MASTER OF COMPUTER APPLICATION

MCA-44

Computer Graphics

Directorate of Distance Education
Guru Jambheshwar University of Science & Technology
Hisar - 125001

CONTENTS

SUBJECT: COMPUTER GRAPHICS	
COURSE CODE: MCA-44(iv)	**AUTHOR: Abhishek Taneja**
LESSON NO. 1	
Overview of Computer Graphics	

UNIT STRUCTURE

1.1 Objectives

At the end of this chapter the reader will be able to:

- Describe Computer Graphics and its applications.
- Describe and distinguish between Interactive and Passive Graphics.
- Describe advantages of Interactive Graphics.
- Describe applications of Computer Graphics.

1.1 Introduction

The term **computer graphics** includes almost everything on computers that is not text or sound. Today almost every computer can do some graphics, and people have even come to expect to control their computer through icons and pictures rather than just by typing. Here in our lab at the Program of Computer Graphics, we think of computer graphics as drawing pictures on computers, also called rendering. The pictures can be photographs, drawings, movies, or simulations - pictures of things, which do not yet exist and maybe could never exist. Or they may be pictures from places we cannot see directly, such as medical images from inside your body. We spend much of our time improving the way computer pictures can simulate real world scenes. We want images on computers to not just look more realistic, but also to be more realistic in their colors, the way objects and rooms are lighted, and the way different materials appear. We call this work “realistic image synthesis”.

1.2 Interactive Graphics

In interactive computer graphics user have some control over the picture i.e user can make any change in the produced image. One example of it is the ping pong game. The conceptual model of any interactive graphics system is given in the picture shown in Figure 1.1. At the hardware level (*not shown in picture*), a computer receives input from interaction devices, and outputs images to a display device. The software has three components. The first is the application program, it creates, stores into, and retrieves from the second component, the application model, which represents the the graphic primitive to be shown on the screen. The application program also handles user input. It produces views by sending to the third component, the graphics system, a series of graphics output commands that contain both a detailed geometric description of what is to be viewed and the attributes describing how the objects

should appear. After the user input is processed, it sent to the graphics system is for actually producing the picture. Thus the graphics system is a layer in between the application program and the display hardware that effects an output transformation from objects in the application model to a view of the model.

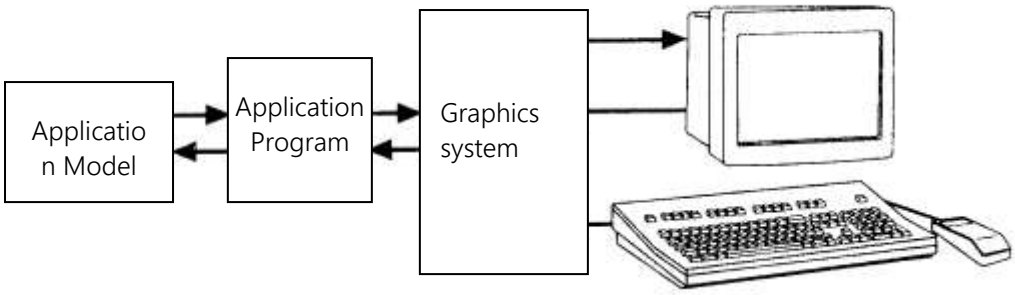


Figure 1.1: Conceptual model for interactive graphics

The objective of the application model is to captures all the data, objects, and relationships among them that are relevant to the display and interaction part of the application program and to any nongraphical postprocessing modules.

1.2 Passive Graphics

A computer graphics operation that transfers automatically and without operator intervention. Non-interactive computer graphics involves one-way communication between the computer and the user. Picture is produced on the monitor and the user does not have any control over the produced picture.

1.3 Advantages of Interactive Graphics

Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D and 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly and efficiently. In Many design, implementation, and construction processes today, the information pictures can give is virtually indispensable. Scientific visualization became an important field in the late 1980s, when scientists and engineers realized that they could not interpret the data and prodigious quantities of data produced in supercomputer runs without summarizing the data and highlighting trends and phenomena in various kinds of graphical representations.

Creating and reproducing pictures, however, presented technical problems that stood in the way of their widespread use. Thus, the ancient Chinese proverb “a picture is worth ten thousand words” became a cliché in our society only after the advent of inexpensive and simple technology for producing pictures—first the printing press, then photography.

Interactive computer graphics is the most important means of producing pictures since the invention of photography and television; it has the added advantage that, with the computer, we can make pictures not only of concrete, “real-world” objects but also of abstract, synthetic objects, such as mathematical surfaces in 4D and of data that have no inherent geometry, such as survey results. Furthermore, we are not confined to static images. Although static pictures are a good means of communicating information, dynamically varying pictures are frequently even better–to time-varying phenomena, both real (e.g., growth trends, such as nuclear energy use in the United States or population movement form cities to suburbs and back to the cities). Thus, a movie can show changes over time more graphically than can a sequence of slides. Thus, a sequence of frames displayed on a screen at more than 15 frames per second can convey smooth motion or changing form better than can a jerky sequence, with several seconds between individual frames. The use of dynamics is especially effective when the user can control the animation by adjusting the speed, the portion of the total scene in view, the amount of detail shown, the geometric relationship of the objects in the another, and so on. Much of interactive graphics technology therefore contains hardware and software for user-controlled motion dynamics and update dynamics.

With motion dynamics, objects can be moved and tumbled with respect to a stationary observer. The objects can also remain stationary and the viewer can move around them, pan to select the portion in view, and zoom in or out for more or less detail, as though looking through the viewfinder of a rapidly moving video camera. In many cases, both the objects and the camera are moving. A typical example is the flight simulator, which combines a mechanical platform supporting a mock cockpit with display screens for windows. Computers control platform motion, gauges, and the simulated world of both stationary and moving objects through which the pilot navigates. These multimillion-dollar systems train pilots by letting the pilots maneuver a simulated craft over a simulated 3D landscape and around simulated

vehicles. Much simpler fight simulators are among the most popular games on personal computers and workstations. Amusement parks also offer "motion-simulator" rides through simulated terrestrial and extraterrestrial landscapes. Video arcades offer graphics-based dexterity games and racecar-driving simulators, video games exploiting interactive motion dynamics: The player can change speed and direction with the "gas pedal" and "steering wheel," as trees, buildings, and other cars go whizzing by. Similarly, motion dynamics lets the user fly around the through buildings, molecules, and 3D or 4D mathematical space. In another type of motion dynamics, the "camera" is held fixed, and the objects in the scene are moved relative to it. For example, a complex mechanical linkage, such as the linkage on a stream engine, can be animated by moving or rotating all the pieces appropriately.

Update dynamics is the actual change of the shape, color, or other properties of the objects being viewed. For instance, a system can display the deformations of an airplane structure in flight or the state changes in a block diagram of a nuclear reactor in response to the operator's manipulation of graphical representations of the many control mechanisms. The smoother the change, the more realistic and meaningful the result. Dynamic interactive graphics offers a large number of user-controllable modes with which to encode and communicate information: the 2D or 3D shape of objects in a picture, their gray scale or color, and the time variations of these properties. With the recent development of digital signal processing (DSP) and audio synthesis chips, audio feedback can now be provided to augment the graphical feedback and to make the simulated environment even more realistic.

Interactive computer graphics thus permits extensive, high-bandwidth user-computer interaction. This significantly enhances our ability to understand data, to perceive trends, and to visualize real or imaginary objects–indeed, to create "virtual worlds" that we can explore from arbitrary points of view. By making communication more efficient, graphics make possible higher-quality and more precise results or products, greater productivity, and lower analysis and design costs.

1.4 How The Interactive Graphics Display Works

The modern graphic display is very simple in construction. It consists of the three components shown in figure 1.2 below.

(1) **Frame Buffer**

(2) **Monitor** like a TV set without the tuning and receiving electronics.

(3) **Display Controller** It passes the contents of the frame buffer to the monitor.

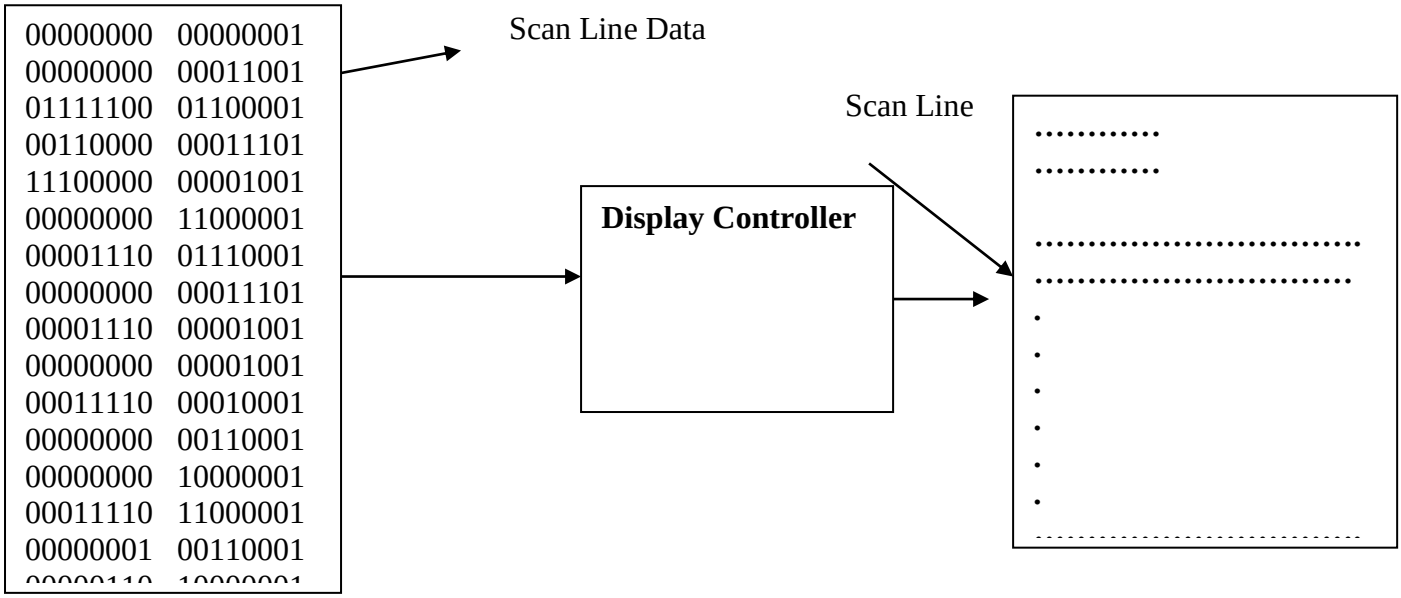


Figure 1.2

Inside the frame buffer the image is stored as a pattern of binary digital numbers, which represent a array of picture elements, or pixels. In the simplest case, where you want to store only black and white images, you can represent black pixels by "1's" and white pixels by "0's" in the frame buffer. Therefore, a array of black and white pixels of 16X16 could be represented by 32 bytes, stored in frame buffer.

The display controller reads each successive byte of data from the frame buffer and converts its 0's and 1's into corresponding video signals. This signal is then fed to the monitor, producing a black and white image on the screen. The display controller repeats this operation 30 times a second to maintain a steady picture on the monitor. If you want to change the image, then you need to modify the frame buffer's contexts to represent the new pattern of pixels.

1.5 Applications of Computer Graphics

Classification of Applications

The diverse uses of computer graphics listed in the previous section differ in a variety of ways, and a number of classification is by type (dimensionality) of the object to be represented and the kind of picture to be produced. The range of possible combinations is indicated in Table 1.1.

Classification of Computer Graphics by Object and
Picture
Type of Object
2D
3D
Pictorial Representation
Line drawing
Gray scale image
Color image
Line drawing (or *wire-frame)*
Line drawing, with various effects Shaded, color image with
various effects

Table 1.1

Computer graphics is used today in many different areas of industry, business, government, education, entertainment, and most recently, the home. The list of applications is enormous and is growing rapidly as computers with graphics capabilities become commodity products. Let`s look at a representative sample of these areas.

- Cartography: Computer graphics is used to produce both accurate and schematic representations of geographical and other natural phenomena from measurement data. Examples include geographic maps, relief maps, exploration maps for drilling and mining, oceanographic charts, weather maps, contour maps, and population-density maps.

- User interfaces: As soon mentioned, most applications that run on personal computers and workstations, and even those that run on terminals attached to time shared computers and network compute servers, have user interfaces that rely on desktop window systems to manage multiple simultaneous activities, and on point and click facilities to allow users to select menu items, icons, and objects on the screen; typing is necessary only to input text to be stored and manipulated. Word-

processing, spreadsheet, and desktop-publishing programs are typical applications that take

- Advantage of such user-interface techniques: The authors of this book used such programs to create both the text and the figures; then, the publisher and their contractors produced the book using similar typesetting and drawing software.

- (Interactive) plotting in business, science and technology: The next most common use of graphics today is probably to create 2D and 3D graphs of mathematical, physical, and economic functions; histograms, bar and pie charts; task-scheduling charts; inventory and production charts, and the like. All these are used to present meaningfully and concisely the trends and patterns gleaned from data, so as to clarify complex phenomena and to facilitate informed decision making.

- Office automation and electronic publishing: The use of graphics for the creation and dissemination of information has increased enormously since the advent of desktop publishing on personal computers. Many organizations whose publications used to be printed by outside specialists can now produce printed materials inhouse. Office automation and electronic publishing can produce both traditional printed (hardcopy) documents and electronic (softcopy) documents that allow browsing of networks of interlinked multimedia documents are proliferating

- Computer-aided drafting and design: In computer-aided design (CAD), interactive graphics is used to design components and systems of mechanical, electrical, electromechanical, and electronic devices, including structure such as buildings, automobile bodies, airplane and ship hulls, very large scale-integrated (VLSI) chips, optical systems, and telephone and computer networks. Sometimes, the use; merely wants to produce the precise drawings of components and assemblies, as for online drafting or architectural blueprints Color Plate 1.8 shows an example of such a 3D design program, intended for nonprofessionals also a customize your own patio deck" program used in lumber yards. More frequently however the emphasis is on interacting with a computer based model of the component or system being designed in order to test, for example, its structural, electrical, or thermal properties. Often, the model is interpreted by a simulator that feeds back the behavior of the system to the user for further interactive design and test cycles. After objects have been designed, utility programs can postprocess the design

database to make parts lists, to process ‘bills of materials’, to define numerical control tapes for cutting or drilling parts, and so on.

- Simulation and animation for scientific visualization and entertainment: Computer produced animated movies and displays or the time-varying behavior of real and simulated objects are becoming increasingly popular for scientific and engineering visualization. We can use them to study abstract mathematical entries as well as mathematical models of such phenomena as fluid flow, relativity, nuclear and chemical reactions, physiological system and organ function, and deformation of mechanical structures under various kinds of loads. Another advanced-technology area is interactive cartooning. The simpler kinds of systems for producing ‘Flat” cartons are becoming cost-effective in creating routine ‘in-between” frames that interpolate between two explicity specified ‘key frames”. Cartoon characters will increasingly be modeled in the computer as 3D shape descriptions whose movements are controlled by computer commands, rather than by the figures being drawn manually by cartoonists. Television commercials featuring flying logos and more exotic visual trickery have become common, as have elegant special effects in movies. Sophisticated mechanisms are available to model the objects and to represent light and shadows.

- Art and commerce: Overlapping the previous categories the use of computer graphics in art and advertising here, computer graphics is used to produce pictures that express a message and attract attention. Personal computers and Teletext and Videotexts terminals in public places such as in private homes, offer much simpler but still informative pictures that let users orient themselves, make choices, or even “teleshop” and conduct other business transactions. Finally, slide production for commercial, scientific, or educational presentations is another cost-effective use of graphics, given the steeply rising labor costs of the traditional means of creating such material.

- Process control: Whereas flight simulators or arcade games let users interact with a simulation of a real or artificial world, many other applications enable people or interact with some aspect of the real world itself. Status displays in refineries, power plants, and computer networks show data values from sensors attached to critical system components, so that operators can respond to problematic

conditions. For example, military commanders view field data – number and position of vehicles, weapons launched, troop movements, causalities – on command and control displays to revise their tactics as needed; flight controller airports see computer-generated identification and status information for the aircraft blips on their radar scopes, and can thus control traffic more quickly and accurately than they could with the uninitiated radar data alone; spacecraft controllers monitor telemetry data and take corrective action as needed.

1.6 Summary

- Computer graphics includes the process and outcomes associated with using computer technology to convert created or collected data into visual representations.

- Graphical interfaces have replaced textual interfaces as the standard means for user-computer interaction. Graphics has also become a key technology for communicating ideas, data, and trends in most areas of commerce, science, engineering, and education with graphics, we can create artificial realities, each a computer-based "exploratorium" for examining objects and phenomena in a natural and intuitive way that exploits our highly developed skills in visual-pattern recognition.

