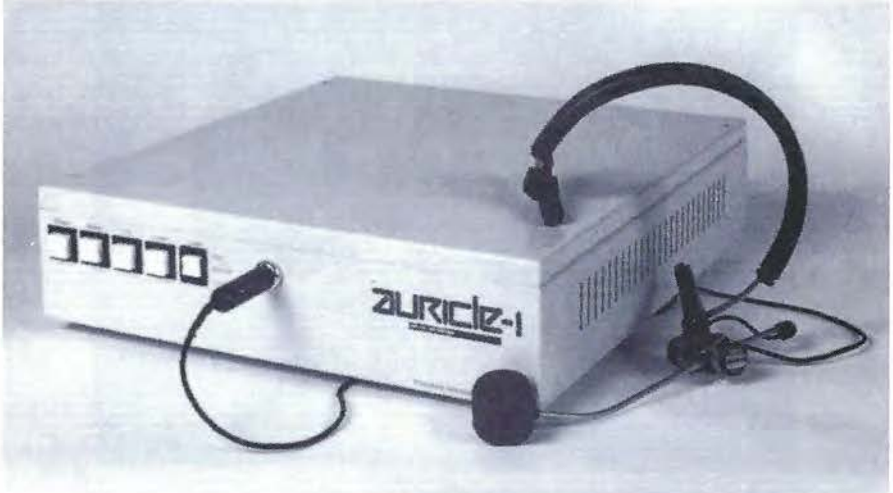
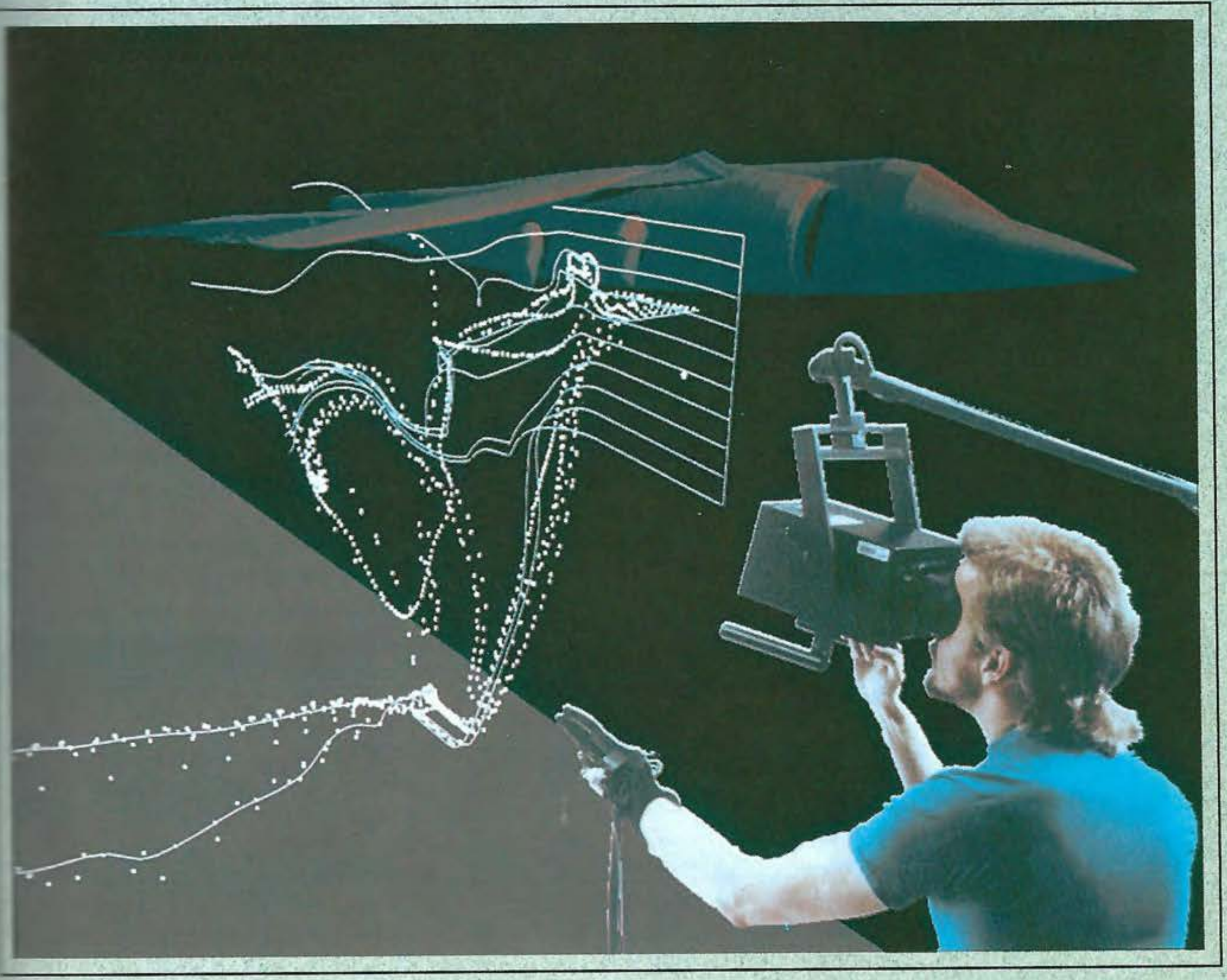


Figure 2-57
A speech-recognition system. *(Courtesy of Threshold Technology, Inc.)*

CHAPTER 8

Graphical User Interfaces and Interactive Input Methods



The human–computer interface for most systems involves extensive gr ics, regardless of the application. Typically, general systems now cons windows, pull-down and pop-up menus, icons, and pointing devices, such mouse or spaceball, for positioning the screen cursor. Popular graphical use terfaces include X Windows, Windows, Macintosh, OpenLook, and Motif. Th interfaces are used in a variety of applications, including word proces spreadsheets, databases and file-management systems, presentation systems, page-layout systems. In graphics packages, specialized interactive dialogues designed for individual applications, such as engineering design, architec design, data visualization, drafting, business graphs, and artist's paintbrush grams. For general graphics packages, interfaces are usually provided throu standard system. An example is the X Window System interface with PHIGS this chapter, we take a look at the basic elements of graphical user interfaces the techniques for interactive dialogues. We also consider how dialogues graphics packages, in particular, can allow us to construct and manipulate ture components, select menu options, assign parameter values, and select position text strings. A variety of input devices exists, and general grap packages can be designed to interface with various devices and to provide ex sive dialogue capabilities.

8-1 THE USER DIALOGUE

For a particular application, the *user's model* serves as the basis for the desig the dialogue. The user's model describes what the system is designed to acc plish and what graphics operations are available. It states the type of objects can be displayed and how the objects can be manipulated. For example, graphics system is to be used as a tool for architectural design, the model scribes how the package can be used to construct and display views of build by positioning walls, doors, windows, and other building components. Simil for a facility-layout system, objects could be defined as a set of furniture (tables, chairs, etc.), and the available operations would include those for p tioning and removing different pieces of furniture within the facility layout. a circuit-design program might use electrical or logic elements for objects, positioning operations available for adding or deleting elements within the all circuit design.

All information in the user dialogue is then presented in the language of the application. In an architectural design package, this means that all interactions are described only in architectural terms, without reference to particular data structures or other concepts that may be unfamiliar to an architect. In the following sections, we discuss some of the general considerations in structuring a user dialogue.

Windows and Icons

Figure 8-1 shows examples of common window and icon graphical interfaces. Visual representations are used both for objects to be manipulated in an application and for the actions to be performed on the application objects.

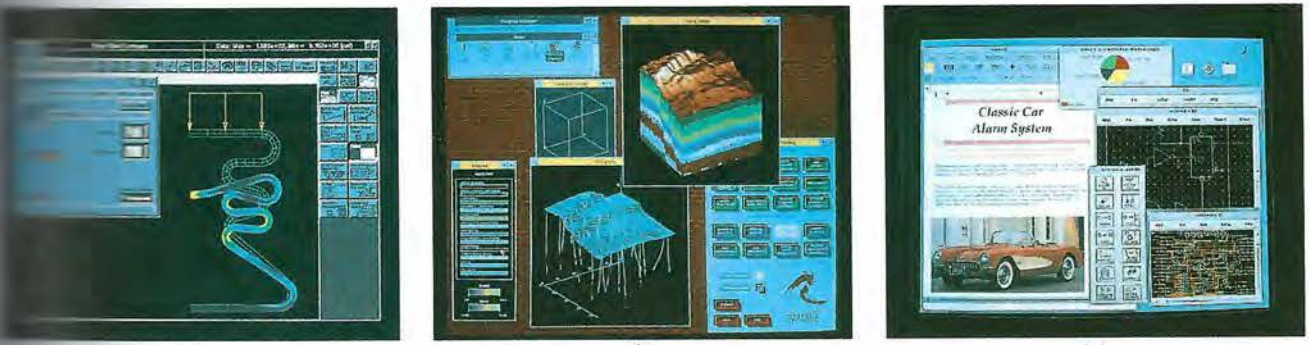
A window system provides a window-manager interface for the user and functions for handling the display and manipulation of the windows. Common functions for the window system are opening and closing windows, repositioning windows, resizing windows, and display routines that provide interior and exterior clipping and other graphics functions. Typically, windows are displayed with sliders, buttons, and menu icons for selecting various window options. Some general systems, such as X Windows and NeWS, are capable of supporting multiple window managers so that different window styles can be accommodated, each with its own window manager. The window managers can then be designed for particular applications. In other cases, a window system is designed for one specific application and window style.

Icons representing objects such as furniture items and circuit elements are often referred to as **application icons**. The icons representing actions, such as rotate, magnify, scale, clip, and paste, are called **control icons**, or **command icons**.

Accommodating Multiple Skill Levels

Usually, interactive graphical interfaces provide several methods for selecting actions. For example, options could be selected by pointing at an icon and clicking different mouse buttons, or by accessing pull-down or pop-up menus, or by typing keyboard commands. This allows a package to accommodate users that have different skill levels.

For a less experienced user, an interface with a few easily understood operations and detailed prompting is more effective than one with a large, compre-



(a)

(b)

(c)

Figure 8-1

Examples of screen layouts using window systems and icons. *(Courtesy of (a) Intergraph Corporation, (b) Visual Numerics, Inc., and (c) Sun Microsystems.)*

hensive operation set. A simplified set of menus and options is easy to lear remember, and the user can concentrate on the application instead of on th tails of the interface. Simple point-and-click operations are often easiest for a experienced user of an applications package. Therefore, interfaces typically vide a means for masking the complexity of a package, so that beginners ca the system without being overwhelmed with too much detail.

Experienced users, on the other hand, typically want speed. This m fewer prompts and more input from the keyboard or with multiple mouse ton clicks. Actions are selected with function keys or with simultaneous com tions of keyboard keys, since experienced users will remember these shortcu commonly used actions.

Similarly, help facilities can be designed on several levels so that begi can carry on a detailed dialogue, while more experienced users can redu eliminate prompts and messages. Help facilities can also include one or mo torial applications, which provide users with an introduction to the capab and use of the system.

Consistency

An important design consideration in an interface is consistency. For examp particular icon shape should always have a single meaning, rather than se to represent different actions or objects depending on the context. Some othe amples of consistency are always placing menus in the same relative positio that a user does not have to hunt for a particular option, always using a par lar combination of keyboard keys for the same action, and always color codi that the same color does not have different meanings in different situations.

Generally, a complicated, inconsistent model is difficult for a user to un stand and to work with in an effective way. The objects and operations prov should be designed to form a minimum and consistent set so that the syste easy to learn, but not oversimplified to the point where it is difficult to apply.

Minimizing Memorization

Operations in an interface should also be structured so that they are easy to derstand and to remember. Obscure, complicated, inconsistent, and abbrev command formats lead to confusion and reduction in the effectiveness of the of the package. One key or button used for all delete operations, for examp easier to remember than a number of different keys for different types of de operations.

Icons and window systems also aid in minimizing memorization. Diffe kinds of information can be separated into different windows, so that we do have to rely on memorization when different information displays overlap. can simply retain the multiple information on the screen in different windo and switch back and forth between window areas. Icons are used to reduce me orizing by displaying easily recognizable shapes for various objects and acti To select a particular action, we simply select the icon that resembles that actio

Backup and Error Handling

A mechanism for backing up, or aborting, during a sequence of operations is other common feature of an interface. Often an operation can be canceled be operation is completed, with the system restored to the state it was in before the operation was started. With the ability to back up at any point, we can confidently explore the capabilities of the system, knowing that the effects of a mistake can be erased.

Backup can be provided in many forms. An standard *undo* key or command is used to cancel a single operation. Sometimes a system can be backed up through several operations, allowing us to reset the system to some specified point. In a system with extensive backup capabilities, all inputs could be saved so that we can back up and "replay" any part of a session.

Sometimes operations cannot be undone. Once we have deleted the trash in the desktop wastebasket, for instance, we cannot recover the deleted files. In this case, the interface would ask us to verify the delete operation before proceeding.

Good diagnostics and error messages are designed to help determine the cause of an error. Additionally, interfaces attempt to minimize error possibilities by anticipating certain actions that could lead to an error. Examples of this are not allowing us to transform an object position or to delete an object when no object has been selected, not allowing us to select a line attribute if the selected object is not a line, and not allowing us to select the paste operation if nothing is in the clipboard.

Feedback

Interfaces are designed to carry on a continual interactive dialogue so that we are informed of actions in progress at each step. This is particularly important when the response time is high. Without feedback, we might begin to wonder what the system is doing and whether the input should be given again.

As each input is received, the system normally provides some type of response. An object is highlighted, an icon appears, or a message is displayed. This not only informs us that the input has been received, but it also tells us what the system is doing. If processing cannot be completed within a few seconds, several feedback messages might be displayed to keep us informed of the progress of the system. In some cases, this could be a flashing message indicating that the system is still working on the input request. It may also be possible for the system to display partial results as they are completed, so that the final display is built up a piece at a time. The system might also allow us to input other commands or data while one instruction is being processed.

Feedback messages are normally given clearly enough so that they have little chance of being overlooked, but not so overpowering that our concentration is interrupted. With function keys, feedback can be given as an audible click or by lighting up the key that has been pressed. Audio feedback has the advantage that it does not use up screen space, and we do not need to take attention from the work area to receive the message. When messages are displayed on the screen, a fixed message area can be used so that we always know where to look for messages. In some cases, it may be advantageous to place feedback messages in the work area near the cursor. Feedback can also be displayed in different colors to distinguish it from other displayed objects.

To speed system response, feedback techniques can be chosen to take advantage of the operating characteristics of the type of devices in use. A typical raster feedback technique is to invert pixel intensities, particularly when making menu selections. Other feedback methods include highlighting, blinking, and color changes.

Special symbols are designed for different types of feedback. For example, a cross, a frowning face, or a thumbs-down symbol is often used to indicate an error; and a blinking "at work" sign is used to indicate that processing is in progress. This type of feedback can be very effective with a more experienced user, but the beginner may need more detailed feedback that not only clearly indicates what the system is doing but also what the user should input next.

With some types of input, *echo* feedback is desirable. Typed characters can be displayed on the screen as they are input so that we can detect and correct errors immediately. Button and dial input can be echoed in the same way. Scalar values that are selected with dials or from displayed scales are usually echoed on the screen to let us check input values for accuracy. Selection of coordinate points can be echoed with a cursor or other symbol that appears at the selected position. For more precise echoing of selected positions, the coordinate values can be displayed on the screen.

8-2 INPUT OF GRAPHICAL DATA

Graphics programs use several kinds of input data. Picture specifications need values for coordinate positions, values for the character-string parameters, scalar values for the transformation parameters, values specifying menu options, and values for identification of picture parts. Any of the input devices discussed in Chapter 2 can be used to input the various graphical data types, but some devices are better suited for certain data types than others. To make graphics packages independent of the particular hardware devices used, input functions can be structured according to the data description to be handled by each function. This approach provides a **logical input-device classification** in terms of the kind of data to be input by the device.

Logical Classification of Input Devices

The various kinds of input data are summarized in the following six logical device classifications used by PHIGS and GKS:

LOCATOR—a device for specifying a coordinate position (x, y)

STROKE—a device for specifying a series of coordinate positions

STRING—a device for specifying text input

VALUATOR—a device for specifying scalar values

CHOICE—a device for selecting menu options

PICK—a device for selecting picture components

In some packages, a single logical device is used for both locator and stroke operations. Some other mechanism, such as a switch, can then be used to indicate whether one coordinate position or a "stream" of positions is to be input.

Each of the six logical input device classifications can be implemented with any of the hardware devices, but some hardware devices are more convenient for certain kinds of data than others. A device that can be pointed at a screen position is more convenient for entering coordinate data than a keyboard, for example. In the following sections, we discuss how the various physical devices are used to provide input within each of the logical classifications.

Locator Devices

A standard method for interactive selection of a coordinate point is by positioning the screen cursor. We can do this with a mouse, joystick, trackball, spaceball, thumbwheels, dials, a digitizer stylus or hand cursor, or some other cursor-positioning device. When the screen cursor is at the desired location, a button is activated to store the coordinates of that screen point.

Keyboards can be used for locator input in several ways. A general-purpose keyboard usually has four cursor-control keys that move the screen cursor up, down, left, and right. With an additional four keys, we can move the cursor diagonally as well. Rapid cursor movement is accomplished by holding down the selected cursor key. Alternatively, a joystick, joydisk, trackball, or thumbwheels can be mounted on the keyboard for relative cursor movement. As a last resort, we could actually type in coordinate values, but this is a slower process that also requires us to know exact coordinate values.

Light pens have also been used to input coordinate positions, but some special implementation considerations are necessary. Since light pens operate by detecting light emitted from the screen phosphors, some nonzero intensity level must be present at the coordinate position to be selected. With a raster system, we can paint a color background onto the screen. As long as no black areas are present, a light pen can be used to select any screen position. When it is not possible to eliminate all black areas in a display (such as on a vector system, for example), a light pen can be used as a locator by creating a small light pattern for the pen to detect. The pattern is moved around the screen until it finds the light pen.

Stroke Devices

This class of logical devices is used to input a sequence of coordinate positions. Stroke-device input is equivalent to multiple calls to a locator device. The set of input points is often used to display line sections.

Many of the physical devices used for generating locator input can be used as stroke devices. Continuous movement of a mouse, trackball, joystick, or tablet hand cursor is translated into a series of input coordinate values. The graphics tablet is one of the more common stroke devices. Button activation can be used to place the tablet into "continuous" mode. As the cursor is moved across the tablet surface, a stream of coordinate values is generated. This process is used in paintbrush systems that allow artists to draw scenes on the screen and in engineering systems where layouts can be traced and digitized for storage.

String Devices

The primary physical device used for string input is the keyboard. Input character strings are typically used for picture or graph labels.

Other physical devices can be used for generating character patterns in a "text-writing" mode. For this input, individual characters are drawn on the screen with a stroke or locator-type device. A pattern-recognition program then interprets the characters using a stored dictionary of predefined patterns.

Valuator Devices

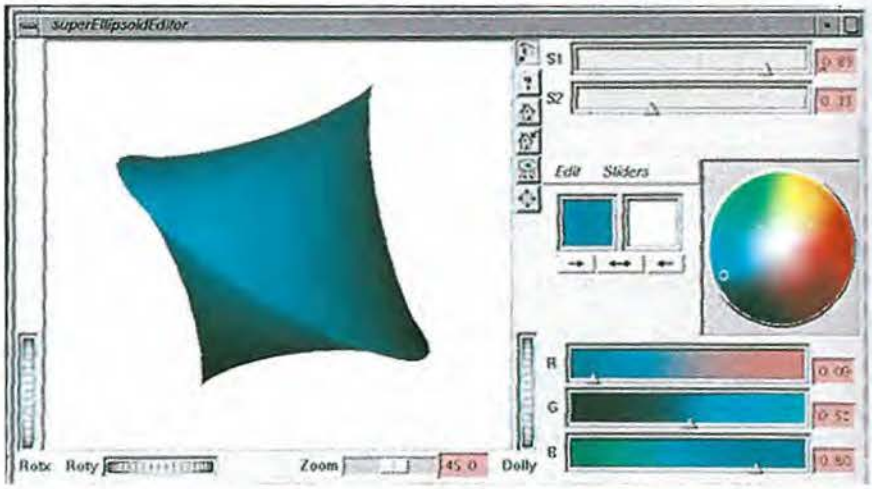
This logical class of devices is employed in graphics systems to input scalar values. Valuators are used for setting various graphics parameters, such as rotation angle and scale factors, and for setting physical parameters associated with a particular application (temperature settings, voltage levels, stress factors, etc.).