- Until the late eighties, the bulk of computer-graphics applications dealt with 2D objects, 3D applications were relatively rare, both because 3D software is intrinsically far more complex than is 2D software and because a great deal of computing power is required to render pseudorealistic images. Therefore, until recently, real-time user interaction with 3D models and pseudorealistic images was feasible on only very expensive high-performance workstations with dedicated, special-purpose graphics hardware. The spectacular progress of VLSI semiconductor technology that was responsible for the advent of inexpensive microprocessors and memory led in the early 1980s to the creation of 2D, bitmap-graphics-based personal computers. That same technology has made it possible, less than a decade later, to create subsystems of only a few chips that do real-time 3D animation with color-shaded images of complex objects, typically described by thousands of polygons. These subsystems can be added as 3D accelerators to workstations or even to personal computers using commodity microprocessors It is

clear that an explosive growth of 3D applications will parallel the current growth in applications.

- Much of the task of creating effective graphic communication, whether 2D or 3D, lies in modeling the objects whose images we want to produce. The graphics system acts as the intermediary between the application model and the output device. The application program is responsible for creating and updating the model based on user interaction; the graphics system does the best-understood, most routine part of the job when it creates views of objects and passes user events to the application.

1.7 Keywords

Computer Graphics: Computer graphics refers to the field of study and practice that involves creating, manipulating, and displaying visual content using computers. It encompasses various techniques and technologies to generate and render images, animations, and interactive graphics.

Interactive Graphics: Interactive graphics refers to computer-generated visual content that allows users to actively engage with and manipulate the displayed elements. It involves real-time rendering and user input to create dynamic and responsive graphical experiences. Interactive graphics can be found in various applications, including video games, virtual reality, user interfaces, data visualization, and educational software.

Passive Graphics: Passive graphics refers to computer-generated visual content that is displayed to users without any direct interactivity or user input. Unlike interactive graphics, passive graphics are pre-rendered and do not respond to user actions or dynamically change based on user input in real-time. Instead, the visual content is typically designed to be viewed and observed rather than actively manipulated.

1.8 Self-Assessment Questions

1. Write a short note on the interactive computer graphics.
2. Discuss the relative advantage of interactive and passive graphics.
3. What are the applications of computer graphics?

4. Nominate an application of computers that can be accommodated by either textual or graphical computer output. Explain when and why graphics output would be more appropriate in this application.
5. Explain briefly the classification of computer graphics.

1.9 References/Suggested Readings

1. Computer Graphics, Principles and Practice, Second Edition, by James D. Foley, Andries van Dam, Steven K. Feiner, John F. Hughes, Addison- Wesley
2. Computer Graphics, Second Edition, by Pradeep K. Bhatia, I. K. International Publisher.
3. High Resolution Computer Graphics using Pascal/C, by Ian O. Angell and Gareth Griffith, John Wiley & Sons
4. Computer Graphics (C Version), Donald Hearn and M. Pauline Baker, Prentice Hall,
5. Advanced Animation and Rendering Techniques, Theory and Practice, Alan Watt and Mark Watt, ACM Press/Addison-Wesley
6. Graphics Gems I-V, various authors, Academic Press
7. Computer Graphics, Plastok, TMH
8. Principles of Interactive Computer Graphics, Newman, TMH

SUBJECT: COMPUTER GRAPHICS

COURSE CODE: MCA-44(iv)	AUTHOR: Abhishek Taneja

LESSON NO. 2

Display Devices

UNIT STRUCTURE

1.0 Objectives

At the end of this chapter the reader will be able to:

- Describe and distinguish raster and random scan displays
- Describe various display devices.
- Describe how colour CRT works.

2.1 Introduction

The principle of producing images as collections of discrete points set to appropriate colours is now widespread throughout all fields of image production. The most common graphics output device is the video monitor which is based on the standard cathode ray tube(CRT) design, but several other technologies exist and solid state monitors may eventually predominate.

2.2 Refresh CRT

Basic Operation of CRT

Figure 2.1 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.

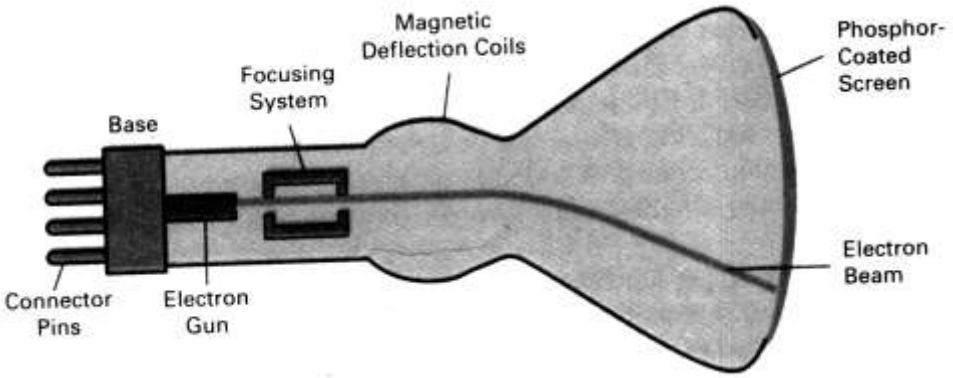


Figure 2.1: Basic Design of a magnetic-deflection CRT

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.

Working

Beam passes between two pairs of metal plates, one vertical and other horizontal. A voltage difference is applied to each pair of plates according to the amount that the beam is to be deflected in each direction. As the electron beam passes between each pair of plates, it is bent towards the plate with the higher positive voltage. In figure 2.2 the beam is first deflected towards one side of the screen. Then, as the beam passes through the horizontal plates, it is deflected towards, the top or bottom of the screen. To get the proper deflection, adjust the current through coils placed around the outside of the CRT loop. The primary components of an electron gun in a CRT are the heated metal cathode and a control grid (Fig. 2.2). 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 accelerating voltage can be generated with a positively charged metal coating on the in- side of the CRT envelope near the phosphor screen, or an accelerating anode can be used, as in Fig. 2.2. Sometimes the electron gun is built to contain the accelerating anode and focusing system within the same unit.

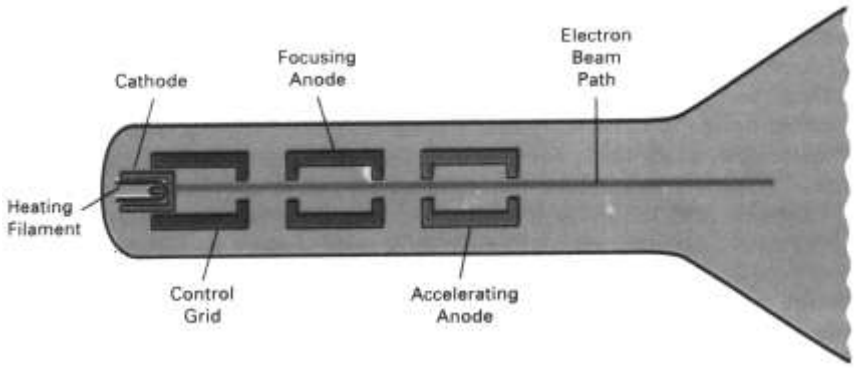


Figure 2.2: Operation of an electron gun with an acceleration anode

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.

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. 2.1. 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. 2.3).

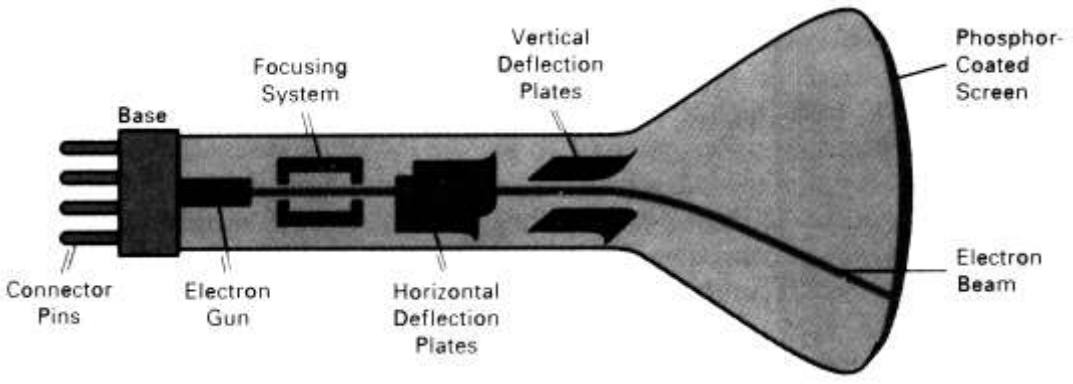


Figure 2.3: Electrostatic deflection of the electron beam in a CRT

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 phosphor coating, they are stopped and their kinetic energy is absorbed by the phophor. 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 quanturn-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.

Figure 2.4 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 2.4: Intensity distribution of an illuminated phosphor spot on a CRT screen

Resolution

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. This depends on the type of phosphor used and the focusing and deflection system.

Aspect Ratio

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.

2.3 Random-Scan and Raster Scan Monitor

2.3.1 Random-Scan/Calligraphic displays

Random scan system uses an electron beam which operates like a pencil to create a line image on the CRT. The image is constructed out of a sequence of straight line segments. Each line segment is drawn on the screen by directing the beam to move from one point on screen to the next, where each point is defined by its x and y coordinates. After drawing the picture, the system cycles back to the first line and design all the lines of the picture 30 to 60 time each second. 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) Fig. 2.5. A pen plotter operates in a similar way and is an example of a random-scan, hard-copy device.

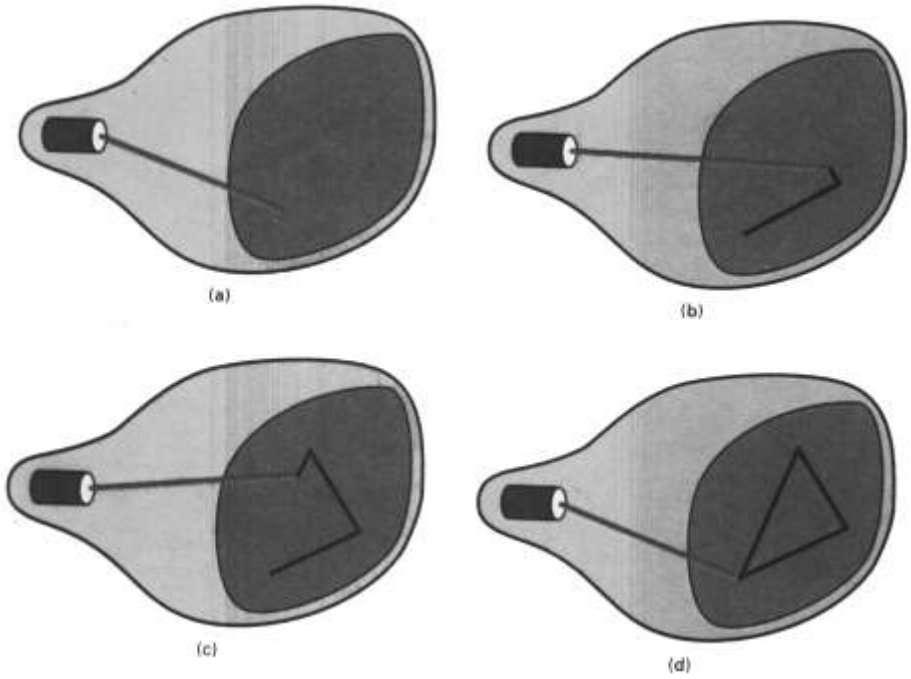


Figure 2.5: A random-scan system draws the component lines of an object in any order specified

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. Random-scan systems are designed for line-drawing applications and can-not 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.

2.3.2 Raster-Scan Displays

In raster scan approach, the viewing screen is divided into a large number of discrete phosphor picture elements, called pixels. The matrix of pixels constitutes the raster. The number of separate pixels in the raster display might typically range from 256X256 to 1024X 1024. Each pixel on the screen can be made to glow with a different brightness. Colour screen provide for the pixels to have different colours as well as brightness. 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. 2.6). 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.

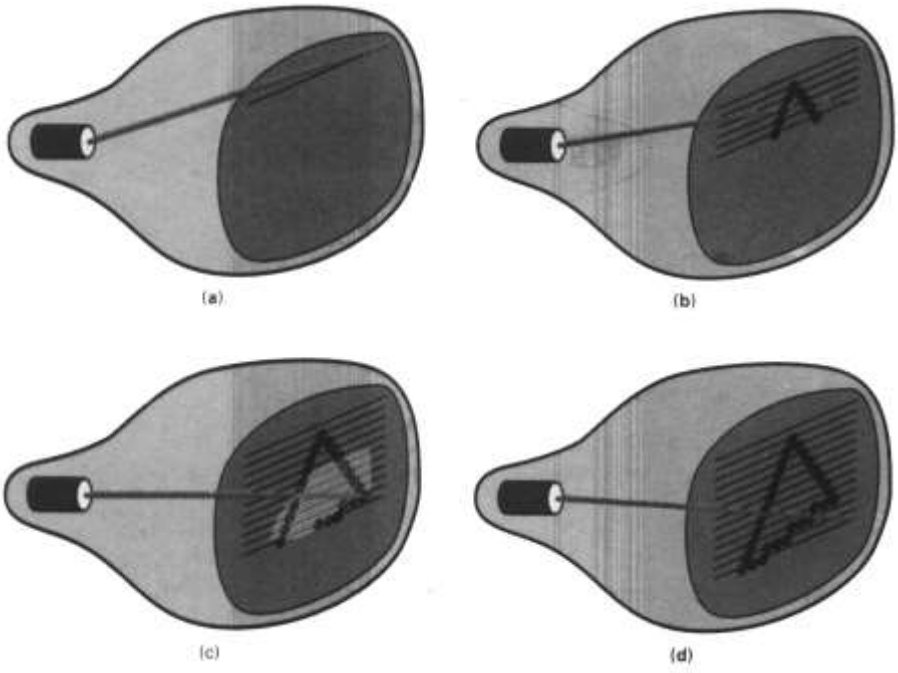


Figure 2.6: A raster-scan system displays an object as a set of discrete points across each scan line

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. 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 re- trace, the beam sweeps out the remaining scan lines (Fig. 2.7). 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 l/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.

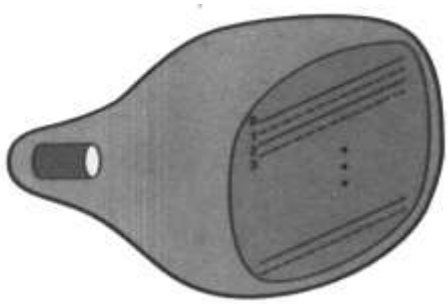


Figure 2.7: Interlacing Scan lines on a raster-scan display. First , all points on the even-numbered (solid) scan lines are displayed; then all points along the odd-numbered (dashed) lines are displayed

2.4 Color CRT Monitors

To display colour pictures, combination of phosphorus is used that emits different coloured light. There are two different techniques for producing colour displays with a CRT.

1. Beam Penetration Method
2. Shadow Mask Method

Beam Penetration 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 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 Method

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 2.8 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 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.

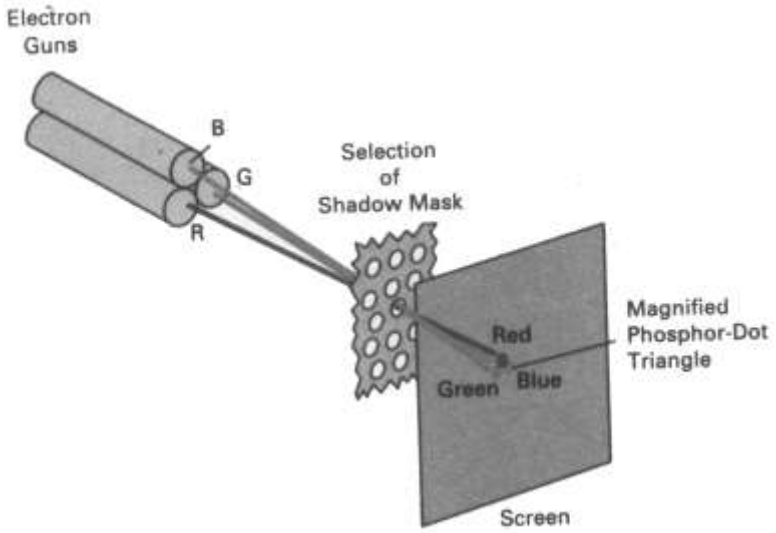


Figure 2.8: Operation of a delta–delta, shadow-mask CRT. Three electron guns, aligned with the triangular color-dot patterns on the screen, are directed to each dot triangle by a shadow mask.