A typical physical device used to provide valuator input is a set of control dials. Floating-point numbers within any predefined range are input by rotating the dials. Dial rotations in one direction increase the numeric input value, and opposite rotations decrease the numeric value. Rotary potentiometers convert dial rotation into a corresponding voltage. This voltage is then translated into a real number within a defined scalar range, such as −10.5 to 25.5. Instead of dials, slide potentiometers are sometimes used to convert linear movements into scalar values.

Any keyboard with a set of numeric keys can be used as a valuator device. A user simply types the numbers directly in floating-point format, although this is a slower method than using dials or slide potentiometers.

Joysticks, trackballs, tablets, and other interactive devices can be adapted for valuator input by interpreting pressure or movement of the device relative to a scalar range. For one direction of movement, say, left to right, increasing scalar values can be input. Movement in the opposite direction decreases the scalar input value.

Another technique for providing valuator input is to display sliders, buttons, rotating scales, and menus on the video monitor. Figure 8-2 illustrates some possibilities for scale representations. Locator input from a mouse, joystick, spaceball, or other device is used to select a coordinate position on the display, and the screen coordinate position is then converted to a numeric input value. As a feedback mechanism for the user, the selected position on a scale can be marked with some symbol. Numeric values may also be echoed somewhere on the screen to confirm the selections.



S1 0.89
S2 0.33
R 0.09
G 0.53
B 0.80
Zoom 45.0

Figure 8-2
Scales displayed on a video monitor for interactive selection of parameter values. In this display, sliders are provided for selecting scalar values for superellipse parameters, s1 and s2, and for individual R, G, and B color values. In addition, a small circle can be positioned on the color wheel for selection of a combined RGB color, and buttons can be activated to make small changes in color values.

Choice Devices

Graphics packages use menus to select programming options, parameter values, and object shapes to be used in constructing a picture (Fig. 8-1). A choice device is defined as one that enters a selection from a list (menu) of alternatives. Commonly used choice devices are a set of buttons; a cursor positioning device, such as a mouse, trackball, or keyboard cursor keys; and a touch panel.

A function keyboard, or "button box", designed as a stand-alone unit, is often used to enter menu selections. Usually, each button is programmable, so that its function can be altered to suit different applications. Single-purpose buttons have fixed, predefined functions. Programmable function keys and fixed-function buttons are often included with other standard keys on a keyboard.

For screen selection of listed menu options, we can use cursor-control devices. When a coordinate position (x, y) is selected, it is compared to the coordinate extents of each listed menu item. A menu item with vertical and horizontal boundaries at the coordinate values $x_{\min}$, $x_{\max}$, $y_{\min}$, and $y_{\max}$ is selected if the input coordinates (x, y) satisfy the inequalities

$$x_{\min} \leq x \leq x_{\max}, \qquad y_{\min} \leq y \leq y_{\max} \tag{8-1}$$

For larger menus with a few options displayed at a time, a touch panel is commonly used. As with a cursor-control device, such as a mouse, a selected screen position is compared to the area occupied by each menu choice.

Alternate methods for choice input include keyboard and voice entry. A standard keyboard can be used to type in commands or menu options. For this method of choice input, some abbreviated format is useful. Menu listings can be numbered or given short identifying names. Similar codings can be used with voice-input systems. Voice input is particularly useful when the number of options is small (20 or less).

Pick Devices

Graphical object selection is the function of this logical class of devices. Pick devices are used to select parts of a scene that are to be transformed or edited in some way.

Typical devices used for object selection are the same as those for menu selection: the cursor-positioning devices. With a mouse or joystick, we can position the cursor over the primitives in a displayed structure and press the selection button. The position of the cursor is then recorded, and several levels of search may be necessary to locate the particular object (if any) that is to be selected. First, the cursor position is compared to the coordinate extents of the various structures in the scene. If the bounding rectangle of a structure contains the cursor coordinates, the picked structure has been identified. But if two or more structure areas contain the cursor coordinates, further checks are necessary. The coordinate extents of individual lines in each structure can be checked next. If the cursor coordinates are determined to be inside the coordinate extents of only one line, for example, we have identified the picked object. Otherwise, we need additional checks to determine the closest line to the cursor position.

One way to find the closest line to the cursor position is to calculate the distance squared from the cursor coordinates (x, y) to each line segment whose bounding rectangle contains the cursor position (Fig. 8-3). For a line with endpoints (x_1, y_1) and (x_2, y_2), distance squared from (x, y) to the line is calculated as

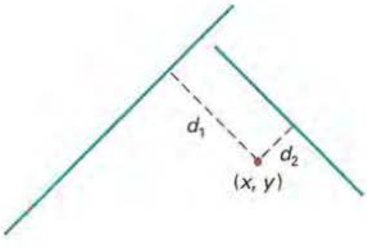


Figure 8-3
Distances to line segments from the pick position.

$$d^2 = \frac{[\Delta x(y-y_1)-\Delta y(x-x_1)]^2}{\Delta x^2+\Delta y^2}$$

where $\Delta x = x_2 - x_1$, and $\Delta y = y_2 - y_1$. Various approximations can be used to speed up this distance calculation, or other identification schemes can be used.

Another method for finding the closest line to the cursor position is to specify the size of a **pick window**. The cursor coordinates are centered on this window and the candidate lines are clipped to the window, as shown in Fig. 8-4. By making the pick window small enough, we can ensure that a single line will cross the window. The method for selecting the size of a pick window is described in Section 8-4, where we consider the parameters associated with various input functions.

A method for avoiding the calculation of pick distances or window clipping intersections is to highlight the candidate structures and allow the user to resolve the pick ambiguity. One way to do this is to highlight the structures that overlap the cursor position one by one. The user then signals when the desired structure is highlighted.

An alternative to cursor positioning is to use button input to highlight successive structures. A second button is used to stop the process when the desired structure is highlighted. If very many structures are to be searched in this way, the process can be speeded up and an additional button is used to help identify the structure. The first button can initiate a rapid successive highlighting of structures. A second button can again be used to stop the process, and a third button can be used to back up more slowly if the desired structure passed before the operator pressed the stop button.

Finally, we could use a keyboard to type in structure names. This is a straightforward, but less interactive, pick-selection method. Descriptive names can be used to help the user in the pick process, but the method has several drawbacks. It is generally slower than interactive picking on the screen, and a user will probably need prompts to remember the various structure names. In addition, picking structure subparts from the keyboard can be more difficult than picking the subparts on the screen.

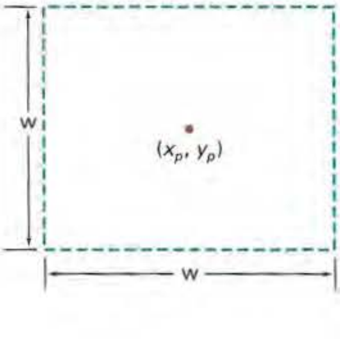


Figure 8-4
A pick window, centered on pick coordinates (x_p, y_p), used to resolve pick object overlaps.

8-3 INPUT FUNCTIONS

Graphical input functions can be set up to allow users to specify the following options:

- Which physical devices are to provide input within a particular logical classification (for example, a tablet used as a stroke device).
- How the graphics program and devices are to interact (input mode). Either the program or the devices can initiate data entry, or both can operate simultaneously.
- When the data are to be input and which device is to be used at that time to deliver a particular input type to the specified data variables.

Input Modes

Functions to provide input can be structured to operate in various **input modes**, which specify how the program and input devices interact. Input could be initiated by the program, or the program and input devices both could be operating simultaneously, or data input could be initiated by the devices. These three input modes are referred to as request mode, sample mode, and event mode.

In **request mode**, the application program initiates data entry. Input values are requested and processing is suspended until the required values are received. This input mode corresponds to typical input operation in a general programming language. The program and the input devices operate alternately. Devices are put into a wait state until an input request is made; then the program waits until the data are delivered.

In **sample mode**, the application program and input devices operate independently. Input devices may be operating at the same time that the program is processing other data. New input values from the input devices are stored, replacing previously input data values. When the program requires new data, it samples the *current* values from the input devices.

In **event mode**, the input devices initiate data input to the application program. The program and the input devices again operate concurrently, but now the input devices deliver data to an input queue. All input data are saved. When the program requires new data, it goes to the data queue.

Any number of devices can be operating at the same time in sample and event modes. Some can be operating in sample mode, while others are operating in event mode. But only one device at a time can be providing input in request mode.

An input mode within a logical class for a particular physical device operating on a specified workstation is declared with one of six input-class functions of the form

```
set ... Mode (ws, deviceCode, inputMode, echoFlag)
```

where `deviceCode` is a positive integer; `inputMode` is assigned one of the values *request*, *sample*, or *event*; and parameter `echoFlag` is assigned either the value *echo* or the value *noecho*. How input data will be echoed on the display device is determined by parameters set in other input functions to be described later in this section.

TABLE 8-1
ASSIGNMENT OF INPUT-DEVICE CODES

Device Code	*Physical Device Type*
1	Keyboard
2	Graphics Tablet
3	Mouse
4	Joystick
5	Trackball
6	Button

Device code assignment is installation-dependent. One possible assignment of device codes is shown in Table 8-1. Using the assignments in this table, we could make the following declarations:

```
setLocatorMode (1, 2, sample, noecho)
setTextMode (2, 1, request, echo)
setPickMode (4, 3, event, echo)
```

Thus, the graphics tablet is declared to be a locator device in sample mode on workstation 1 with no input data feedback echo; the keyboard is a text device in request mode on workstation 2 with input echo; and the mouse is declared to be a pick device in event mode on workstation 4 with input echo.