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.

2.5 Direct-View Storage Tubes (DVST)

This is an alternative method to monitor a screen image, as it sores 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.

2.6 Flat-Panel Displays

The term flat panel display refers to a class of video device that have reduced volume , weight and power requirement 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.

2.7 Light-emitting Diode (LED) and Liquid-crystal Displays (LCDs)

2.7.1 Light-emitting Diode (LED)

In LED, a matrix of diodes is arranged to form the pixel positions in the display and picture definition is stored in a refresh buffer. Information 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.

2.7.2 Liquid-crystal Displays (LCDs)

Liquid crystal displays are the divices that produce a picture by passing polarized light from the surroundings or from an internal light source through a liquid crystal material that transmit the light. Liquid-crystal displays (LCDs) are commonly used in small systems, such ' as calculators and portable, laptop computers. These non-emissive 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. 2-9. 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. 2.9. 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.

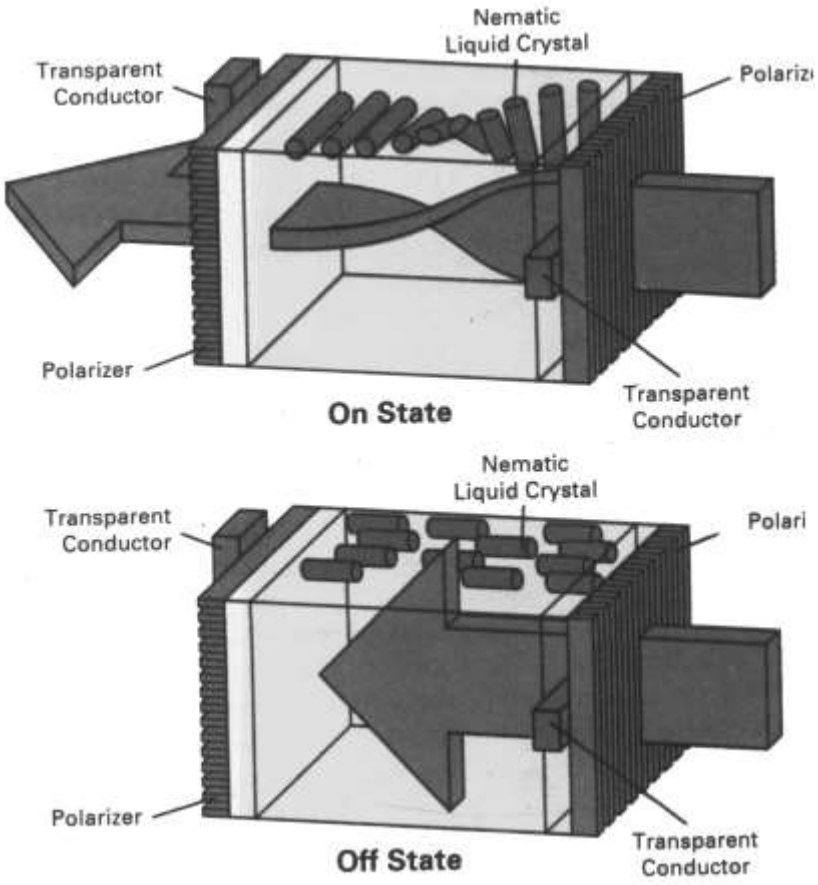


Figure 2.9: The light-twisting, shutter effect used in the design of most liquid-crystal display devices

2.8 Hard Copy Devices

The printer is an important accessory of any computing system. In a graphics system, it is the quality of printed output which is one of the key factors necessary to convince both the user and the customer. The major factors which control the quality of a printer are individual dot size on the paper and the number of dots per inch.

We can obtain hard-copy output for our images in several formats. For presentations or archiving, we can send image files to devices or service bureaus that will produce 35-mm slides or overhead transparencies. To put images on film, we can simply photograph a scene displayed on a video monitor. And we can put our pictures on paper by directing graphics output to a printer or plotter.

Printers produce output by either impact or nonimpact methods. Impact printers press formed character faces against an inked ribbon onto the paper. A line printer is an example of an impact device, with the typefaces mounted on bands, chains, drums, or wheels. Nonimpact printers and plotters use laser techniques, ink-jet sprays, xerographic processes (as used in photocopying machines), electrostatic methods, and electrothermal methods to get images onto paper.

In a laser device, a laser beam creates a charge distribution on a rotating drum coated with a photoelectric material, such as selenium. Toner is applied to the drum and then transferred to paper. Figure 2.10 shows examples of desktop laser printers with a resolution of 360 dots per inch.

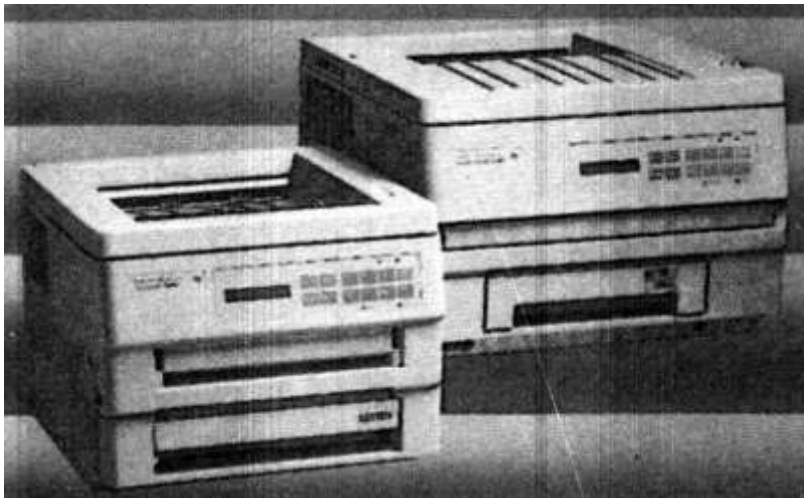


Figure 2.10: Small-footprint laser printers

Ink-jet methods produce output by squirting ink in horizontal rows across a roll of paper wrapped on a drum. The electrically charged ink stream is deflected by an electric field to produce dot-matrix patterns.

2.9 Summary

- P**ersistence** is defined as the time it takes the emitted light from screen to decay to one-tenth of its original intensity.
- This chapter have surveyed the major hardware and software features of computer graphics systems. Hardware components include video monitors, hard-copy devices, keyboards, and other devices for graphics input or output. Graphics software includes special applications packages and general programming packages.

- The dominant graphics display device is the raster refresh monitor, based on television technology. A raster system uses a frame buffer to store intensity information for each screen position (pixel). Pictures are then painted on the screen by retrieving this information from the frame buffer as the electron beam in the CRT sweeps across each scan line, from top to bottom. Older vector displays constructs pictures by drawing lines between specified line endpoints. Picture information is then stored as a set of line-drawing instructions.

- Various other video display devices are available. In particular, flat-panel display technology is developing at a rapid rate, and these devices may largely replace raster displays in the near future. At present, flat-panel displays are commonly used in the small systems and in special-purpose systems. Flat-panel displays include plasma panels and liquid-crystal devices. Although vector monitors can be used to display high-quality line drawings, improvements in raster display technology have caused vector monitors to be largely replaced with raster systems.

- Hard-copy devices for graphics workstations include standard printers and plotters, in addition to devices for producing slides, transparencies, and film output. Printing methods include dot matrix, laser, ink jet, electrostatic, and electrothermal. Plotter methods include pen plotting and combination printer-plotter devices.

2.10 Key Words

Random scan display, raster scan display, CRT, persistence, aspect ratio

2.11 Self Assessment Questions (SAQ)

1. Write a short note on hard copy devices.
2. How the colours are focused in coloured CRT? Discuss
3. Is the refreshing is necessary? Explain.
4. Discuss the detailed of DVST.
5. Explain about the display technologies?
6. Explain various display devices?
7. What are the different hardware and software of graphics?

8 List five graphic soft copy devices for each one briefly explain?

A. How it works.

B. Its advantages and limitations.

C. The circumstances when it would be more useful.

9. List five graphic hard copy devices for each one briefly explain?

a) How it works.

b) Its advantages and limitations.

c) The circumstances when it would be more useful.

2.12 References/Suggested Readings

1. Computer Graphics, Principles and Practice, Second Edition, by James D. Foley, Andries van Dam, Steven K. Feiner, John F. Hughes, Addison-Wesley
2. Computer Graphics , Second Edition , by Pradeep K. Bhatia , I.K .International Publisher.
3.
4. High Resolution Computer Graphics using Pascal/C, by Ian O. Angell and Gareth Griffith, John Wiley & Sons
5. Computer Graphics (C Version), Donald Hearn and M. Pauline Baker, Prentice Hall

SUBJECT: COMPUTER GRAPHICS

COURSE CODE: MCA-44(iv)	AUTHOR: Abhishek Taneja

LESSON NO. 3

Scan Conversion

UNIT STRUCTURE

3.0 Objectives

At the end of this chapter the reader will be able to:

- Describe scan conversion
- Describe how to scan convert basic graphic primitives like point, line, circle, ellipse

3.1 Introduction

We have studied various display devices in the previous chapter. It is clear that these devices need special procedures for displaying any graphic object: line, circle, curves, and even characters. Irrespective of the procedures used, the system can generate the images on these raster devices by turning the pixels on or off. The process in which the object is represented as the collection of discrete pixels is called **scan conversion**. The video output circuitry of a computer is capable of converting binary values stored in its display memory into pixel-on, pixel-off information that can be used by a raster output device to display a point. This ability allows graphics computers to display models composed of discrete dots.

Almost any model can be reproduced with a sufficiently dense matrix of dots (pointillism), most human operators generally think in terms of more complex graphics objects such as points, lines, circles and ellipses. Since the inception of computer graphics, many algorithms have been developed to provide human users with fast, memory-efficient routines that generate higher-level objects of this kind. However, regardless of what routines are developed, the computer can produce images on raster devices only by turning the appropriate pixels on or off. Many scan-conversion algorithms are implemented in computer hardware or firmware. However, a specific graphics algorithm, the scan-conversion algorithm can be implemented in software. The most commonly used graphics objects are the line, the sector, the arc, the ellipse, the rectangle and the polygon.

3.2 Scan-converting a Point

We have already defined that a pixel is collection of number of points. Thus it does not represent any mathematical point. Suppose we wish to display a point C(5.4, 6.5). It means that we wish to illuminate that pixel, which contains this point C. Refer to figure 3.1, which shows that pixel corresponding to point C. What happens if we try to display C'(5.3, 6.4)? Well, it also corresponding to the same pixel as that of C (5.4,6.5).Thus we can say that point

C(x,y) is represented by an integer part of x and integer part of y. So, we can use the command as

Putpixel(int x, int y);

We will now look into the actual process of plotting a point.

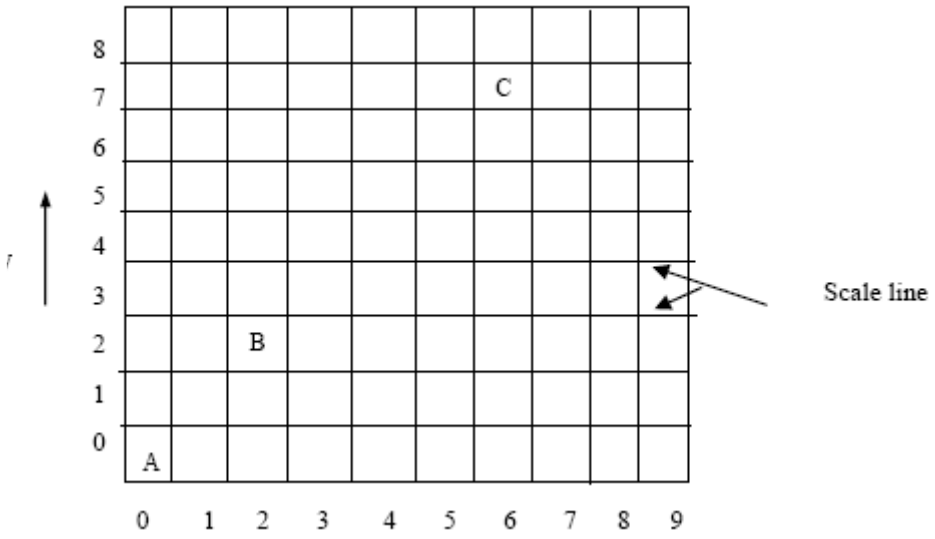


Figure 3.1: Scan-converting point

We normally use right handed cortesian coordinate system. The origin in this system starts at the bottom. However in case of computer system, due to the memory organization, the system turns out ot left handed Cartesian system. Thus there is a difference in the actual representation and the way in which we work with the points.

The basic steps involved in converting Cartesian coordinate system to the system understable points are:

Step 1: Identify the starting address corresponding to the line on which the point is to be displayed.

Step 2: Find the byte address in which the point is to be displayed.

Step 3: Compute the value for the byte that represents the point.

Step 4: Logically OR the calculated value with the present value of the byte.

Step 5: Store the value found in step 4 in the byte found in steps 1 and 2.

Step 6: Stop.

3.3 Scan-converting a Straight Line

A scan conversion of line locates the coordinates of the pixels lie on or near an ideal straight line impaired on 2D raster grid. Before discussing the various methods, let us see what are the characteristics of line. One expects the following features of line:

1. The line should appear straight line.
2. The line should have equal brightness throughout their length.
3. The line must be drawn rapidly.

Even though the rasterization tries to generate a completely straight line, yet in few cases we may not get equal brightness. Basically, the lines which are horizontal or vertical or oriented by 45^0 , have equal brightness. But for the lines with larger length and different orientations, we need to have complex computations in our algorithms. This may reduce the speed of generation of line. Thus we make some sort of compromise while generating the lines, such as:

1. Calculate only the approximate line length.
2. Make use of simple arithmetic computations, preferabley integer arithmetic.
3. Implement result in hardware or firmware.

A straight line may be defined by two endpoints and an equation figure 3.2. In figure 3.1 the two endpoints are described by (x1, y1) and (x2, y2). The equation of the line is used to describe the x, y coordinates of all the points that lie between these two endpoints. Using the equation of a straight line, $y = mx + b$ where $m = \Delta y/\Delta x$ and b = the y intercept, we can find values of y by incrementing x = x1 to x = x2. By scan-converting these calculated x = x2. By scan-converting these calculated x, y values, we represent the line as a sequence on pixels.

While this method of scan-converting a straight line is adequate for many graphics applications, interactive graphics systems require a much faster response than the method described above can provide. Interactive graphics is a graphics system in which the user dynamically controls the presentation of graphics models on a computer display.

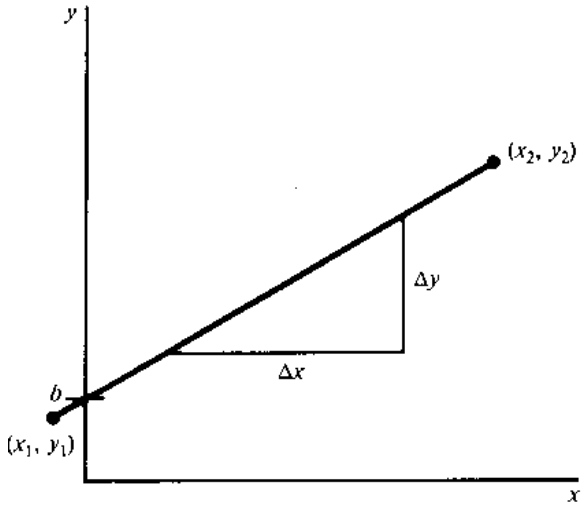


Figure 3.2

3.3.1 Direct Use of Line Equation

A simple approach to scan-converting a line is to first scan-convert P_1 and P_2 to pixel coordinates (x'_1, y'_1) and (x'_2, y'_2), respectively; then set $m = (y'_2, y'_1)(x'_2, x'_1)$ and $b = y'_1 - mx'_1$. If $[m] \leq 1$, then for every integer value of x between and excluding x'_1 and x'_2, calculate the corresponding value of y using the equation and scan-convert (x, y). If $[m] > 1$, then for every integer value of y between and excluding y'_1 and y'_2 calculate the corresponding value of x using the equation and scan-convert (x, y).

While this approach is mathematically sound, it involves floating-point computation (multiplication and addition) in every step that uses the line equation since m and b are generally real numbers. The challenge is to find a way to achieve the same goal as quickly as possible.

3.3.2 DDA Algorithm

This algorithm works on the principle of obtaining the successive pixel values basede on the differential equation goverening the line. Since screen pixels are referred with integer values, or plotted positions, which may only approximate the calculated coordinates – i.e., pixels which are intensified are those which lie very close to the line path if not exactly on the line path which in this case are perfectly horizontal, vertical or 45° lines only. Standard algorithms are available to determine which pixels provide the best approximation to the desired line, one such algorithm is the DDA

(Digital Differential Analyser) algorithm. Before going to the details of the algorithm, let us discuss some general appearances of the line segment, because the respective appearance decides which pixels are to be intensified. It is also obvious that only those pixels that lie very close to the line path are to be intensified because they are the ones which best approximate the line. Apart from the exact situation of the line path, which in this case are perfectly horizontal, vertical or 45° lines (i.e., slope zero, infinite, one) only. We may also face a situation where the slope of the line is > 1 or < 1.Which is the case shown in Figure 3.3.

In Figure 3.3, there are two lines. Line 1 (slope<1) and line 2 (slope>1). Now let us discuss the general mechanism of construction of these two lines with the DDA algorithm. As the slope of the line is a crucial factor in its construction, let us consider the algorithm in two cases depending on the slope of the line whether it is > 1 or < 1.

Case 1: slope (m) of line is < 1 (i.e., line 1): In this case to plot the line we have to move the direction of pixel in x by 1 unit every time and then hunt for the pixel value of the y direction which best suits the line and lighten that pixel in order to plot the line.

So, in Case 1 i.e., 0 < m < 1 where x is to be increased then by 1 unit every time and proper y is approximated.

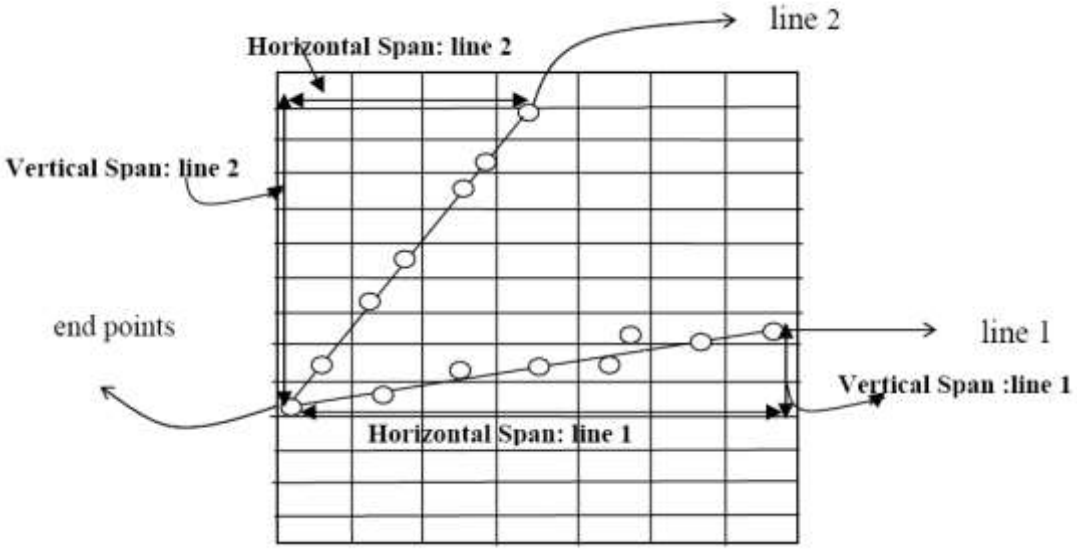


Figure 3.3 DDA Line Generation

Case 2: slope (m) of line is > 1 (i.e., line 2) if m > 1 i.e., case of line 2, then the most appropriate strategy would be to move towards the y direction by 1 unit every time and determine the pixel in x direction which best suits the line and get that pixel lightened to plot the line.

So, in Case 2, i.e., (infinity) > m > 1 where y is to be increased by 1 unit every time and proper x is approximated.

3.3.3 Bresenham's Line Algorithm

Bresenham's line-drawing algorithm uses an iterative scheme. A pixel is plotted at the starting coordinate of the line, and each iteration of the algorithm increments the pixel one unit along the major, or x-axis. The pixel is incremented along the minor, or y-axis, only when a decision variable (based on the slope of the line) changes sign. A key feature of the algorithm is that it requires only integer data and simple arithmetic. This makes the algorithm very efficient and fast.

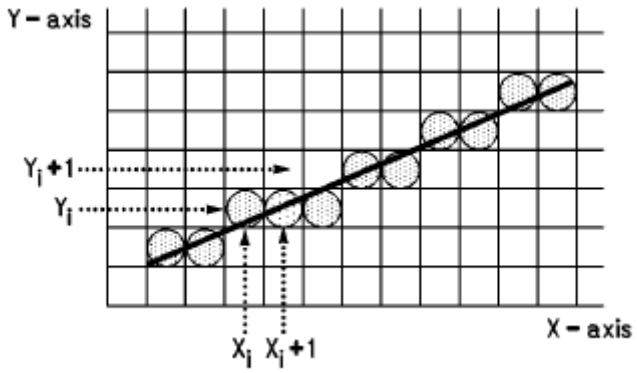


Figure 3.4

The algorithm assumes the line has positive slope less than one, but a simple change of variables can modify the algorithm for any slope value.

Bresenham's Algorithm for 0 < slope < 1

Figure 3.4 shows a line segment superimposed on a raster grid with horizontal axis X and vertical axis Y. Note that xi and yi are the integer abscissa and ordinate respectively of each pixel location on the grid. Given (xi, yi) as the previously plotted pixel location for the line segment, the next pixel to be plotted is either (x_i + 1, y_i) or (xi + 1, yi + 1). Bresenham's algorithm determines which of these two pixel locations is nearer to the actual line by calculating the distance from each pixel to the line, and plotting that pixel with the smaller distance. Using the familiar equation of a straight line, y = mx + b, the y value corresponding to xi + 1 is y=m(xi+1) + b The two distances are then calculated as:

d1 = y- yi

d1= m(xi + 1) + b- yi

d2 = (yi+ 1)- y

d2 = (yi+ 1) - m(xi + 1)- b

and,

d1 - d2 = m(xi+ 1) + b - yi - (yi + 1) + m(xi + 1) + b

d1 - d2 = 2m(xi + 1) - 2yi + 2b - 1

Multiplying this result by the constant dx, defined by the slope of the line m = dy/dx, the equation becomes:

dx(d1-d2)= 2dy(xi)- 2dx(yi) + c

where c is the constant 2dy + 2dxb - dx. Of course, if d2 > d1, then (d1-d2) < 0, or conversely if d1 > d2, then (d1-d2) >0. Therefore, a parameter pi can be defined such that

pi = dx(d1-d2)

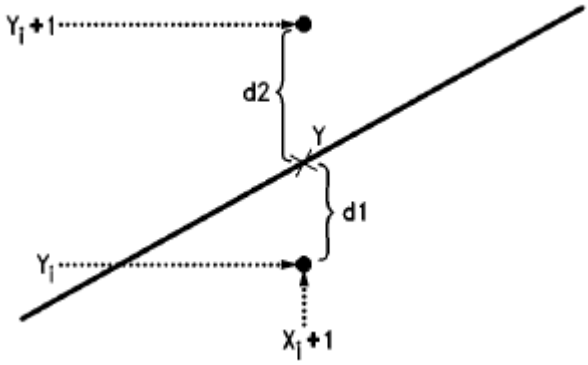


Figure 3.5

pi = 2dy(xi) - 2dx(yi) + c

If pi > 0, then d1 > d2 and y_{i+1} is chosen such that the next plotted pixel is (xi + 1, yi). Otherwise, if pi < 0, then d2 > d1 and (xi + 1, yi + 1) is plotted. (See Figure3.5 .)

Similarly, for the next iteration, p_{i+1} can be calculated and compared with zero to determine the next pixel to plot. If $p_{i+1} < 0$, then the next plotted pixel is at (x_{i+1} + 1, Yi+1); if $p_{i+1} < 0$, then the next point is (x_{i+1} + 1, y_{i+1} + 1). Note that in the equation for p_{i+1}, $x_{i+1} = x_{i+1}$.

p_{i+1} = 2dy(xi + 1) - 2dx(yi + 1) + c.

Subtracting pi from p_{i+1}, we get the recursive equation:

p_{i+1} = pi + 2dy - 2dx(y_{i+1} - yi)

Note that the constant c has conveniently dropped out of the formula. And, if pi < 0 then y_{i+1}= yi in the above equation, so that:

$p_{i+1} = pi + 2dy$

or, if pi > 0 then yi + 1 = yi + 1, and

pi + 1 = pi + 2(dy-dx)

To further simplify the iterative algorithm, constants c1 and c2 can be initialized at the beginning of the program such that c1 = 2dy and c2 = 2(dy-dx). Thus, the actual meat of the algorithm is a loop of length dx, containing only a few integer additions and two compares (Figure 3.5) .

3.4 Scan-converting a Circle

Circle is one of the basic graphic component, so in order to understand its generation, let us go through its properties first. A circle is a symmetrical figure. Any circle-generating algorithm can take advantage of the circle's symmetry to plot of eight points for each value that the algorithm calculates. Eight-way symmetry is used by reflecting each calculated point around each 45˚ axis. For example, if point 1 in Fig. 3.6 were calculated with a circle algorithm, seven more points could be found by reflection. The reflection is accomplished by reversing the x, y coordinates as in point 2, reversing the x,y coordinates and reflecting about the y axis as in point 3, reflecting about the y axis in point 4, switching the signs of x and y as in point 5, reversing the x, y coordinates, reflecting about the y axis and reflecting about the x axis.

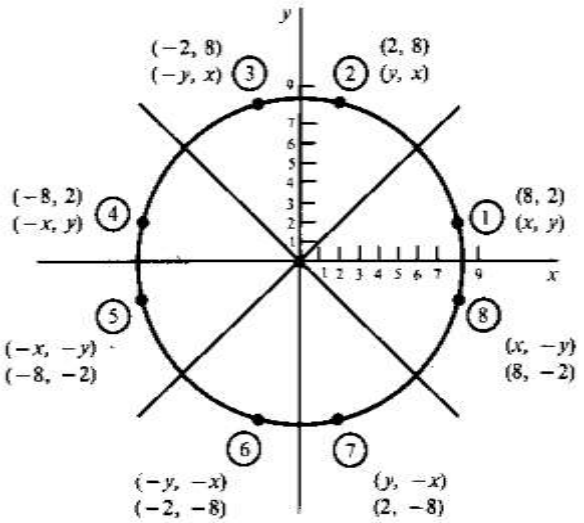


Figure 3.6: Eight–way symmetry of a circle.

As in point 6, reversing the x, y coordinates and reflecting about the y-axis as in point 7, and reflecting about the x-axis as in point 8.

To summarize:

$P_1 = (x, y)$ $P_5 = (- y,-x)$

$P_2 = (y, x)$ $P_6 = (-y, - x)$

$P_3 = (-y, x)$ $P_7 = (y, -x)$

$P_4 = (-x, y)$ $P_8 = (x, -y)$

3.4.1 Defining a Circle

There are two standard methods of mathematically defining a circle centered at the origin. The first method defines a circle with the second-order polynomial equation (see Fig. 3.7).

$y_2 = r^2 – x^2$

Where *x* = the x coordinate

y = the y coordinate

r = the circle radius

With this method, each x coordinate in the sector, from 90 to 45˚, is found by stepping x from 0 to $r/\sqrt{2}$, and each y coordinate is found by evaluating $\sqrt{r^2 + x^2}$ for each step of x. This is a very inefficient method, however, because for each point both x and r must be squared and subtracted from each other; then the square root of the result must be found.

The second method of defining a circle makes use of trigonometric functions (see Fig. 3.8):

$x = r \cos \theta$ $y = r \sin \theta$

where θ = current angle

r = circle radius

x = x coordinate

y = y coordinate

By this method, θ is stepped from θ to $\pi/4$, and each value of x and y is calculated. However, computation of the values of sin θ and cos θ is even more time-consuming than the calculations required by the first method.

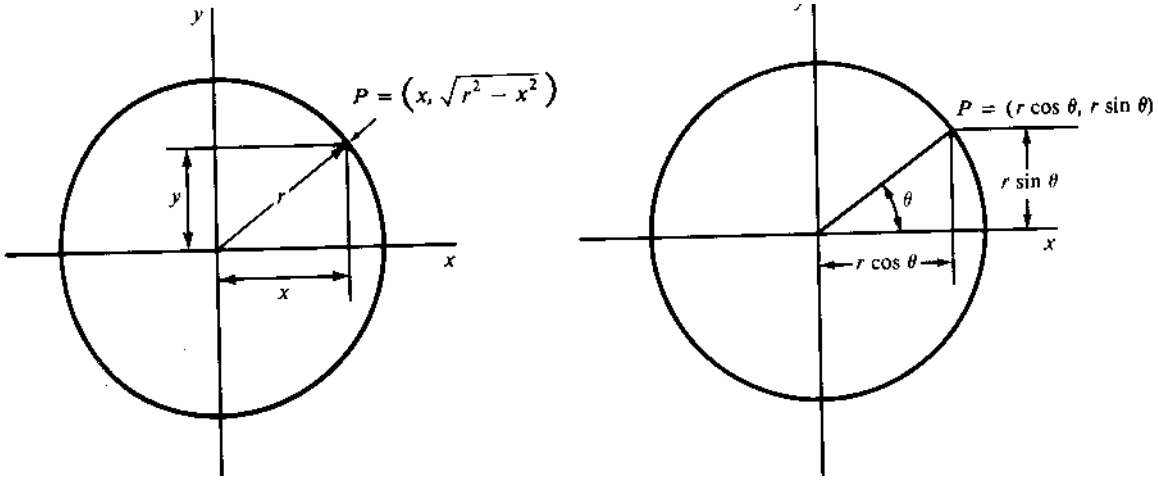


Figure 3.7 & 3.8: Circle defined with a second-egree Polynomial equation and circle defined with trignometric functions respectively.

3.4.2 Bresenham's Circle Algorithm

If a circle is to be plotted efficiently, the use of trigonometric and power functions must be avoided. And as with the generation of a straight line, it is also desirable to perform the calculations necessary to find the scan-converted points with only integer addition, subtraction, and multiplication by powers of 2. Bresenham's circle algorithm allows these goals to be met.

Scan-converting a circle Bresenham's algorithm works as follows. If the eight-way symmetry of a circle is used to generate a circle, points will only have to be generated through a 45°, moves will be made only in the +x and –y directions (see Fig. 3.9).

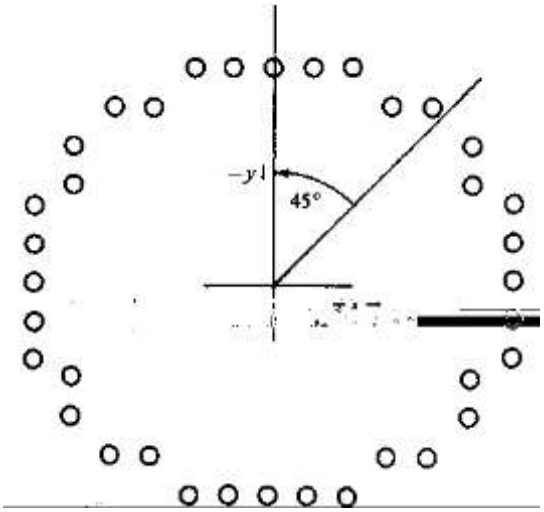


Figure 3.9: Circle scan-converted with Bresenham's algorithm

The best approximation of the true circle will be described by those pixels in the raster that fall the least distance from the true circle. Examine Figs. 3.10 (a) and 3.10 (b). Notice that if points are generated from 90 and 45°, each new point closest to the true circle can be found by taking either of two actions:(1) move in the *x* direction one unit or (2) move in the *x* direction one unit. Therefore, a method of selecting between these two choices is all that is necessary to find the points closets to the true circle.