Request Mode

Input commands used in this mode correspond to standard input functions in a high-level programming language. When we ask for an input in request mode, other processing is suspended until the input is received. After a device has been assigned to request mode, as discussed in the preceding section, input requests can be made to that device using one of the six logical-class functions represented by the following:

```
request ... (ws, deviceCode, status, ... )
```

Values input with this function are the workstation code and the device code. Returned values are assigned to parameter `status` and to the data parameters corresponding to the requested logical class.

A value of *ok* or *none* is returned in parameter `status`, according to the validity of the input data. A value of *none* indicates that the input device was activated so as to produce invalid data. For locator input, this could mean that the coordinates were out of range. For pick input, the device could have been activated while not pointing at a structure. Or a "break" button on the input device could have been pressed. A returned value of *none* can be used as an end-of-data signal to terminate a programming sequence.

Locator and Stroke Input in Request Mode

The request functions for these two logical input classes are

```
requestLocator (ws, devCode, status, viewIndex, pt)
requestStroke (ws, devCode, nMax, status, viewIndex, n, pts)
```

For locator input, pt is the world-coordinate position selected. For stroke input, pt is a list of n coordinate positions, where parameter nMax gives the maximum number of points that can go in the input list. Parameter viewIndex is assigned the two-dimensional view index number.

Determination of a world-coordinate position is a two-step process: (1) The physical device selects a point in device coordinates (usually from the video-display screen) and the inverse of the workstation transformation is performed to obtain the corresponding point in normalized device coordinates. (2) Then, the inverse of the window-to-viewport mapping is carried out to get to viewing coordinates, then to world coordinates.

Since two or more views may overlap on a device, the correct viewing transformation is identified according to the **view-transformation input priority number**. By default, this is the same as the view index number, and the lower the number, the higher the priority. View index 0 has the highest priority. We can change the view priority relative to another (reference) viewing transformation with

```
setViewTransformationInputPriority (ws, viewIndex,
                        refViewIndex, priority)
```

where viewIndex identifies the viewing transformation whose priority is to be changed, refViewIndex identifies the reference viewing transformation, and parameter priority is assigned either the value *lower* or the value *higher*. For example, we can alter the priority of the first four viewing transformations on workstation 1, as shown in Fig. 8-5, with the sequence of functions:

```
setViewTransformationInputPriority (1, 3, 1, higher)
setViewTransformationInputPriority (1, 0, 2, lower)
```

String Input in Request Mode

Here, the request input function is

```
requestString (ws, devCode, status, nChars, str)
```

Parameter str in this function is assigned an input string. The number of characters in the string is given in parameter nChars.

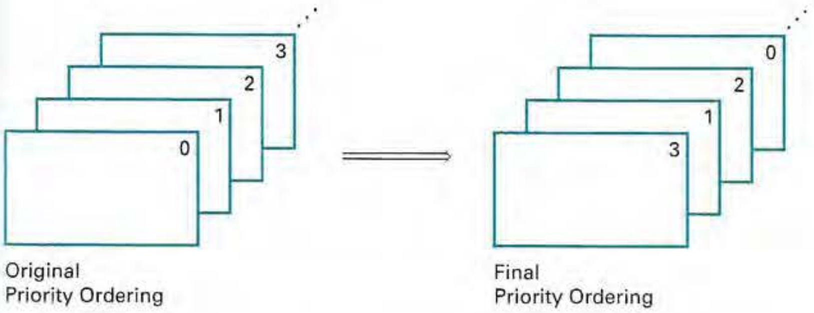


Figure 8-5
Rearranging viewing priorities.

Valuator Input in Request Mode

A numerical value is input in request mode with

```
requestValuator (ws, devCode, status, value)
```

Parameter `value` can be assigned any real-number value.

Choice Input in Request Mode

We make a menu selection with the following request function:

```
requestChoice (ws, devCode, status, itemNum)
```

Parameter `itemNum` is assigned a positive integer value corresponding to the menu item selected.

Pick Input in Request Mode

For this mode, we obtain a structure identifier number with the function

```
requestPick (ws, devCode, maxPathDepth, status, pathDepth,
                        pickPath)
```

Parameter `pickPath` is a list of information identifying the primitive selected. This list contains the structure name, pick identifier for the primitive, and the element sequence number. Parameter `pickDepth` is the number of levels returned in `pickPath`, and `maxPathDepth` is the specified maximum path depth that can be included in `pickPath`.

Subparts of a structure can be labeled for pick input with the following function:

```
setPickIdentifier (pickID)
```

An example of sublabeling during structure creation is given in the following programming sequence:

```
openStructure (id);
     for k := 1 to n do begin
                setPickIdentifier (k);
                          .
                          .
                          .
     end;
closeStructure;
```

Picking of structures and subparts of structures is also controlled by some workstation filters (Section 7-1). Objects cannot be picked if they are invisible. Also, we can set the ability to pick objects independently of their visibility. This is accomplished with the pick filter:

```
setPickFilter (ws, devCode, pickables, nonpickables)
```

the set pickables contains the names of objects (structures and primi- that we may want to select with the specified pick device. Similarly, the set ckables contains the names of objects that we do not want to be avail- picking with this input device.

e Mode

mple mode has been set for one or more physical devices, data input be- thout waiting for program direction. If a joystick has been designated as a device in sample mode, coordinate values for the current position of the d joystick are immediately stored. As the activated stick position changes, red values are continually replaced with the coordinates of the current sition.

Sampling of the current values from a physical device in this mode begins a sample command is encountered in the application program. A locator is sampled with one of the six logical-class functions represented by the ing:

```
ample ... (ws, deviceCode, ... )
```

device classes have a status parameter in sample mode, and some do not. input parameters are the same as in request mode.

As an example of sample input, suppose we want to translate and rotate a d object. A final translation position for the object can be obtained with a and the rotation angle can be supplied by a valuator device, as demon- d in the following statements.

```
ampleLocator (ws1, dev1, viewIndex, pt)
ampleValuator (ws2, dev2, angle)
```

Mode

an input device is placed in event mode, the program and device operate aneously. Data input from the device is accumulated in an event queue, or queue. All input devices active in event mode can enter data (referred to as ts") into this single-event queue, with each device entering data values as are generated. At any one time, the event queue can contain a mixture of types, in the order they were input. Data entered into the queue are identi- according to logical class, workstation number, and physical-device code.

An application program can be directed to check the event queue for any with the function

```
awaitEvent (time, ws, deviceClass, deviceCode)
```

meter time is used to set a maximum waiting time for the application pro- . If the queue happens to be empty, processing is suspended until either the ber of seconds specified in time has elapsed or an input arrives. Should the ng time run out before data values are input, the parameter deviceClass ssigned the value *none*. When time is given the value 0, the program checks queue and immediately returns to other processing if the queue is empty.

If processing is directed to the event queue with the `awaitEvent` function and the queue is not empty, the first event in the queue is transferred to a *current event record*. The particular logical device class, such as locator or stroke, that made this input is stored in parameter `deviceClass`. Codes, identifying the particular workstation and physical device that made the input, are stored in parameters `ws` and `deviceCode`, respectively.

To retrieve a data input from the current event record, an event-mode input function is used. The functions in event mode are similar to those in request and sample modes. However, no workstation and device-code parameters are necessary in the commands, since the values for these parameters are stored in the data record. A user retrieves data with

```
get ... ( ... )
```

For example, to ask for locator input, we invoke the function

```
getLocator (viewIndex, pt)
```

In the following program section, we give an example of the use of the `awaitEvent` and `get` functions. A set of points from a tablet (device code 2) on workstation 1 is input to plot a series of straight-line segments connecting the input coordinates:

```
setStrokeMode (1, 2, event, noecho);

repeat
    awaitEvent (0, ws, deviceClass, deviceCode)
until deviceClass = stroke;
getStroke (nMax, viewIndex, n, pts);
polyline (n, pts);
```

The `repeat-until` loop bypasses any data from other devices that might be in the queue. If the tablet is the only active input device in event mode, this loop is not necessary.

A number of devices can be used at the same time in event mode for rapid interactive processing of displays. The following statements plot input lines from a tablet with attributes specified by a button box:

```
setPolylineIndex (1);
{ set tablet to stroke device, event mode }
setStrokeMode (1, 2, event, noecho);
{ set buttons to choice device, event mode }
setChoiceMode (1, 6, event, noecho);
repeat
  awaitEvent (60, ws, deviceClass, deviceCode);
  if deviceClass = choice then
    begin
      getChoice (status, option);
      setPolylineIndex (option);
    end;
  else
    if deviceClass = stroke then
      begin
        getStroke (nMax, viewIndex, n, pts);
        polyline (n, pts);
      end;
until deviceClass = none;
```

Some additional housekeeping functions can be used in event mode. Functions for clearing the event queue are useful when a process is terminated and a new application is to begin. These functions can be set to clear the entire queue or to clear only data associated with specified input devices and workstations.

Concurrent Use of Input Modes

An example of the simultaneous use of input devices in different modes is given in the following procedure. An object is dragged around the screen with a mouse. When a final position has been selected, a button is pressed to terminate any further movement of the object. The mouse positions are obtained in sample mode, and the button input is sent to the event queue.

```
{ drags object in response to mouse input }
{ terminate processing by button press        }
begin
  setLocatorMode (1, 3, sample, echo);
  setChoiceMode (1, 6, event, noecho);
  repeat
    sampleLocator (1, 3, viewIndex, pt);
      { translate object centroid to position pt and draw }
    awaitEvent (0, ws, class, code)
  until class = choice
end;
```

8-4 INITIAL VALUES FOR INPUT-DEVICE PARAMETERS

Quite a number of parameters can be set for input devices using the initialize function for each logical class:

```
initialize ... (ws, deviceCode, ... , pe, coordExt, dataRec)
```

Parameter pe is the prompt and echo type, parameter coordExt is assigned a set of four coordinate values, and parameter dataRec is a record of various control parameters.

For locator input, some values that can be assigned to the prompt and echo parameter are

pe = **1:** installation defined
pe = **2:** crosshair cursor centered at current position
pe = **3:** line from initial position to current position
pe = **4:** rectangle defined by current and initial points

Several other options are also available.