The process is as follows. Assume that the last scan-converted pixel is P1 [see Fig. 3-10(b)]. Let the distance from the origin to the true circle squared minus the distance to point P3 squared = D(Si). Then let the distance from the origin to the true circle squared minus the distance to point P2 squared = D (Ti). As the only possible valid moves are to move either one step in the x direction or one step in the x direction and one step in the negative y direction, the following expressions can be developed:

$$D(S) = (x_{i-1})^2 + y^2_{i-1} - r^2 \qquad D(T_i) = (x_{i-1})^2 + (y_{i-1} - 1)^2 - r^2$$

Since D (Si) will always be positive and D (Ti) will always be negative, a decision variable d may be defined as follows:

$$d_i = D(S) + D(T_i)$$

Therefore

$$d_i = (x_{i-1} + 1)^2 + y^2_{i-1} - r^2 + (x_{i-1} + 1)^2 + (y_{i-1} - 1)^2 - r^2$$

From this equation we can derive

$$d_1 = 3 - 2r$$

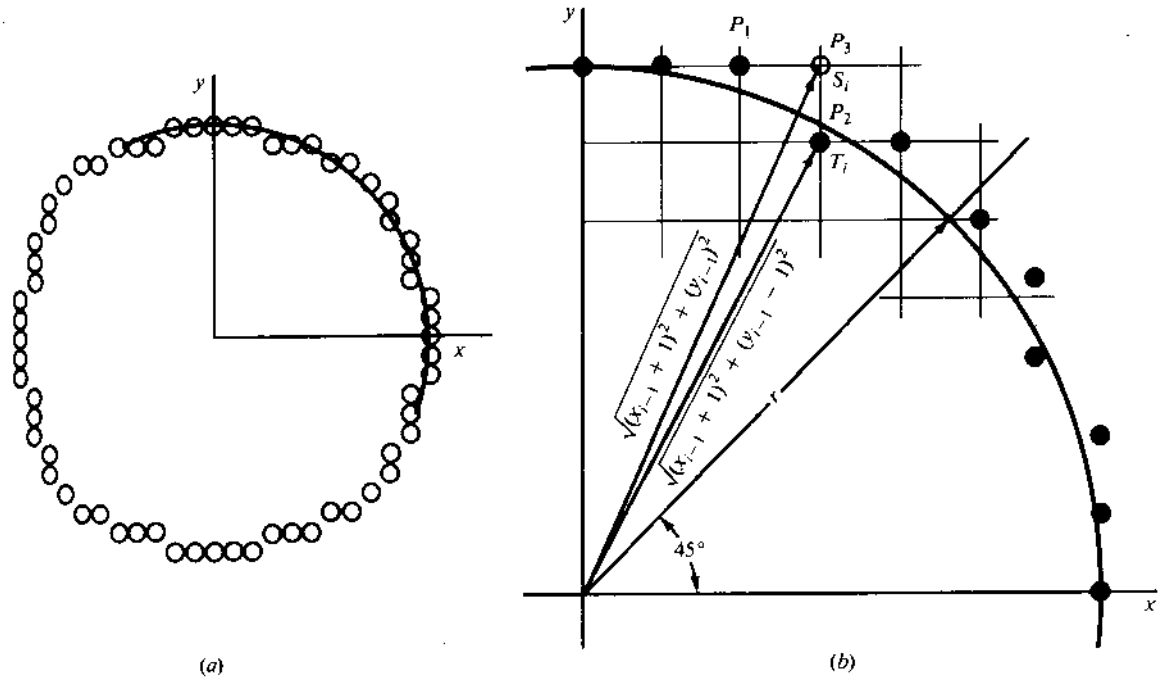


Figure 3.10

Thereafter, if $d_i > 0$, then only *x* is incremented:

$$x_{i+1} = x_i + 1 \qquad d_{i+1} = d_i + 4x_i + 6$$

and if $d \le 0$, then *x* and y are incremented:

$$x_{i+1} = x_i + 1 \qquad y_{i+1} = y_i - 1 \qquad d_{i+1} = d_i + 4(x_i - y_i) + 10$$

3.4.3 Mid point Circle Algorithm

We present another incremental circle algorithm that is very similar to Bresenham's approach. It is based on the following function for testing, the spatial relationship between an arbitrary (x, y) and a circle of radius r centered at the origin:

$$f(x,y) = x^2 + y^2 - r^2$$

<0 (X,Y) inside the circle

=0 (X,Y) on the circle

>0 (X,Y) outside the circle

Now consider the coordinates of the point halfway between pixel T and pixel S in Fig. 3-8: $(x_i + 1, y_i - \frac{1}{2})$. This is called the midpoint and we use it to define decision parameter:

$$p_i = f(x_i + 1, y_i - \frac{1}{2}) = (x_i + 1)^2 + (y_i - \frac{1}{2})^2 - r^2$$

If p_i is negative, the midpoint is inside the circle, and we choose pixel T. On the other hand, if p_i is positive (or equal to zero), the midpoint is outside the circle (or on the circle), and we choose pixel S. Similarly, the decision parameter for the next step is

$$p_{i+1} = (x_{i+1} + 1)^2 + (y_{i+1} - \frac{1}{2})^2 - r^2$$

Since $x_{i+1} = x_i + 1$, we have

$$p_{i+1} - p_i = [(x_i + 1)1]^2 - (x_i + 1)^2 + \left(y_{i+1} - \frac{1}{2}\right)^2 - \left(y_i - \frac{1}{2}\right)^2$$

Hence

$$p_{i+1} = p + 2(x_i + 1) + 1 + (y_i^2 + y_i^2) - (y_{i+1} - y_i)$$

If pixel T is chosen (meaning $p_i < 0$), we have $y_{i+1} = y_i$. On the other hand, if pixel S is chosen (meaning $p_i \geq 0$), we have $y_{i+1} = y_i - 1$. Thus

$$p_{i+1} = \begin{cases} p_i + 2(x_i + 1) + 1 & \text{If } p_i < 0 \\ p_i + 2(x_i + 1) + 1 - 2(y_i - 1) & \text{If } p_i \geq 0 \end{cases}$$

We can continue to simplify this in terms of (x_i, y_i) and get

$$p_{i+1} = \begin{cases} p_i + 2x_i + 3 & \text{If } p_i < 0 \quad \text{If } p_i < 0 \\ p^i + 2(x_i - y_{i+1}) + 1 & \text{If } p_i \geq 0 \end{cases}$$

Finally, we compute the initial value for the decision parameter using the original definition of p_i and (0, r):

$$p_i = (0 + 1)^2 + \left(r - \frac{1}{2}\right)^2 - r^2 = \frac{5}{4} - r$$

One can see that this is not really integer computation. However, when r is a integer we can simply set p1 = 1 – r. The error of being $\frac{1}{4}$ less than the precise value does not prevent p1 from getting the appropriate sign. It does not affect the rest of the scan-conversion process either, because the decision variable is only updated with integer increments in subsequent steps.

The following is a description of this midpoint circle algorithm that generates the pixel coordinates in the 90^o to 45^o octant:

```
int x = 0, y = r, p = 1 – r;
while (x < = y) {
setPixel (x, y);
If (p < 0)
p = p + 2x + 3;
else {
p = p + 2 (x – y) + 5;
y –;
}
x++;
}
```

3.5 Scan-converting an Ellipse

The ellipse, like the circle, shows symmetry. In the case of an ellipse, however, symmetry is four-rather than eight-way. There are two methods of mathematically defining an ellipse.

3.5.1 Polynomial Method of Defining an Ellipse

The polynomial method of defining an ellipse (Fig. 3.11 is given by the expression

$$\frac{(x-h)^2}{a^2}+\frac{(y-k)^2}{b^2}=1$$

where (h, k) = ellipse center

a = length of major axis

b = length of minor axis

When the polynomial method is used to define an ellipse, the value of x is incremented from from h to a. For each step of x, each value of y is found by evaluating the expression

$$y = b\sqrt{1 - \frac{x - h^2}{a^2}} + k$$

This method is very inefficient, however, because the squares of a and (x – h) must be found; then floating-point division of $(x-h)^2$ by a^2] and floating point multiplication of the square root of $[1-(x-h)^2/\ a^2]$ by *b* must be performed

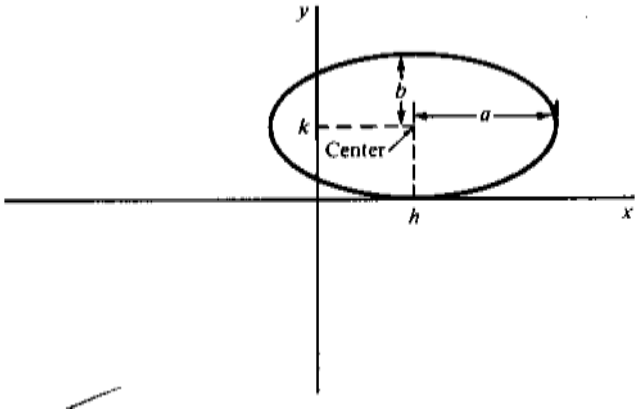


Figure 3.11: Polynomial description of an ellipse

Routines have been found that will scan-convert general polynomial equations, including the ellipse. However, these routines are logic intensive and thus are very slow methods for scan-converting ellipses.

Steps for generating ellipse using polynomial method are:

1. Set the initial variables: a = length of majo0r axis; b = length of minor axis; (h, k) = coordinates of ellipse center; x = 0; i = step; x_{end} = a.
2. Test to determine whether the entire ellipse has been scan-converted. If x> x_{end}, stop.

3. Compute the value of the y coordinate:

$$y = b\sqrt{1 - \frac{x^2}{a^2}}$$

4. Plot the four points, found by symmetry, at the current (x, y) coordinates:

   Plot (x + h, y + k) Plot (-x + h, -y + k)

   Plot (-y - h, x + k) Plot (y + h, -x + k)

5. Increment x; x = x + i.

6. Go to step 2.

3.5.2 Trigonometric Method of Defining an Ellipse

A second method of defining an ellipse makes use of trigonometric relationships (see Fig. 3.12). The following equations define an ellipse trigonometrically:

$x = a * \cos(0) + h$ and $y = b * \sin(0) + k$

where (x, y) = current coordinate

a = length of major axis

b = length of minor axis

θ = current angle

(*h*, *c*) = ellipse center

For the generation of an ellipse using the trigonometric method, the value of θ is varied from 0 to $\pi/2$ radians (rad). The remaining points are found by symmetry. While this method is also inefficient and thus generally too slow for interactive applications, a lookup table containing the values for $\sin(\theta)$ and $\cos(\theta)$ with θ ranging from 0 to $\pi/2$ rad can be used. This method would have been considered unacceptable at one time because of the relatively high cost of the computer memory used to store the values .

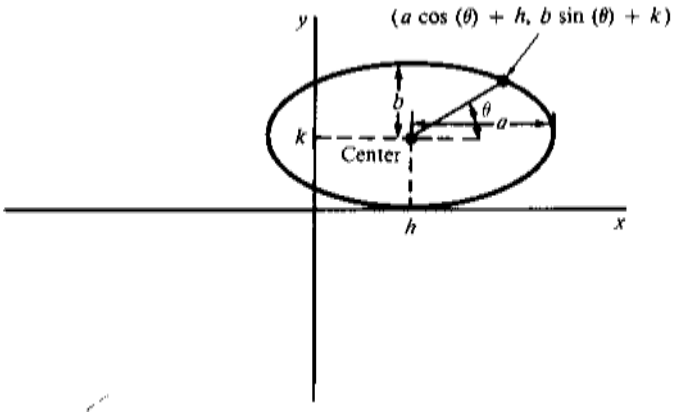


Figure 3.12: Trigonometric description of an ellipse

θ. However, because the cost of computer memory has plummeted in recent years, this method is now quite acceptable.

3.5.3 Ellipse Axis Rotation

Since the ellipse shows fourway symmetry, it can easily be rotated $90°$. The new equation is found by trading a and b, the values which describe the major and minor axes. When the polynomial method is used, the equations used to describe the ellipse become

$$\frac{(x-h)^2}{b^2}+\frac{(y-k)^2}{a^2}=1$$

where (h, k) = ellipse center

a = length of major axis

b = length of minor axis

When the trigonometric is used, the equations used to describe the ellipse become

$x = b * \cos(\theta) + h$ and $y = a * \sin(\theta) + k$

Where (x, y) = current coordinates

a = length of major axis

b = length of minor axis

θ = current angle

(h, k) = ellipse center

Assume that you would like to rotate the ellipse through an angle other than 90 degrees. It can be seen from Fig. 3.12 that rotation of the ellipse may be accomplished by rotating the *x*, *y* coordinates of each scan-converted point which become

$$x = a\cos(0) - b\sin(0+\propto) + h \quad y = b\sin(0) + a\cos(0+\propto) + k$$

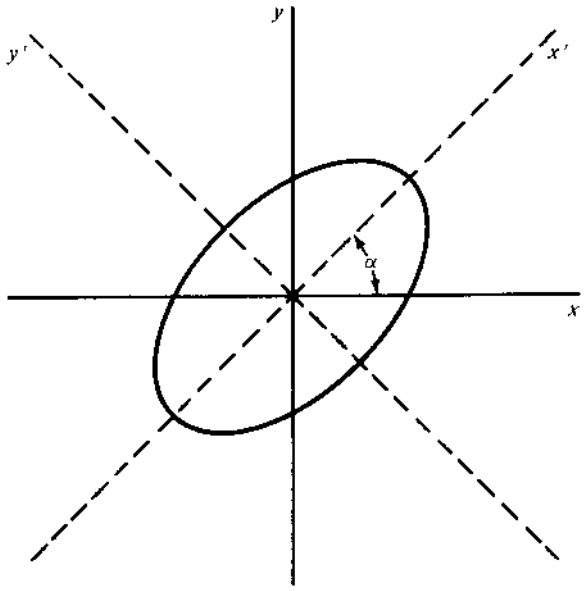


Figure 3.12 : Rotation of an ellipse

3.6 Summary

- The process of representing continuous graphic objects as a collection of discrete pixels is called **scan conversion**.
- Scan-converting a point involves illuminating the pixel that contains the point
- Interactive graphics is a graphics system in which the user dynamically controls the presentation of graphics models on a computer display.
- Bresenham's line algorithm is an efficient method for scan-converting straight lines in and uses only integer addition, subtraction, and multiplication by
- The polynomial or the trigonometric method may generate an arc

3.7 Key Words

Scan conversition, scan conversion of circle, ellipse, line and point

3.8 Self Assessment Questions (SAQ)

1. What is raster graphics? differentiate b/w raster and vector graphics?

2. Explain how Bresenham's algorithm takes advantage of the connectivity of pixels in drawing straight lines on a raster output device.
3. Explain midpoint line algorithm? Write alogorithm in your own words
4. What steps are required to plot a line whose slope is between 45 and 90° using Bresenham's method?
5. What steps are required to plot a dashed line using Bresenham's method?
6. Show graphically that an ellipse has four-way symmetry by plotting four points on the ellipse: $x = a * \cos(0) + h$ $y = b * \sin(0) + k$

   where a = 2 b = 1 h = 0 k = 0 $\theta = \pi/4, 3\pi/4, 5\pi/4, 7\pi/4$
7. How must Prob. 3.9 be modified if an ellipse is to be rotated (a) $\pi/4$, (b) $\pi/9$, and (c) $\pi/2$ radians?
8. What steps are required to scan-convert a sector using the trigonometric method?

3.9 References/Suggested Readings

1. Computer Graphics, Principles and Practice, Second Edition, by James D. Foley, Andries van Dam, Steven K. Feiner, John F. Hughes, Addison-Wesley
2. Computer Graphics , Second Edition , by Pradeep K. Bhatia , I.K .International Publisher.
3. High Resolution Computer Graphics using Pascal/C, by Ian O. Angell and Gareth Griffith, John Wiley & Sons
4. Computer Graphics (C Version), Donald Hearn and M. Pauline Baker, Prentice Hall,
5. Advanced Animation and Rendering Techniques, Theory and Practice, Alan Watt and Mark Watt , ACM Press/Addison-Wesley

SUBJECT: COMPUTER GRAPHICS

COURSE CODE: MCA-44(iv)	AUTHOR: Abhishek Taneja

LESSON NO. 4

Two Dimensional Transformation

Unit Structure

4.0 Objectives

At the end of this chapter the reader will be able to:

- Describe two dimensional transformations
- Describe and distinguish between two dimensional geometric and coordinate transformations
- Describe composite transformations
- Describe shear transformations

4.1 Introduction

Transformations are fundamental part of computer graphics. In order to manipulate object in two dimensional space, we must apply various transformation functions to object. This allows us to change the position, size, and orientation of the objects. Transformations are used to position objects, to shape objects, to change viewing positions, and even to change how something is viewed.

There are two complementary points of view for describing object movement. The first is that the object itself is moved relative to a stationary coordinate system or background. The mathematical statement of this viewpoint is described by geometric transformations applied to each point of the object. The second point of view holds that the object is held stationary while the coordinate system is moved relative to the object. This effect is attained through the application of coordinate transformations. An example involves the motion of an automobile against a scenic background. We can also keep the automobile fixed while moving the backdrop fixed (a geometric transformation). We can also keep the automobile fixed while moving the backdrop scenery (a coordinate transformation). In some situations, both methods are employed.

Coordinate transformations play an important role in the instancing of an object – the placement of objects, each of which is defined in its own coordinate system, into an overall picture or design defined with respect to a master coordinate system.

4.2 Geometric Transformations

An object in the plane is represented as a set of points (vertices). Let us impose a coordinate system on a plane. An object Obj in the plane can be considered as a set of points. Every object point P has coordinates (x, y), and so the object is the sum total of all its coordinate points. If the object is moved to a new position, it can be

regarded as a new object Obj', all of whose coordinate point P' can be obtained from the original points P by the application of a geometric transformation.