For structure picking, we have the following options:

pe = **1:** highlight picked primitives
pe = **2:** highlight all primitives with value of pick id
pe = **3:** highlight entire structure

as well as several others.

When an echo of the input data is requested, it is displayed within bounding rectangle specified by the four coordinates in parameter coor Additional options can also be set in parameter dataRec. For example, we set any of the following:

- size of the pick window
- minimum pick distance
- type and size of cursor display
- type of structure highlighting during pick operations
- range (min and max) for valuator input
- resolution (scale) for valuator input

plus a number of other options.

8-5 INTERACTIVE PICTURE-CONSTRUCTION TECHNIQUES

There are several techniques that are incorporated into graphics packages t the interactive construction of pictures. Various input options can be provide that coordinate information entered with locator and stroke devices can be justed or interpreted according to a selected option. For example, we can res all lines to be either horizontal or vertical. Input coordinates can establish the sition or boundaries for objects to be drawn, or they can be used to rearrange p viously displayed objects.

Basic Positioning Methods

Coordinate values supplied by locator input are often used with positi methods to specify a location for displaying an object or a character string. W teractively select coordinate positions with a pointing device, usually by p tioning the screen cursor. Just how the object or text-string positioning is formed depends on the selected options. With a text string, for example screen point could be taken as the center string position, or the start or end p tion of the string, or any of the other string-positioning options discusse Chapter 4. For lines, straight line segments can be displayed between tw lected screen positions.

As an aid in positioning objects, numeric values for selected positions be echoed on the screen. Using the echoed coordinate values as a guide, we make adjustments in the selected location to obtain accurate positioning.

Constraints

With some applications, certain types of prescribed orientations or object a ments are useful. A constraint is a rule for altering input-coordinate values produce a specified orientation or alignment of the displayed coordinates. T are many kinds of constraint functions that can be specified, but the most c mon constraint is a horizontal or vertical alignment of straight lines. This typ constraint, shown in Figs. 8-6 and 8-7, is useful in forming network layouts. W this constraint, we can create horizontal and vertical lines without worry about precise specification of endpoint coordinates.

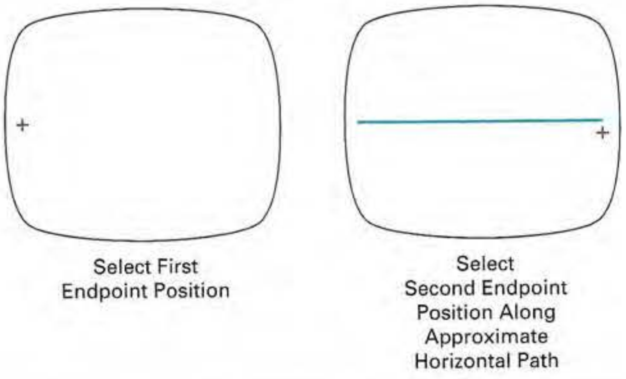

Select First Endpoint Position
Select Second Endpoint Position Along Approximate Horizontal Path

Figure 8-6
Horizontal line constraint.

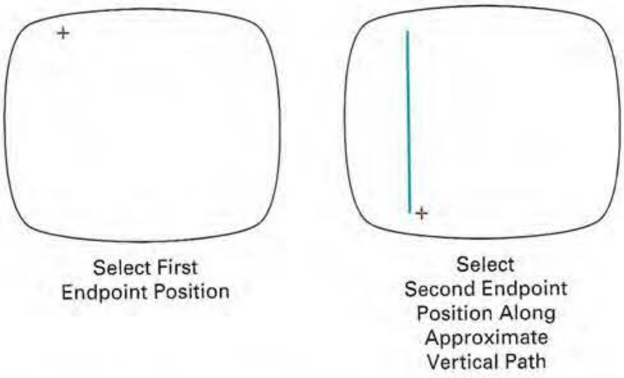

Select First Endpoint Position
Select Second Endpoint Position Along Approximate Vertical Path

Figure 8-7
Vertical line constraint.

A horizontal or vertical constraint is implemented by determining whether any two input coordinate endpoints are more nearly horizontal or more nearly vertical. If the difference in the y values of the two endpoints is smaller than the difference in x values, a horizontal line is displayed. Otherwise, a vertical line is drawn. Other kinds of constraints can be applied to input coordinates to produce a variety of alignments. Lines could be constrained to have a particular slant, such as 45°, and input coordinates could be constrained to lie along predefined paths, such as circular arcs.

Grids

Another kind of constraint is a grid of rectangular lines displayed in some part of the screen area. When a grid is used, any input coordinate position is rounded to the nearest intersecton of two grid lines. Figure 8-8 illustrates line drawing with a grid. Each of the two cursor positions is shifted to the nearest grid intersection point, and the line is drawn between these grid points. Grids facilitate object constructions, because a new line can be joined easily to a previously drawn line by selecting any position near the endpoint grid intersection of one end of the displayed line.

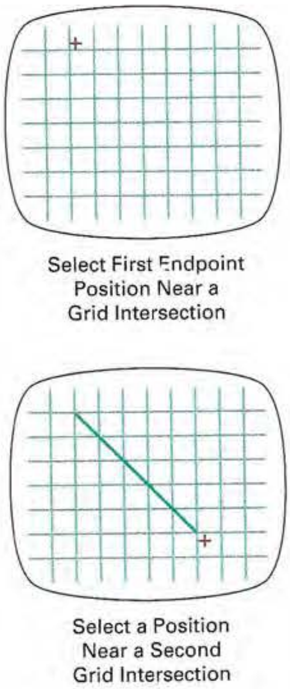

Select First Endpoint Position Near a Grid Intersection
Select a Position Near a Second Grid Intersection

Figure 8-8
Line drawing using a grid.

Spacing between grid lines is often an option that can be set by the user. Similarly, grids can be turned on and off, and it is sometimes possible to use partial grids and grids of different sizes in different screen areas.

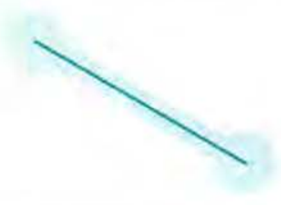


Figure 8-9
Gravity field around a line. Any selected point in the shaded area is shifted to a position on the line.

Gravity Field

In the construction of figures, we sometimes need to connect lines at positions between endpoints. Since exact positioning of the screen cursor at the connecting point can be difficult, graphics packages can be designed to convert any input position near a line to a position on the line.

This conversion of input position is accomplished by creating a *gravity field* area around the line. Any selected position within the gravity field of a line is moved ("gravitated") to the nearest position on the line. A gravity field area around a line is illustrated with the shaded boundary shown in Fig. 8-9. Areas around the endpoints are enlarged to make it easier for us to connect lines at their endpoints. Selected positions in one of the circular areas of the gravity field are attracted to the endpoint in that area. The size of gravity fields is chosen large enough to aid positioning, but small enough to reduce chances of overlap with other lines. If many lines are displayed, gravity areas can overlap, and it may be difficult to specify points correctly. Normally, the boundary for the gravity field is not displayed.

Rubber-Band Methods

Straight lines can be constructed and positioned using *rubber-band* methods, which stretch out a line from a starting position as the screen cursor is moved. Figure 8-10 demonstrates the rubber-band method. We first select a screen position for one endpoint of the line. Then, as the cursor moves around, the line is displayed from the start position to the current position of the cursor. When we finally select a second screen position, the other line endpoint is set.

Rubber-band methods are used to construct and position other objects besides straight lines. Figure 8-11 demonstrates rubber-band construction of a rectangle, and Fig. 8-12 shows a rubber-band circle construction.

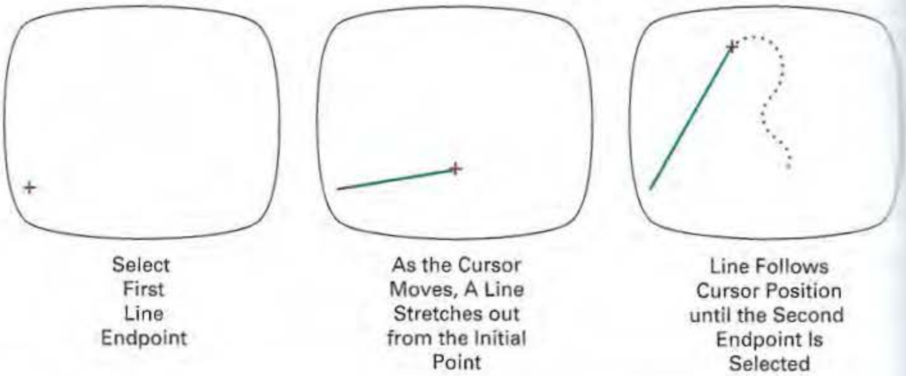

Select First Line Endpoint
As the Cursor Moves, A Line Stretches out from the Initial Point
Line Follows Cursor Position until the Second Endpoint Is Selected

Figure 8-10
Rubber-band method for drawing and positioning a straight line segment.

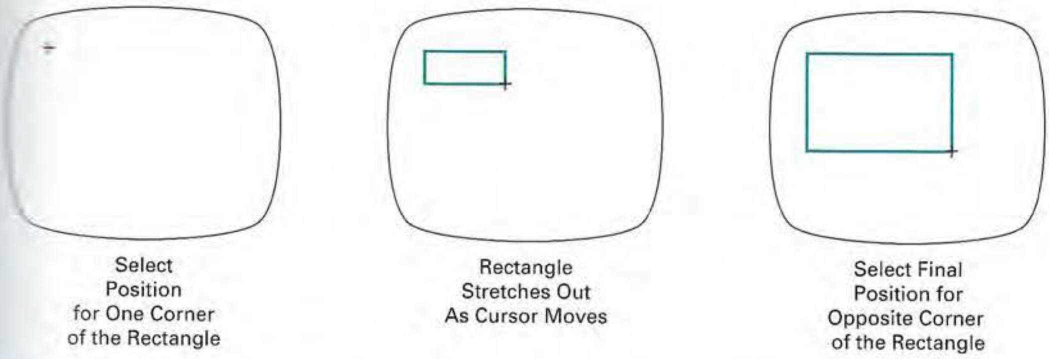


Figure 8-11
Rubber-band method for constructing a rectangle.