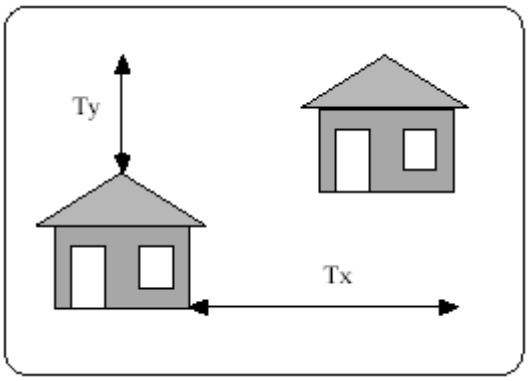


Figure 4.1

Points in 2-dimensional space will be represented as column vectors:

We are interested in three types of transformation:

- Translation
- Scaling
- Rotation
- Mirror Reflection

4.2.1 Translation

In translation, an object is displaced a given and direction from its original position. If the displacement is given by the vector $v = t_x I + t_y J$, the new object point $P'(x', y')$ can be found by applying the transformation T_v to P(x, y) (see Fig. 4.1).

$P' = T_v(P)$ where $x' = x + t_x$ and $y' = y + t_y$.

4.2.2 Rotation about the origin

In rotation, the object is rotated $\theta°$ about the origin. The convention is that the direction of rotation is counterclockwise if θ is a positive angle and clockwise if θ is a negative angle (see Fig. 4.2). The transformation of rotation R_θ is

$P' = R_\theta(P)$

where $x' = x\cos(\theta) - y\sin(\theta)$ and $y' = x\sin(\theta) + y\cos(\theta)$

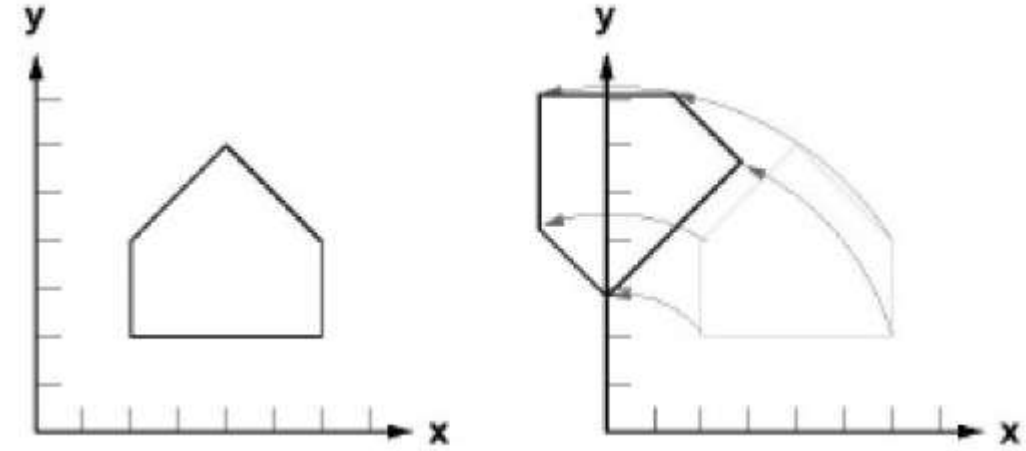


Figure 4.2

4.2.3 Scaling with Respect to the origin

Scaling is the process of expanding or compressing the dimension of an object. Positive scaling constants S_x and S_y, are used to describe changes in length with respect to the x direction and *y* direction, respectively. A scaling constant greater than one indicates an expansion of length, and less than one, compression of length. The scaling transformation $S_{s_x s_y}$ is given by $P' = S_{s_x s_y}(P)$ where $x' = s_x.x$ and $y' = s_x.y$. Notice that after a scaling transformation is performed, the new object is located at a different position relative to the origin. In fact, in a scaling transformation the only point that remains fixed is the origin (Figure 4.3).

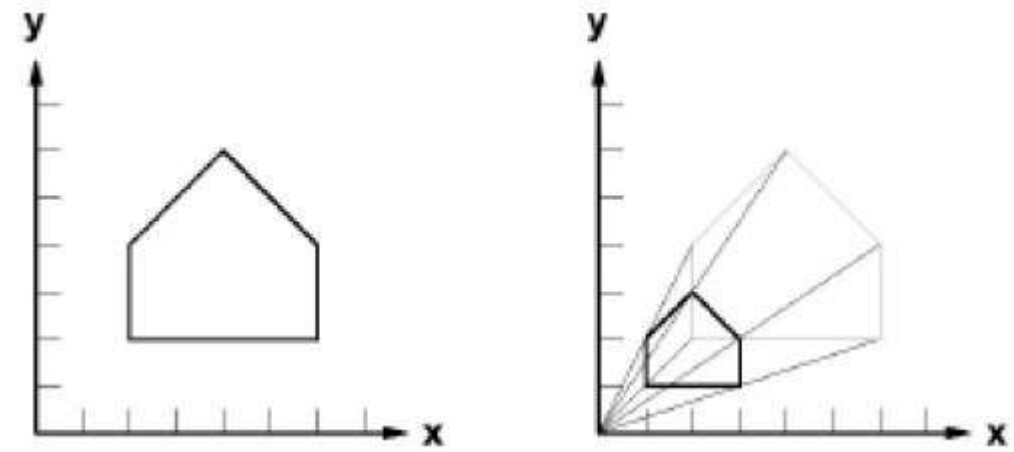


Figure 4.3

If both scaling constants have the same value *s*, the scaling transformation is said to be homogeneous. Furthermore, if s > 1, it is a magnification and for s < 1, a reduction

4.2.4 Mirror Reflection about an Axis

If either the x and y axis is treated as a mirror, the object has a mirror image or reflection. Since the reflection P' of an object point P is located the same distance from the mirror as P (Fig. 4.4), the mirror reflection transformation M_x about the x-axis is given by

$$P' = M_x(P)$$

where $x' = x$ and $y' = -y$.

Similarly, the mirror reflection about the y-axis is

$$P' = M_y(P)$$

where $x' = -x$ and $y' = y$.

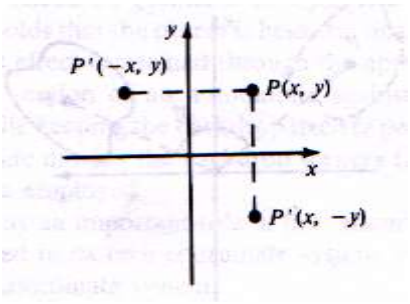


Figure 4.4

4.2.5 Inverse Geometric Transformation

Each geometric transformation has an inverse, which is described by the opposite operation performed by the transformation.

Translation: $T_v^{-1} = T_{-v}$ or translation in the opposite direction

Rotation: $R_\theta^{-1} = R_{-\theta}$ or rotation in the opposite direction

Scaling: $S_{s_x,s_y}^{-1} = S_{1/s_x,1/s_y}$

Mirror reflection: $M_x^{-1} = M_x$ and $M_y^{-1} = M_y$

4.3 Coordinate Transformations

Suppose that we have two coordinate systems in the plane. The first system is located at origin O and has coordinate axes xy figure 4.6. The second coordinate system is located at origin O' and has coordinate axes $x'y'$ Now each point in the plane has two coordinate descriptions: (x, y) or (x', y'), depending on which coordinate system is used. If we think of the second system $x'y'$ as arising from a transformation applied to the first system xy, we say that a coordinate transformation has been applied. We can describe this transformation by determining how the (x', y') coordinates of a point P are related to the (x, y) coordinates of the same point.

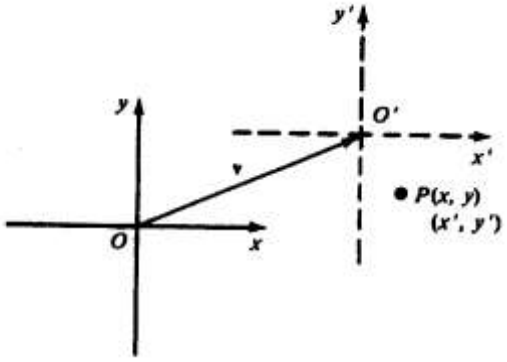


Figure 4.5

4.3.1 Translation

If the xy coordinate system is displaced to a new position, where the direction and distance of the displacement is given by the vector $\mathrm{v} = t_x \mathrm{I} + t_y \mathrm{J}$, the coordinates of a point in both systems are related by the translation transformation $\overline{\mathrm{T}}_{\mathrm{v}}$:

$$(x', y') = \overline{\mathrm{T}}_{\mathrm{v}}(x, y)$$

where $x' = x - t_x$ and $y' = y - t_y$

4.3.2 Rotation about the Origin

The *xy* system is rotated by $\theta°$ about the origin figure 4.6. Then the coordinates of a point in both systems are related by the rotation transformation $\overline{\mathrm{R}}_{\theta}$:

$(x', y') = \bar{R}_{\theta}(x, y)$

where $x' = x\cos(\theta) + y'\sin(\theta)$ and $y' = -x\sin(\theta) + y\cos(\theta)$.

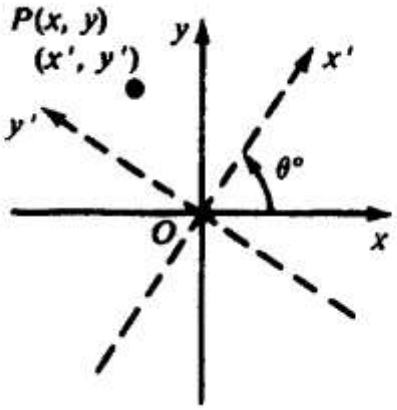


Figure 4.6

4.3.3 Scaling with Respect to the Origin

Suppose that a new coordinate system is formed by leaving the origin and coordinate axes unchanged, but introducing different units of measurement along the x and y axes. If the new units are obtained from the old units by a scaling of s_y units along the y-axis, the coordinates in the new system are related to coordinates in the old system through the scaling transformation $\bar{S}_{s_x, s_y}$:

$(x', y') = \bar{S}_{s_x, s_y}(x, y)$

where $x' = 1/s_x.x$ and $y' = 1/s_y.y$. Figure 4.7 shows coordinate scaling transformation using scaling factors $s_x = 2$ and $s_y = \frac{1}{2}$.

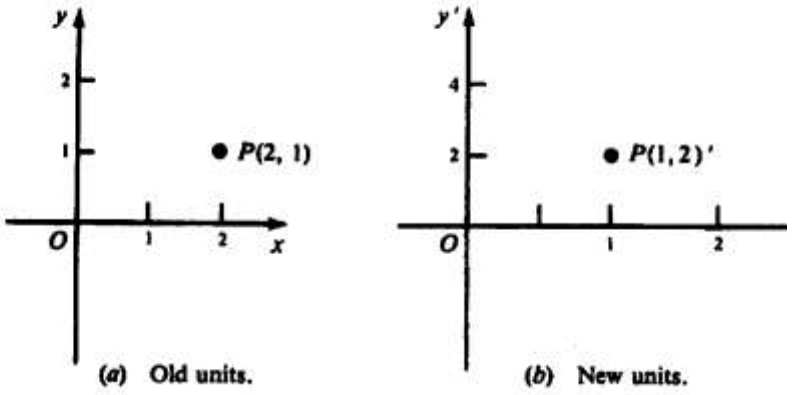


Figure 4.7

4.3.4 Mirror Reflection about an Axis

If the new coordinate system is obtained by reflecting the old system about either *x* or *y* axis, the relationship between coordinates is given by the coordinate transformations $\overline{M}_x$ and $\overline{M}_y$: For reflection about the x axis (figure 4.8 (a))

$$(x', y') = \overline{M}_x(x, y)$$

where $x' = x$ and $y' = -y$. For reflection about the *y* axis [figure 4.8(b)]

$$(x', y') = \overline{M}_y(x, y)$$

where $x' = -x$ and $y' = y$.

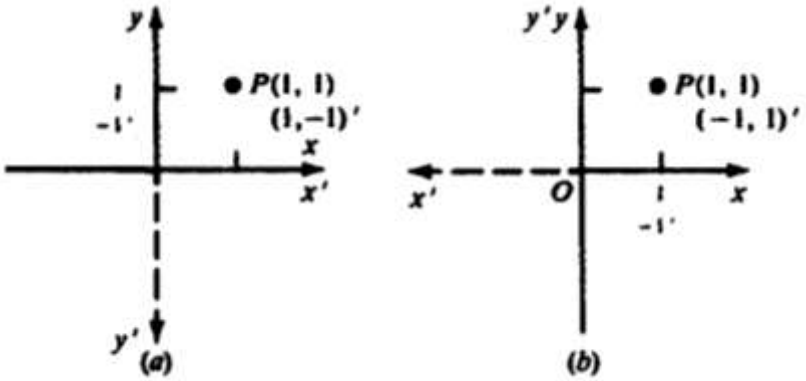


Figure 4.8

Notice that the reflected coordinate system is left-handed; thus reflection changes the orientation of the coordinate system.

4.3.5 Inverse Coordinate Transformation

Each coordinate transformation has an inverse which can be found by applying the opposite transformation:

Translation : $\overline{T}_v^{-1} = \overline{T}_{-v}$ translation in the opposite direction

Rotation: $\overline{R}_\theta^{-1} = \overline{R}_{-\theta}$ rotation in the opposite direction

Scaling : $\overline{S}_{s_x,s_y}^{-1} = \overline{S}_{1/s_x,1/s_y}$

Mirror reflection: $M_x = \overline{M}_x$ and $M_y = \overline{M}_y$

4.4 Composite Transformations

More complex geometric and coordinate transformations can be built from the basic transformations described above by using the process of composition of functions. For example, such operations as rotation about a point other than the origin or reflection about lines other than the axes can be constructed from the basic transformations.

Matrix Description of the Basic Transformations

The transformations of rotation, scaling, and reflection can be represented as matrix functions:

Geometric Transformations	**Coordinate Transformations**
$R_\theta = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$	$\overline{R}_\theta = \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix}$
$S_{s_x,s_y} = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix}$	$\overline{S}_{s_x,s_y} = \begin{pmatrix} \frac{1}{s_x} & 0 \\ 0 & \frac{1}{s_y} \end{pmatrix}$
$M_x = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	$\overline{M}_x = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$
$M_y = \begin{pmatrix} -1 & 0 \\ 1 & 1 \end{pmatrix}$	$\overline{M}_y = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$

The translation transformation cannot be expressed as a 2 x 2 matrix function. However, a certain artifice allows us to introduce a 3 x 3 matrix function, which performs the translation transformation.

We represent the coordinate pair (x, y) of a point P by the triple (x, y, 1). This is simply the homogeneous representation of P. Then translation in the direction $\mathrm{v} = t_x \mathrm{I} + t_y \mathrm{J}$ can be expressed by the matrix function.

$$T_v = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix}$$

Then

$$\begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x + t_x \\ y + t_y \\ 1 \end{pmatrix}$$

From this we extract the coordinate pair $(x + t_x, y + t_y)$.

Concatenation of Matrices

The advantage of introducing a matrix form for translation is that we can now build complex transformation by multiplying the basic matrix transformations. This process is sometimes called concatenation of matrices. Here, we are using the fact that the composition of matrix functions is equivalent to matrix multiplication. We must be able to represent the basic transformations as 3 x 3 homogeneous coordinate matrices so as to be compatible (from the point of view of matrix multiplication) with the matrix of translation. This is accomplished by augmenting the 2 x 2 matrices with a third column $\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$ and a third row (0 0 1). That is

$$\begin{pmatrix} a & b & 0 \\ b & d & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Here we share some solved problems to better understand the transformations that we discussed above.

Problem 1 Derive the transformation that rotates an object point θ° about the origin. Write the matrix representation for this rotation.

Answer Refer to Fig. 4.9. Definition of the trigonometric functions sin and cos yields

$$x' = r\cos(\theta+\phi) \qquad y' = r\sin(\theta+\phi)$$

and

$$x = r\cos\phi \qquad y = r\sin\phi$$

Using trigonometric identities, we obtain

$$r\cos(\theta+\phi) = r(\cos\theta\cos\phi - \sin\theta\sin\phi) = x\cos\theta - y\sin\theta$$

and

$$r\sin(\theta+\phi) = r(\sin\theta\cos\phi - \cos\theta\sin\phi) = x\sin\theta - y\cos\theta$$

or

$$x' = x\cos\theta - y\sin\theta \qquad y' = x\sin\theta + y\cos\theta$$

Writing $P' = \begin{pmatrix} x' \\ y' \end{pmatrix}$, $P = \begin{pmatrix} x \\ y \end{pmatrix}$, and

$$R_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

we can now write P' = R□. P.