Dragging

A technique that is often used in interactive picture construction is to move objects into position by dragging them with the screen cursor. We first select an object, then move the cursor in the direction we want the object to move, and the selected object follows the cursor path. Dragging objects to various positions in a scene is useful in applications where we might want to explore different possibilities before selecting a final location.

Painting and Drawing

Options for sketching, drawing, and painting come in a variety of forms. Straight lines, polygons, and circles can be generated with methods discussed in the previous sections. Curve-drawing options can be provided using standard curve shapes, such as circular arcs and splines, or with freehand sketching procedures. Splines are interactively constructed by specifying a set of discrete screen points that give the general shape of the curve. Then the system fits the set of points with a polynomial curve. In freehand drawing, curves are generated by following the path of a stylus on a graphics tablet or the path of the screen cursor on a video monitor. Once a curve is displayed, the designer can alter the curve shape by adjusting the positions of selected points along the curve path.

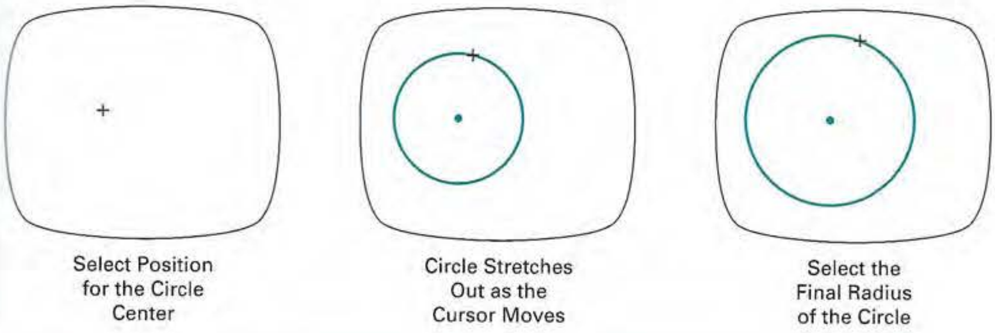


Figure 8-12
Constructing a circle using a rubber-band method.

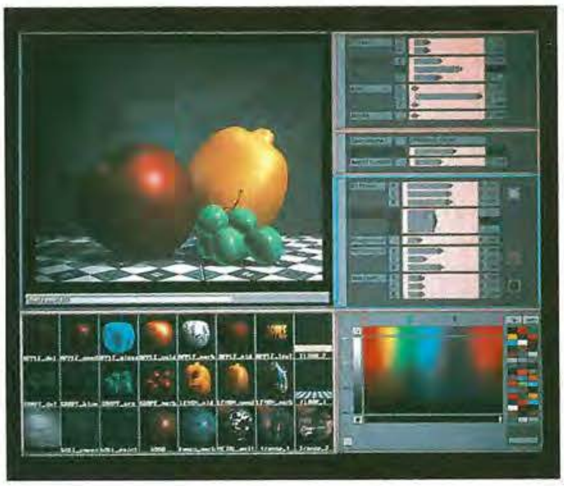


Figure 8-13
A screen layout showing one type of interface to an artist's painting package. (*Courtesy of Thomson Digital Image.*)

Line widths, line styles, and other attribute options are also commonly found in painting and drawing packages. These options are implemented with the methods discussed in Chapter 4. Various brush styles, brush patterns, color combinations, object shapes, and surface-texture patterns are also available on many systems, particularly those designed as artist's workstations. Some paint systems vary the line width and brush strokes according to the pressure of the artist's hand on the stylus. Figure 8-13 shows a window and menu system used with a painting package that allows an artist to select variations of a specified object shape, different surface textures, and a variety of lighting conditions for a scene.

8-6 VIRTUAL-REALITY ENVIRONMENTS

A typical virtual-reality environment is illustrated in Fig. 8-14. Interactive input is accomplished in this environment with a data glove (Section 2-5), which is capable of grasping and moving objects displayed in a virtual scene. The computer-generated scene is displayed through a head-mounted viewing system (Section 2-1) as a stereoscopic projection. Tracking devices compute the position and orientation of the headset and data glove relative to the object positions in the scene. With this system, a user can move through the scene and rearrange object positions with the data glove.

Another method for generating virtual scenes is to display stereoscopic projections on a raster monitor, with the two stereoscopic views displayed on alternate refresh cycles. The scene is then viewed through stereoscopic glasses. Interactive object manipulations can again be accomplished with a data glove and a tracking device to monitor the glove position and orientation relative to the position of objects in the scene.

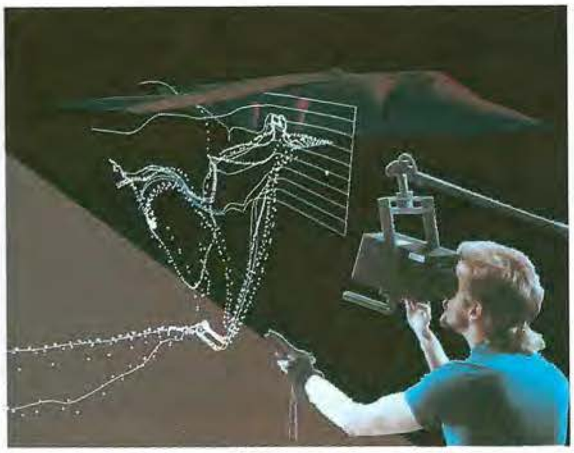


Figure 8-14
Using a head-tracking stereo display, called the BOOM (Fake Space Labs, Inc.), and a Dataglove (VPL, Inc.), a researcher interactively manipulates exploratory probes in the unsteady flow around a Harrier jet airplane. Software developed by Steve Bryson; data from Harrier. *(Courtesy of Sam Uselton, NASA Ames Research Center.)*

SUMMARY

A dialogue for an applications package can be designed from the user's model, which describes the functions of the applications package. All elements of the dialogue are presented in the language of the applications. Examples are electrical and architectural design packages.

Graphical interfaces are typically designed using windows and icons. A window system provides a window-manager interface with menus and icons that allows users to open, close, reposition, and resize windows. The window system then contains routines to carry out these operations, as well as the various graphics operations. General window systems are designed to support multiple window managers. Icons are graphical symbols that are designed for quick identification of application processes or control processes.

Considerations in user-dialogue design are ease of use, clarity, and flexibility. Specifically, graphical interfaces are designed to maintain consistency in user interaction and to provide for different user skill levels. In addition, interfaces are designed to minimize user memorization, to provide sufficient feedback, and to provide adequate backup and error-handling capabilities.

Input to graphics programs can come from many different hardware devices, with more than one device providing the same general class of input data. Graphics input functions can be designed to be independent of the particular input hardware in use, by adopting a logical classification for input devices. That is, devices are classified according to the type of graphics input, rather than a hardware designation, such as mouse or tablet. The six logical devices in mon use are locator, stroke, string, valuator, choice, and pick. Locator devic any devices used by a program to input a single coordinate position. Strok vices input a stream of coordinates. String devices are used to input text. Val devices are any input devices used to enter a scalar value. Choice devices menu selections. And pick devices input a structure name.

Input functions available in a graphics package can be defined in input modes. Request mode places input under the control of the applic program. Sample mode allows the input devices and program to operate co rently. Event mode allows input devices to initiate data entry and control cessing of data. Once a mode has been chosen for a logical device class an particular physical device to be used to enter this class of data, input functio the program are used to enter data values into the program. An application gram can make simultaneous use of several physical input devices operati different modes.

Interactive picture-construction methods are commonly used in a varie applications, including design and painting packages. These methods pro users with the capability to position objects, to constrain figures to prede orientations or alignments, to sketch figures, and to drag objects around screen. Grids, gravity fields, and rubber-band methods are used to aid in tioning and other picture-construction operations.

REFERENCES

Guidelines for user-interface design are presented in Apple (1987), Bleser (1988), D (1989), and OSF/MOTIF (1989). For information on the X Window System, see (1990) and Cutler, Gilly, and Reilly (1992). Additional discussions of interface desig be found in Phillips (1977), Goodman and Spence (1978), Lodding (1983), Swezey Davis (1983), Carroll and Carrithers (1984), Foley, Wallace, and Chan (1984), and Go al. (1984).

The evolution of the concept of logical (or virtual) input devices is discussed in Wa (1976) and in Rosenthal et al. (1982). An early discussion of input-device classificatio to be found in Newman (1968).

Input operations in PHIGS can be found in Hopgood and Duce (1991), Howard e (1991), Gaskins (1992), and Blake (1993). For information on GKS input functions Hopgood et al. (1983) and Enderle, Kansy, and Pfaff (1984).

EXERCISES

8-1. Select some graphics application with which you are familiar and set up a user m that will serve as the basis for the design of a user interface for graphics applicatio that area.

8-2. List possible help facilities that can be provided in a user interface and discuss w types of help would be appropriate for different levels of users.

8-3. Summarize the possible ways of handling backup and errors. State which approa are more suitable for the beginner and which are better suited to the experienced

8-4. List the possible formats for presenting menus to a user and explain under wha cumstances each might be appropriate.

8-5. Discuss alternatives for feedback in terms of the various levels of users.

8-6. List the functions that must be performed by a window manager in handling sc layouts with multiple overlapping windows.

Set up a design for a window-manager package.

Design a user interface for a painting program.

Design a user interface for a two-level hierarchical modeling package.

For any area with which you are familiar, design a complete user interface to a graphics package providing capabilities to any users in that area.

Develop a program that allows objects to be positioned on the screen using a locator device. An object menu of geometric shapes is to be presented to a user who is to select an object and a placement position. The program should allow any number of objects to be positioned until a "terminate" signal is given.

Extend the program of the previous exercise so that selected objects can be scaled and rotated before positioning. The transformation choices and transformation parameters are to be presented to the user as menu options.

Write a program that allows a user to interactively sketch pictures using a stroke device.

Discuss the methods that could be employed in a pattern-recognition procedure to match input characters against a stored library of shapes.

Write a routine that displays a linear scale and a slider on the screen and allows numeric values to be selected by positioning the slide along the scale line. The number value selected is to be echoed in a box displayed near the linear scale.

Write a routine that displays a circular scale and a pointer or a slider that can be moved around the circle to select angles (in degrees). The angular value selected is to be echoed in a box displayed near the circular scale.

Write a drawing program that allows users to create a picture as a set of line segments drawn between specified endpoints. The coordinates of the individual line segments are to be selected with a locator device.

Write a drawing package that allows pictures to be created with straight line segments drawn between specified endpoints. Set up a gravity field around each line in a picture, as an aid in connecting new lines to existing lines.

Modify the drawing package in the previous exercise that allows lines to be constrained horizontally or vertically.

Develop a drawing package that can display an optional grid pattern so that selected screen positions are rounded to grid intersections. The package is to provide line-drawing capabilities, with line endpoints selected with a locator device.

Write a routine that allows a designer to create a picture by sketching straight lines with a rubber-band method.

Write a drawing package that allows straight lines, rectangles, and circles to be constructed with rubber-band methods.

Write a program that allows a user to design a picture from a menu of basic shapes by dragging each selected shape into position with a pick device.

Design an implementation of the input functions for request mode.

Design an implementation of the sample-mode input functions.

Design an implementation of the input functions for event mode.

Set up a general implementation of the input functions for request, sample, and event modes.

CHAPTER

9 Three-Dimensional Concepts

When we model and display a three-dimensional scene, there are many more considerations we must take into account besides just including coordinate values for the third dimension. Object boundaries can be constructed with various combinations of plane and curved surfaces, and we sometimes need to specify information about object interiors. Graphics packages often provide routines for displaying internal components or cross-sectional views of solid objects. Also, some geometric transformations are more involved in three-dimensional space than in two dimensions. For example, we can rotate an object about an axis with any spatial orientation in three-dimensional space. Two-dimensional rotations, on the other hand, are always around an axis that is perpendicular to the *xy* plane. Viewing transformations in three dimensions are much more complicated because we have many more parameters to select when specifying how a three-dimensional scene is to be mapped to a display device. The scene description must be processed through viewing-coordinate transformations and projection routines that transform three-dimensional viewing coordinates onto two-dimensional device coordinates. Visible parts of a scene, for a selected view, must be identified; and surface-rendering algorithms must be applied if a realistic rendering of the scene is required.

9-1 THREE-DIMENSIONAL DISPLAY METHODS

To obtain a display of a three-dimensional scene that has been modeled in world coordinates, we must first set up a coordinate reference for the "camera". This coordinate reference defines the position and orientation for the plane of the camera film (Fig. 9-1), which is the plane we want to use to display a view of the objects in the scene. Object descriptions are then transferred to the camera reference coordinates and projected onto the selected display plane. We can then display

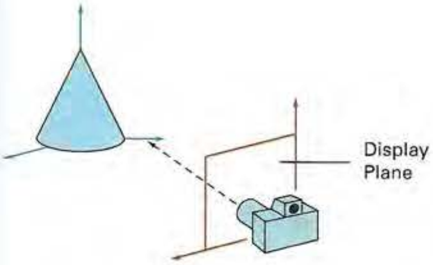


Figure 9-1
Coordinate reference for obtaining a particular view of a three-dimensional scene.

the objects in wireframe (outline) form, as in Fig. 9-2, or we can apply lig and surface-rendering techniques to shade the visible surfaces.

Parallel Projection

One method for generating a view of a solid object is to project points on th ject surface along parallel lines onto the display plane. By selecting diff viewing positions, we can project visible points on the object onto the di plane to obtain different two-dimensional views of the object, as in Fig. 9-3 *parallel projection*, parallel lines in the world-coordinate scene project into pa lines on the two-dimensional display plane. This technique is used in engi ing and architectural drawings to represent an object with a set of views maintain relative proportions of the object. The appearance of the solid objec then be reconstructured from the major views.

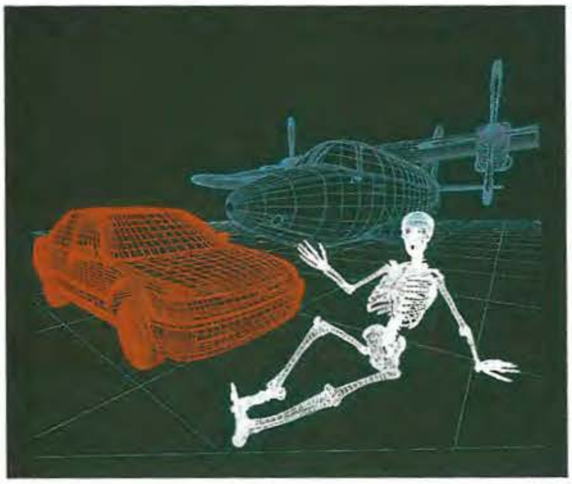


Figure 9-2
Wireframe display of three objects, with back lines removed, from a commercial database of object shapes. Each object in the database is defined as a grid of coordinate points, which can then be viewed in wireframe form or in a surface-rendered form. (*Courtesy of Viewpoint DataLabs.*)

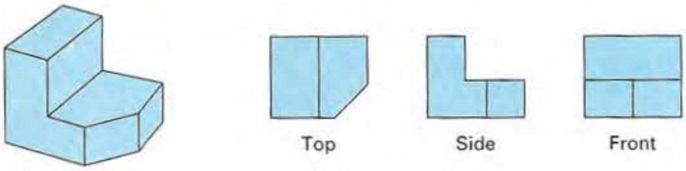


Figure 9-3
Three parallel-projection views of an object, showing relative proportions from different viewing positions.

Perspective Projection

Another method for generating a view of a three-dimensional scene is to project points to the display plane along converging paths. This causes objects farther from the viewing position to be displayed smaller than objects of the same size that are nearer to the viewing position. In a *perspective projection*, parallel lines in a scene that are not parallel to the display plane are projected into converging lines. Scenes displayed using perspective projections appear more realistic, since this is the way that our eyes and a camera lens form images. In the perspective-projection view shown in Fig. 9-4, parallel lines appear to converge to a distant point in the background, and distant objects appear smaller than objects closer to the viewing position.

Depth Cueing

With few exceptions, depth information is important so that we can easily identify, for a particular viewing direction, which is the front and which is the back of displayed objects. Figure 9-5 illustrates the ambiguity that can result when a wireframe object is displayed without depth information. There are several ways in which we can include depth information in the two-dimensional representation of solid objects.

A simple method for indicating depth with wireframe displays is to vary the intensity of objects according to their distance from the viewing position. Figure 9-6 shows a wireframe object displayed with *depth cueing*. The lines closest to

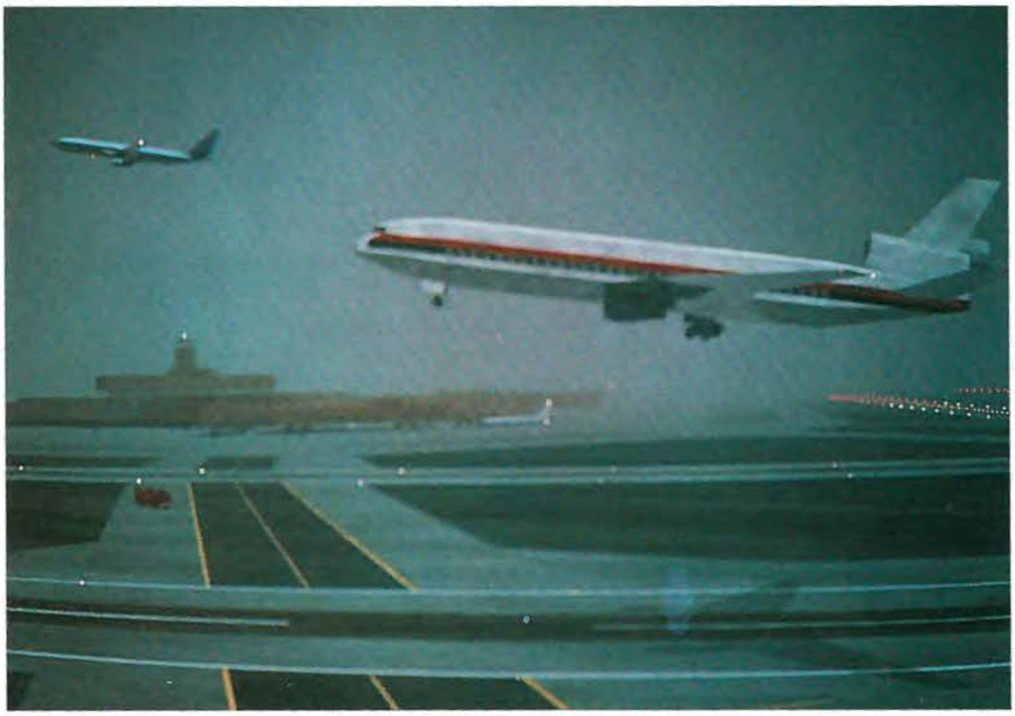


Figure 9-4
A perspective-projection view of an airport scene. *(Courtesy of Evans & Sutherland.)*

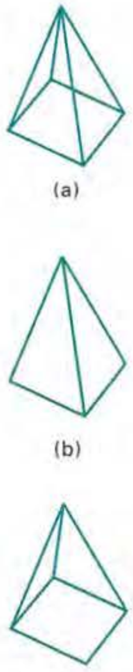


Figure 9-5
The wireframe representation of the pyramid in (a) contains no depth information to indicate whether the viewing direction is (b) downward from a position above the apex or (c) upward from a position below the base.

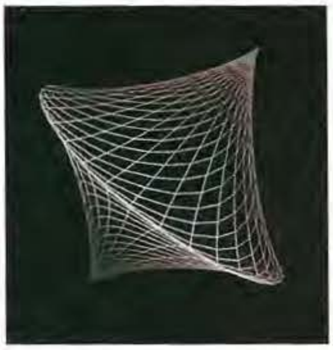


Figure 9-6
A wireframe object displayed with depth cueing, so that the intensity of lines decreases from the front to the back of the object.

the viewing position are displayed with the highest intensities, and lines fa
away are displayed with decreasing intensities. Depth cueing is applie
choosing maximum and minimum intensity (or color) values and a range o
tances over which the intensities are to vary.