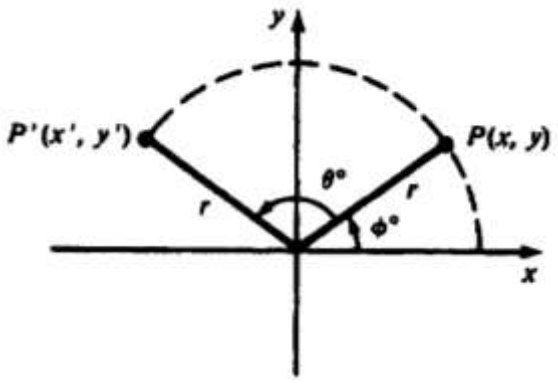


Figure 4.9

Problem 2 Write the general form of the matrix for rotation about a point P (h, k).

Solution Following we write $R_{\theta,P} = T_v . R_\theta . T_{-v}$, where v = hI + kJ. Using the 3×3 homogeneous coordinate form for the rotation and translation matrices, we have

$$R_{\theta,P} = \begin{pmatrix} 1 & 0 & h \\ 0 & 1 & k \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(\theta) & -\sin(\theta) & h \\ \sin(\theta) & \cos(\theta) & k \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -h \\ 0 & 1 & -k \\ 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} \cos(\theta) & -\sin(\theta) & [-h\cos(\theta) + k\sin(\theta) + h] \\ \cos(\theta) & -\sin(\theta) & [-h\cos(\theta) + k\sin(\theta) + h] \\ 0 & 0 & 0 \end{pmatrix}$$

Problem 3

(a) Find the matrix that represents rotation of an object by 30° about the origin.

(b) What are the new coordinates of the point P (2, 4) after the rotation?

Answer

(a) From problem 1

$$R_{30^\circ} = \begin{pmatrix} \cos 30^\circ & -\sin 30^\circ \\ \sin 30^\circ & \cos 30^\circ \end{pmatrix} = \begin{pmatrix} \frac{\sqrt{3}}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{\sqrt{3}}{2} \end{pmatrix}$$

(b) So the new coordinates can be found by multiplying:

$$\begin{pmatrix} \frac{\sqrt{3}}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{\sqrt{3}}{2} \end{pmatrix} \begin{pmatrix} 2 \\ -4 \end{pmatrix} = \begin{pmatrix} \sqrt{3} + 2 \\ 1 - 2\sqrt{3} \end{pmatrix}$$

Problem 4 Perform a 45° rotation of triangle A (0, 0), B (1, 1), C (5, 2)

(a) about the origin and

(b) about P (-1, -1).

Answer

We represent the triangle by a matrix formed from the homogeneous coordinates of the varitices:

$$\begin{pmatrix} A & B & C \\ 0 & 1 & 5 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \end{pmatrix}$$

(a) The matrix of rotation is

$$R_{45^o} = \begin{pmatrix} \cos 45^o & -\sin 45^o & 0 \\ \sin 45^o & \cos 45^o & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

So the coordinates $A'B'C'$ of the rotated triangle ABC can be found as

$$[A'B'C'] = R_{45^o}[ABC] = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 5 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} A' & B' & C' \\ 0 & 0 & \frac{3\sqrt{2}}{2} \\ 0 & \sqrt{2} & \frac{7\sqrt{2}}{2} \\ 1 & 1 & 1 \end{pmatrix}$$

Thus $A' = (0,0), B' = \left(0, \sqrt{2}\right), \text{and} C' = \left(\frac{3}{2}\sqrt{2}, \frac{7}{2}\sqrt{2}\right).$

(b) The rotation matrix is given by $R_{45^o,P} = T_v . R_{45^o} . T_{-v,}$ where $v = -I - J$. So

$$R_{45^o,P} = \begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & -1 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \left(\sqrt{2}-1\right) \\ 0 & 0 & 1 \end{pmatrix}$$

Now

$$[A'B'C'] = R_{45^o}, p[ABC] = \begin{pmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & -1 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \left(\sqrt{2}-1\right) \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 5 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} -1 & -1 & \left(\frac{3}{2}\sqrt{2}-1\right) \\ \left(\sqrt{2}-1\right) & \left(2\sqrt{2}-1\right) & \left(\frac{9}{2}\sqrt{2}-1\right) \\ 1 & 1 & 1 \end{pmatrix}$$

So $A' = \left(-1, \sqrt{2}-1\right), B' = \left(-1, 2\sqrt{2}-1\right), \text{and} C' = \left(\frac{3}{2}\sqrt{2}-1, \frac{9}{2}\sqrt{2}-1\right).$

Problem 5 Write the general form of a scaling matrix with respect to a fixed point $P(h,k).$

Answer

Following the same general procedure as in Example 2 and 3, we write the required transformation with

$v = hI + kJas$

$$S_{a,b,}P = T_v.S_{a,b}T_{-v}$$

$$= \begin{pmatrix} 1 & 0 & h \\ 0 & 1 & k \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & -h \\ 0 & 1 & -k \\ 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} a & 0 & -ab+h \\ 0 & b & -bk+k \\ 0 & 0 & 1 \end{pmatrix}$$

Problem 6 Find the transformation that scales (with respect to the origin) by (a)aunit in the X direction, (b)b units in the Y direction, and (c) simultaneously a units in the X direction and b units in the Y direction.

Answer

(a) The scaling transformation applied to a point $P(x,y)$ produces the point $(ax,y).$ We can write this in matrix form as $S_{a,1}.P,$ or

$$\begin{pmatrix} a & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} ax \\ y \end{pmatrix}$$

(b) As in part (a), the required transformation can be written in matrix from as $S_{1,b}P$. So

$$\begin{pmatrix} 1 & 0 \\ 0 & b \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}=\begin{pmatrix} x \\ by \end{pmatrix}$$

(c) Scaling in both directions is described by the transformation $x' = ax$ and $y' = by$. Writing this in matrix form as $S_{ab}.P$, we have

$$\begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}=\begin{pmatrix} ax \\ by \end{pmatrix}$$

Problem 7 An Observer standing at the origin sees a point P(1,1). If the point is translated one unit in the direction v = I, its new coordinate position is P'(2,1). Suppose instead that the observer stepped back one unit along the x axis. What would be the apparent coordinates of P with respect to the observer?

Answer

The problem can be set as a transformation of coordinate system . If we translate the origin O in the direction v = –I (to a new position at O') the coordinates of P in this system can be found by the translation $\overline{T}v$.

$$\overline{T}_v\, P=\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}=\begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix}$$

So the new coordinates are (2,1)'. This has the following interpretation a displacement of one unit in a given direction can be achieved by either moving the object forward or steping back from it.

4.5 Shear Transformation

The shear transformation distorts an object by scaling one coordinate using the other. If distorts the shape of an object in such a way as if the object were composed of internal layers that has been caused to slide over each other is called shear. Two common shearing transformations are those that shift coordinate x values and those that shift y values.

2D Shear along X-direction

Shear in **X** direction is represented by the following set of equations.

$$\begin{aligned} Sh_x &= P_x + hP_y \\ Sh_y &= P_y \end{aligned}$$

where **h** is the negative or positive fraction of **Y** coordinate of **P** to be added to the **X** coordinate. Sh_x can be any real number.

The matrix of form of shear in x-direction is given by

$$\begin{bmatrix} 1 & 0 & 0 \\ Sh_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \textbf{(0.1)}$$

2D Shear along Y Direction

Similarly, shear along y-direction is given by

$$\begin{bmatrix} 1 & Sh_y & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \textbf{(0.2)}$$

Combining the shear in **X** and **Y** directions,

$$\begin{aligned} Sh_x &= P_x + hP_y \\ Sh_y &= gP_y + P_y \end{aligned}$$

where **g** is a non-zero fraction of P_x to be added to the **Y** coordinate

General matrix form of shear

The general matrix form of shear is

$$\begin{bmatrix} 1 & g \\ h & 1 \end{bmatrix} \quad \textbf{(0.3)}$$

Shear will reduce to a pure shear in the y-direction, when **h=0**.

The inverse of Shear is given by

$$\left[Sh^{-1} = frac11 - gh[\begin{matrix} 1 & -g \\ -h & 1 \end{matrix} \right] \qquad \textbf{(0.4)}$$

Problem 8

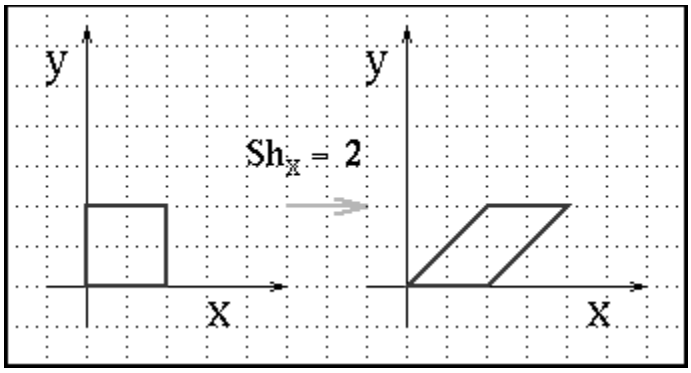
If h=0,5 g=0.8, then shear along X direction of the point P : (8,9) is obtained by substituting these values in (0.3).

$$Sh_x = 8 + (0.5 \times 9)9 = 12.5, 9 \qquad (0.5)$$

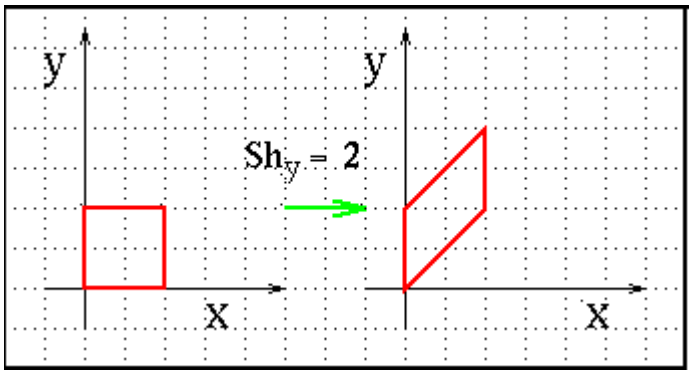
Shear in **Y** direction is

$$Sh_y = 8 + (0.5 \times 9)(0.8 \times 8) + 9 = 12.5, 15.4 \qquad (0.6)$$

$Sh_x = 2$, (2) $Sh_y = 2$



2 Consider a square of side = 2. Show the effect of shear when (1)



4.6 Summary

- Transformation is a process carried out by means of transformation to these object or changing the orientation of the object or may be combination of these.
- In translation, an object is displaced a given and direction from its original position
- If the new coordinate system is obtained by reflecting the old system about either *x* or *y* axis, the relationship between coordinates is given by the coordinate transformations
- Scaling is the process of expanding or compressing the dimension of an object
- Multiplying the basic matrix transformations can do complex transformations
- Shear transformation distorts an object by scaling one coordinate using the other in such a way as if the object were composed of internal layers that has been caused to slide over each other.

4.7 Key Words

Transformation, translation, scaling, rotation, mirror reflection, shear

4.8 Self Assessment Questions (SAQ)

1. Prove that the multiplication of the 3X3 matrices in 2-D geometry in each of the following operations is commutative, that is, independent of the order of executation:

(a) Two successive rotations

(b) Two successive translations

(c) Two successive scaling

2. Briefly explain the concept of 2D graphics.
3. What is inverse geometric transformations?
4. Show that the order in which transformations are performed is important by the transformation of triangle $A(1,0), B(0,1), C(1,1)$, by (a) rotating $45°$ about

the origin and then translating in the direction of vector I, and (b) translating and then rotating.

5. An object point P(x,y) is translated in the direction $v = aI + bJ$ and simultaneously an observer moves in the direction v. Show that there is no apparent motion(from the point of view of the observer) of the object point.

6. Show that reflection about the line y = x is attained by reversing coordinates. That is, $M_L(x,y) = (y,x)$

4.9 References/Suggested Readings

1. Computer Graphics (C Version), Donald Hearn and M. Pauline Baker, Prentice Hall,
2. Computer Graphics , Second Edition , by Pradeep K. Bhatia , I.K .International Publisher.
3. Advanced Animation and Rendering Techniques, Theory and Practice, Alan Watt and Mark Watt , ACM Press/Addison-Wesley
4. Graphics Gems I-V, various authors, Academic Press
5. Computer Graphics, Plastok, TMH
6. Principles of Interactive Computer Graphics, Newman, TMH

SUBJECT: COMPUTER GRAPHICS	
COURSE CODE: MCA-44(iv)	**AUTHOR: Abhishek Taneja**
LESSON NO. 5	
Graphics Operations	

Unit Structure

5.0 Objectives

At the end of this chapter the reader will be able to:

- Describe Windowing concepts
- Describe and distinguish between window and viewport
- Describe Normalized Device Coordiantes
- Describe Clipping

5.1 Introduction

In very basic two dimensional graphics usually use device coordinates. If any graphics primitive lies partially or completely outside the window then the portion outside will not be drawn. It is clipped out of the image. In many situations we have to draw objects whose dimensions are given in units completely incompatible with the screen coordinates system. Programming in device coordinates is not very convenient since the programmer has to do any required scaling from the coordinates natural to the application to device coordinates. This has led to two dimensional packages being developed which allow the application programmer to work directly in the coordiate system which is natural to the application. These user coordinates are usually called **World Coordinates (WC).** The packages then coverts the coordinates to **Device Coordinates (DC)** automatically. The transformation form the WC to DC is often carried out in tow steps. First using the **Normalisation Transformation** and then the **Workstation Transformation**. The **Viewing Transformation** is the process of going form a window in World coordinates to viewport in **Physical Device Coordinates (PDC).**

5.2 Window-to-Viewport Mapping

A window is specified by four world coordinates : wx_{min}, wx_{max}, wy_{min},and wy_{max} (see Fig. 5.1) Similarly, a viewport is described by four normalized device coordinates: vx_{min},vx_{max},vy_{min}, and vy_{max}. The objective of window – to – viewport mapping is to convert the world coordinates (wx, wy) of an arbitrary point to its corresponding normalized device coordinates (vx,vy). In order to maintain the same relative placement of the point in the viewport as in the window, we require:

$$\frac{wx - wx_{\min}}{wx_{\max} - wx_{\min}} = \frac{vx - vx_{\min}}{vx_{\max} - vx_{\min}} \quad \text{and} \quad \frac{wy - wy_{\min}}{wy_{\max} - wy_{\min}} = \frac{vy - vy_{\min}}{vy_{\max} - vy_{\min}}$$

Thus

$$\begin{cases} vx = \dfrac{vx_{\max} - vx_{\min}}{wx_{\max} - wx_{\min}}\left(wx - wx_{\min}\right) + vx_{\min} \\ vy = \dfrac{vy_{\max} - vy_{\min}}{wy_{\max} - wy_{\min}}\left(wy - wy_{\min}\right) + vy_{\min} \end{cases}$$

Since the eight coordinate values that define the window and the viewport are just constants, we can express these two formulas for computing (vx, vy) from (wx, wy) in terms of a translate-scale-translate transformation N

$$\begin{pmatrix} vx \\ vy \\ 1 \end{pmatrix} = N . \begin{pmatrix} wx \\ wy \\ 1 \end{pmatrix}$$

where

$$N = \begin{pmatrix} 1 & 0 & vx_{\min} \\ 0 & 1 & vy_{\min} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \dfrac{vx_{\max} - vx_{\min}}{wx_{\max} - wx_{\min}} & 0 & 0 \\ 0 & \dfrac{vy_{\max} - vy_{\min}}{wy_{\max} - wy_{\min}} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -wx_{\min} \\ 0 & 1 & -wy_{\min} \\ 0 & 0 & 1 \end{pmatrix}$$

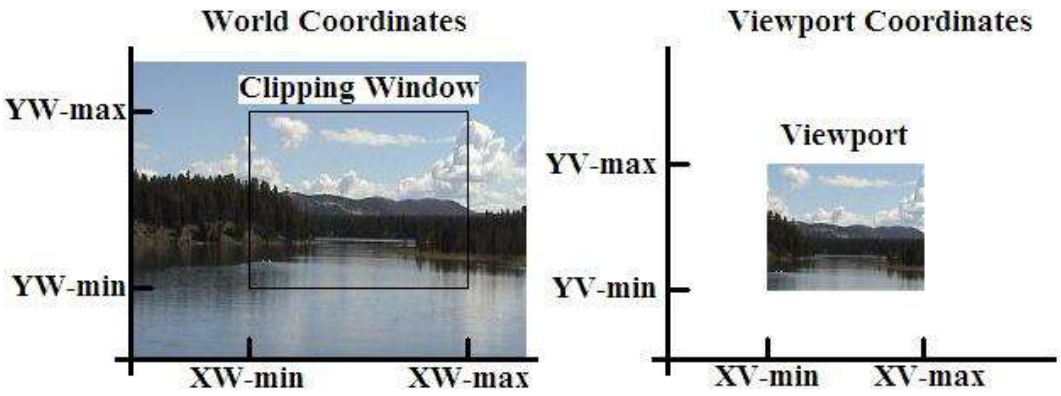


Fig. 5.1: Window-to-viewport mapping

Note that geometric distortions occur (e.g. squares in the window become rectangles in the viewport) whenever the two scalling constants differ.