Another application of depth cueing is modeling the effect of the at
phere on the perceived intensity of objects. More distant objects appear dim
to us than nearer objects due to light scattering by dust particles, haze
smoke. Some atmospheric effects can change the perceived color of an object
we can model these effects with depth cueing.

Visible Line and Surface Identification

We can also clarify depth relationships in a wireframe display by identifying
ble lines in some way. The simplest method is to highlight the visible lines
display them in a different color. Another technique, commonly used for e
neering drawings, is to display the nonvisible lines as dashed lines. Anothe
proach is to simply remove the nonvisible lines, as in Figs. 9-5(b) and 9-5(c
removing the hidden lines also removes information about the shape of the
surfaces of an object. These visible-line methods also identify the visible sur
of objects.

When objects are to be displayed with color or shaded surfaces, we a
surface-rendering procedures to the visible surfaces so that the hidden sur
are obscured. Some visible-surface algorithms establish visibility pixel by p
across the viewing plane; other algorithms determine visibility for object sur
as a whole.

Surface Rendering

Added realism is attained in displays by setting the surface intensity of ob
according to the lighting conditions in the scene and according to assigned
face characteristics. Lighting specifications include the intensity and positio
light sources and the general background illumination required for a scene.
face properties of objects include degree of transparency and how roug
smooth the surfaces are to be. Procedures can then be applied to generate the
rect illumination and shadow regions for the scene. In Fig. 9-7, surface-rende
methods are combined with perspective and visible-surface identification to
erate a degree of realism in a displayed scene.

Exploded and Cutaway Views

Many graphics packages allow objects to be defined as hierarchical structure
that internal details can be stored. Exploded and cutaway views of such ob
can then be used to show the internal structure and relationship of the c
parts. Figure 9-8 shows several kinds of exploded displays for a mechanica
sign. An alternative to exploding an object into its component parts is the
away view (Fig. 9-9), which removes part of the visible surfaces to show inte
structure.

Three-Dimensional and Stereoscopic Views

Another method for adding a sense of realism to a computer-generated sce
to display objects using either three-dimensional or stereoscopic views. As
have seen in Chapter 2, three-dimensional views can be obtained by reflecti

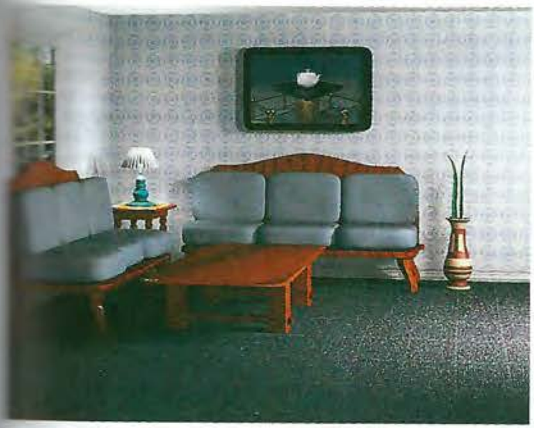


Figure 9-7
A realistic room display achieved with stochastic ray-tracing methods that apply a perspective projection, surface-texture mapping, and illumination models. (*Courtesy of John Snyder, Jed Lengyel, Devendra Kalra, and Al Barr, California Institute of Technology.* Copyright © 1992 Caltech.)

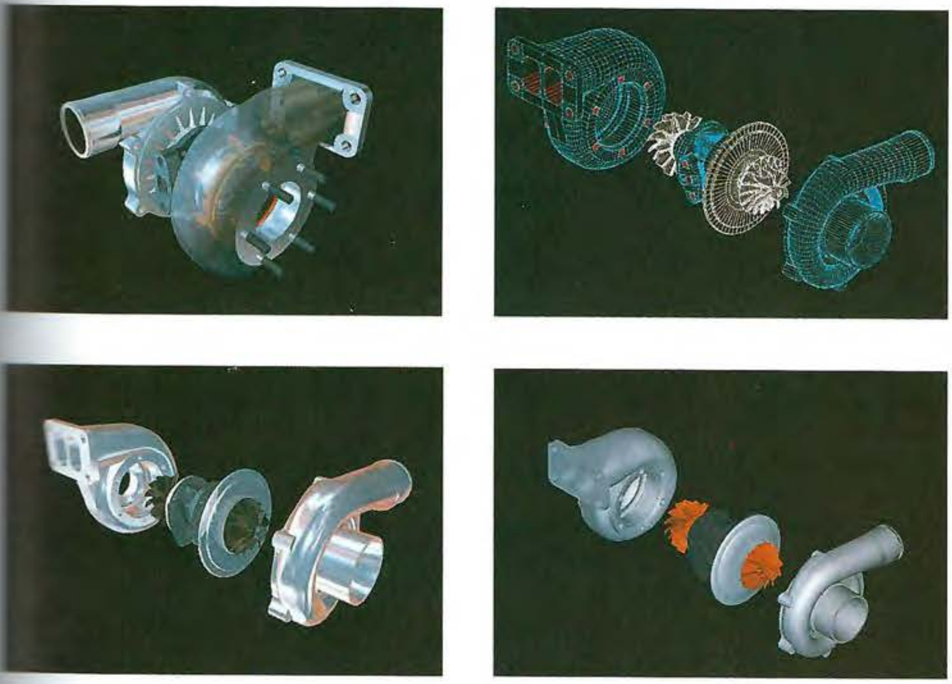


Figure 9-8
A fully rendered and assembled turbine display (a) can also be viewed as (b) an exploded wireframe display, (c) a surface-rendered exploded display, or (d) a surface-rendered, color-coded exploded display. (*Courtesy of Autodesk, Inc.*)

raster image from a vibrating flexible mirror. The vibrations of the mirror are synchronized with the display of the scene on the CRT. As the mirror vibrates, the focal length varies so that each point in the scene is projected to a position corresponding to its depth.

Stereoscopic devices present two views of a scene: one for the left eye and the other for the right eye. The two views are generated by selecting viewing positions that correspond to the two eye positions of a single viewer. These two views then can be displayed on alternate refresh cycles of a raster monitor, and viewed through glasses that alternately darken first one lens then the other in synchronization with the monitor refresh cycles.

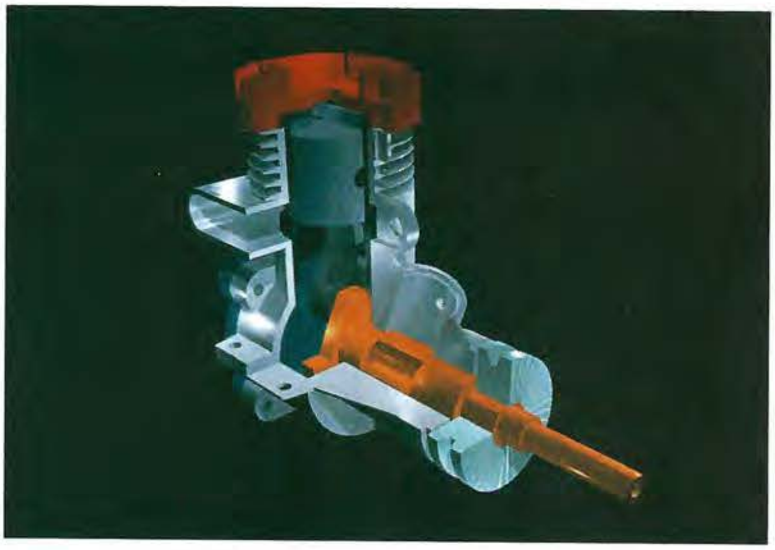


Figure 9-9
Color-coded cutaway view of a lawn mower engine showing the structure and relationship of internal components. (*Courtesy of Autodesk, Inc.*)

9-2
THREE-DIMENSIONAL GRAPHICS PACKAGES

Design of three-dimensional packages requires some considerations that are n necessary with two-dimensional packages. A significant difference between t two packages is that a three-dimensional package must include methods f mapping scene descriptions onto a flat viewing surface. We need to consider i plementation procedures for selecting different views and for using different p jection techniques. We also need to consider how surfaces of solid objects are be modeled, how visible surfaces can be identified, how transformations of o jects are performed in space, and how to describe the additional spatial prope ties introduced by three dimensions. Later chapters explore each of these consi erations in detail.

Other considerations for three-dimensional packages are straightforwa extensions from two-dimensional methods. World-coordinate descriptions a extended to three dimensions, and users are provided with output and input ro tines accessed with specifications such as

```
    polyline3 (n, wcPoints)
    fillarea3 (n, wcPoints)
    text3 (wcPoint, string)
    getLocator3 (wcPoint)
    translate3(translateVector, matrixTranslate)
```

where points and vectors are specified with three components, and transform tion matrices have four rows and four columns.

Two-dimensional attribute functions that are independent of geometric co siderations can be applied in both two-dimensional and three-dimensional app cations. No new attribute functions need be defined for colors, line styles, mark

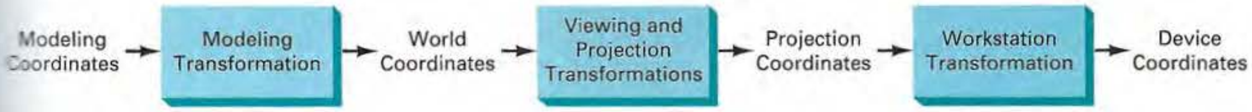


Figure 9-10
Pipeline for transforming a view of a world-coordinate scene to device coordinates.

attributes, or text fonts. Attribute procedures for orienting character strings, however, need to be extended to accommodate arbitrary spatial orientations. Text-attribute routines associated with the up vector require expansion to include *z*-coordinate data so that strings can be given any spatial orientation. Area-filling routines, such as those for positioning the pattern reference point and for mapping patterns onto a fill area, need to be expanded to accommodate various orientations of the fill-area plane and the pattern plane. Also, most of the two-dimensional structure operations discussed in earlier chapters can be carried over to a three-dimensional package.

Figure 9-10 shows the general stages in a three-dimensional transformation pipeline for displaying a world-coordinate scene. After object definitions have been converted to viewing coordinates and projected to the display plane, scan-conversion algorithms are applied to store the raster image.