5.3 Two – Dimensional Viewing and Clipping

Much like what we see in real life through a small window on the wall or the viewfinder of a camera, a Computer-generated image often depicts a partial view of a large scene. Objects are placed into the scene by modeling transformations to a master

coordinate system, commonly referred to as the world coordinate system (WCS). A rectangular window with its edge parallel to the axes of the WCS is used to select the portion of the scene for which an image is to be generated (see Fig. 5.2). Sometimes an additional coordinate system called the viewing coordinate system is introduced to simulate the effect of moving and / or tilting the camera.

On the other hand, an image representing a view often becomes part of a larger image, like a photo on an album page, which models a computer monitor's display area. Since album pages vary and monitor sizes differ from one system to another, we want to introduce a device-independent tool to describe the display area. This tool is called the normalized device coordinate system (NDCS) in which a unit (1 x 1) square whose lower left corner is at the origin of the coordinate system defines the display area of a virtual display device. A rectangular viewport with its edges parallel to the axes of the NDCS is used to specify a sub-region of the display area that embodies the image.

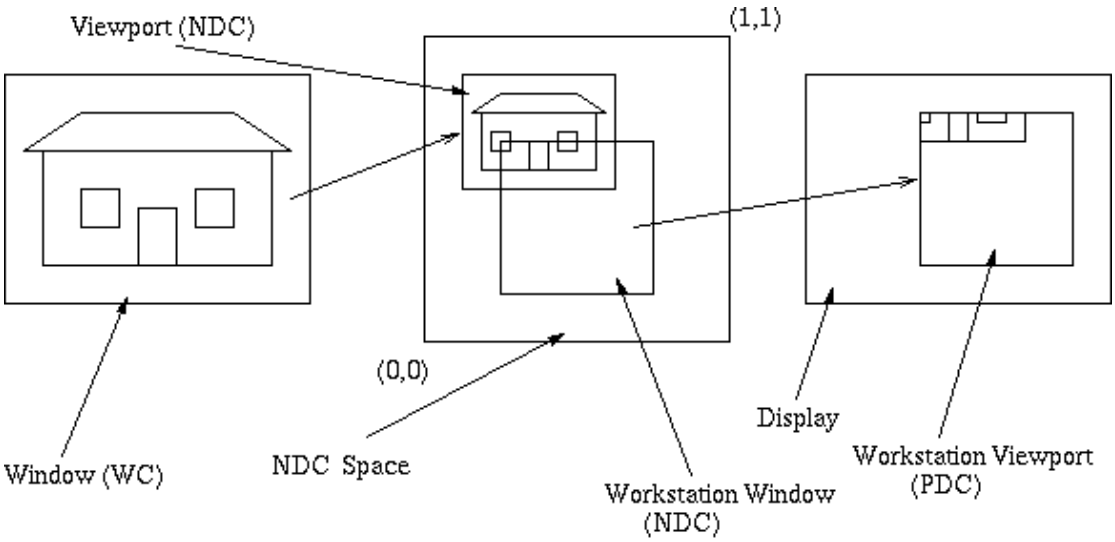


Fig. 5.2: Viewing transformation

The process that converts object coordinates in WCS to normalized device coordinate is called window–to– viewport mapping or normalization transformation. The process that maps normalized device coordinates to Physical Device Co-ordinates (PDC) / image coordinates is called work, station transformation, which is essentially a second window-to-viewport mapping., with a workstation window in the normalized device coordinate system and a workstation viewport in the device coordinate window in the normalized device coordinate system and a workstation viewport in the device

coordinate system. Collectively, these two coordinate mapping operations are referred to as viewing transformation.

Workstation transformation is dependent on the resolution of the display device/frame buffer. When the whole display area of the virtual device is mapped to a physical device that does not have a 1/1 aspect ratio, it may be mapped to a square sub-region (see fig. 5.2) so as to avoid introducing unwanted geometric distortion.

Along with the convenience and flexibility of using a window to specify a localized view comes the need for clipping, since objects in the scene may be completely inside the window, completely outside the window, or partially visible through the window. The clipping operation eliminates objects or portions of objects that are not visible through the window to ensure the proper construction of the corresponding image.

Note that clipping may occur in the world coordinate or viewing coordinate space, where the window is used to clip the objects; it may also occur in the normalized device coordinate space, where the viewport/workstation window is used to clip. In either case we refer to the window or the viewport/workstation window as the clipping window.

5.4 Point Clipping

Point clipping is essentially the evaluation of the following inequalities:

$$x_{min} \leq x \leq x_{max} \quad \text{and} \quad y_{min} \leq y \leq y_{max}$$

Where x_{min}, x_{max}, y_{min} and y_{max} define the clipping window. A point (x,y) is considered inside the window when the inequalities all evaluate to true.

5.5 Line Clipping

Lines that do not intersect the clipping window are either completely inside the window or completely outside the window. On the other hand, a line that intersects the clipping window is divided by the intersection point(s) into segments that are either inside or outside the window. The following algorithms provide efficient ways to decide the spatial relationship between an arbitrary line and the clipping window and to find intersection point(s).

5.5.1 The Cohen-Sutherland Algorithm

In this algorithm we divide the line clipping process into two phases: (1) identify those lines which intersect the clipping window and so need to be clipped and (2) perform the clipping.

All lines fall into one of the following clipping categories:

1. Visible – both endpoints of the line within the window

2. Not visible – the line definitely lies outside the window. This will occur if the line from $(x_1,y_1,)$ to (x_2,y_2) satisfies any one of the following four inequalities:

$$x_{1,}x_2 > x_{\max} \quad y_{1,}y_2 > y\max$$
$$x_{1,}x_2 < x_{\min} \quad y_{,}y_2 < y\min$$

3. Clipping candidate – the line is in neither category 1 and 2

In fig. 5.3, line l_1 is in category 1 (visible); lines l_2 and l_3 are in category 2 (not visible) ; and lines l_4 and l_5 are in category 3 (clipping candidate).

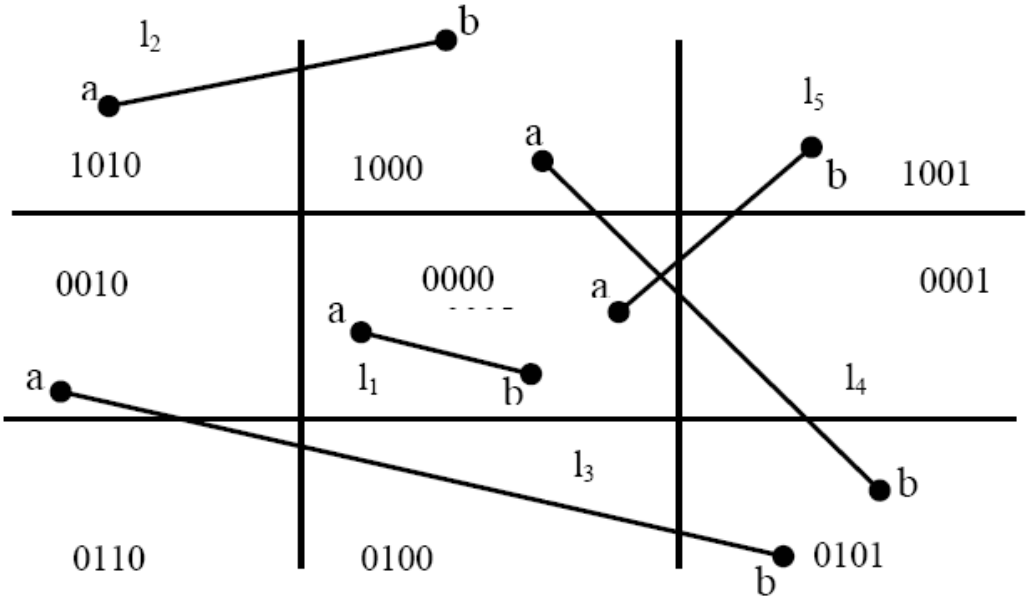
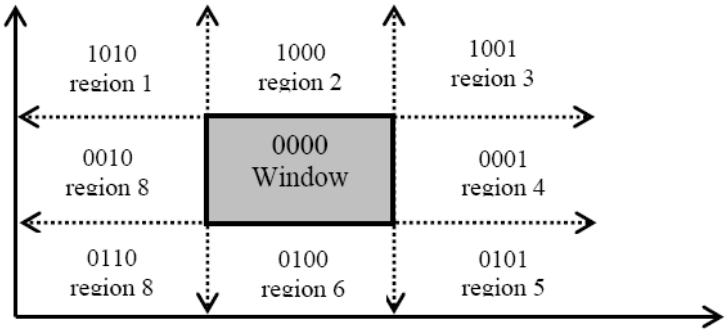


Figure 5.3

The algorithm employs an efficient procedure for finding the category of a line. It proceeds in two steps:

1. Assign a 4-bit region code to each endpoint of the line. The code is determined according to which of the following nine regions of the plane the endpoint lies in



Starting from the leftmost bit, each bit of the code is set to true (1) or false (0) according to the scheme

Bit 1 ≡ endpoint is above the window = sign ($y - y_{max}$)

Bit 2 ≡ endpoint is below the window = sign (y_{min}-y)

Bit 3 ≡ endpoint is to the right of the window = sign ($x - x_{max}$)

Bit 4 ≡ endpoint is to the left of the window = sign ($x_{min} - x$)

We use the convention that sign (a)=1 if a is positive, 0 otherwise. Of course, a point with code 0000 is inside the window.

2. The line is visible if both region codes are 0000, and not visible if the bitwise logical AND of the codes is not 0000, and a candidate for clipping if the bitwise logical AND of the region codes is 0000.

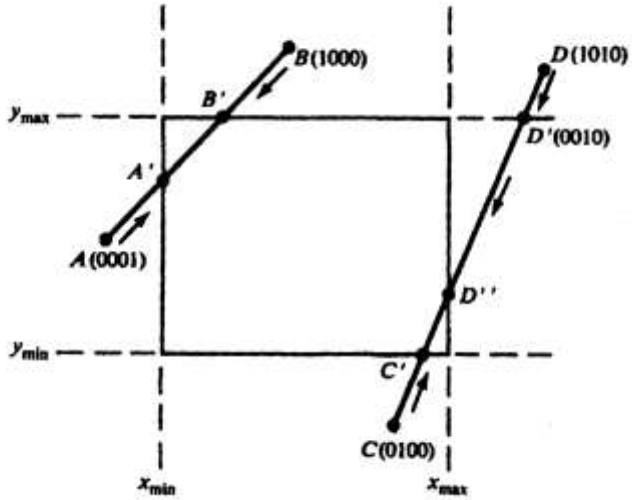


Figure 5.4

For a line in category 3 we proceed to find the intersection point of the line with one of the boundaries of the clipping window, or to be exact, with the infinite extension of one of the boundaries. We choose an endpoint of the line, say (x_1, y_1), that is outside the window, i.e., whose region code is not 0000. We then select an extended boundary line by observing that those boundary lines that are candidates for intersection are of ones for which the chosen endpoint must be "pushed across" so as to change a "1" in its code to a "0" (see Fig. 5.4). This means:

If bit 1 is 1, intersect with line $y = y_{max}$.

If bit 2 is 1, intersect with line $y = y_{min}$.

If bit 3 is 1, intersect with line $x = x_{max}$.

If bit 4 is 1, intersect with line $y = y_{min}$.

Consider line CD is Fig.5.4. If endpoint C is chosen, then the bottom boundary line $y=y_{min}$ is selected for computing intersection. On the other hand, if endpoint D is chosen, then either the top boundary line $y=y_{max}$ or the right boundary line $x = x_{max}$ is used. The coordinates of the intersection point are

$$\begin{cases} x_i = x_{\min} or\, x_{\max} \\ y_i = y_1 + m(x_i - x_1) \end{cases}$$ if the boundary line is vertical

or

$$\begin{cases} x_i = x_1 + (y_{i-} y_1)/m \\ y_i = y_{\min} or\, y_{\max} \end{cases}$$ if the boundary line is horizontal

where $m = (y_2 – y_1)/(x_2 – x_1)$ is the slope of the line.

Now we replace endpoint (x_1, y_1) with the intersection point (x_1, y_i) effectively eliminating the portion of the original line that is on the outside of the selected window boundary. The new endpoint is then assigned an updated region code and the clipped line re-categorized and handled in the same way. This iterative process terminates when we finally reach a clipped line that belongs to either category 1 (visible) or category 2 (not visible).

5.5.2 Midpoint Subdivision

An alternative way to process a line in category 3 is based on binary search. The line is divided at its midpoint into two shorter line segments. The clipping categories of the two new line segments are then determined by their region codes. Each segment in category 3 is divided again into shorter segments and categorized. This bisection and categorization process continues until each line segment that spans across a window boundary (hence encompasses an intersection point) reaches a threshold for line size and all other segments are either in category 1 (visible) or in category 2 (invisible) . The midpoint coordinates $(x_{m,}y_{m})$ of a line joining $(x_{1,}y_{1})$ and $(x_{2,}y_{2})$ are given by

$$x_m = \frac{x_1 + x_2}{2} \qquad y_m = \frac{y_1 + y_2}{2}$$

The example in fig.5.5 illustrates how midpoint subdivision is used to zoom in onto the two intersection points I_i and I_2 with 10 bisections. The process continues until we reach two line segments that are, say, pixel – sized . i.e. mapped to one single pixel each in the image space. If the maximum number of pixels in a line is M , this method will yield a pixel-sized line segment in N subdivisions, where 2^N = M or N = $\log_2$ M. For instance, when M = 1024 we need at most N = $\log_2$ 1024 = 10 subdivisions.

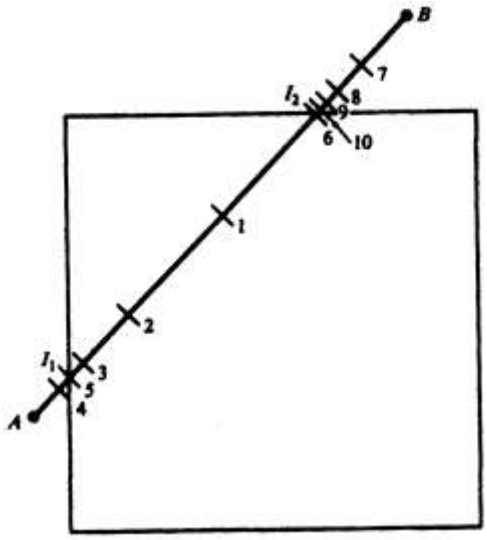


Figure 5.5

Problem 1 Let $S_x = \frac{vx_{max} - vx_{min}}{wx_{max}}$ and $s_y = \frac{vy_{max} - vy_{min}}{wy_{max} - wy_{min}}$

Express window-to-viewport mapping in the form of a composite transformation matrix.

Answer

$$N = \begin{pmatrix} 1 & 0 & vx_{min} \\ 0 & 1 & vx_{min} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -wx_{min} \\ 0 & 1 & -wx_{min} \\ 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} s_x & 0 & -s_x wx_{min} + vx_{min} \\ 0 & s_y & -s_y wy_{min} + vy_{min} \\ 0 & 0 & 1 \end{pmatrix}$$

Problem 2 Find the normalization transformation that maps a window whose lower left corner is at (1,1) and upper right corner is at (3, 5) onto (a) a viewport that is the entire normalized device screen and (b) a viewport that has lower left corner at (0, 0) and upper right corner $\left(\frac{1}{2}, \frac{1}{2}\right)$.

Answer

From problem 1, we need to identify the appropriate parameters

(a) The window parameters are $wx_{min} = 1$, $wy_{max} = 1$, and $wy_{max} = 5$. The viewport parameters are $vx_{min} = 0$, $vx_{max} = 1$, $vy_{min = 0,}$ and $vy_{max} = 1$. Then $s_x = \frac{1}{2}$, $s_y = \frac{1}{4}$, and

$$N = \begin{pmatrix} \frac{1}{2} & 0 & -\frac{1}{2} \\ 0 & \frac{1}{4} & -\frac{1}{4} \\ 0 & 0 & 1 \end{pmatrix}$$

(b) The window parameters are the same as in (a). The viewport parameters are not $vx_{min} = 0$, $vx_{max} = \frac{1}{2}$, $vy_{min} = 0$, $vy_{max} = \frac{1}{2}$. The $s_{x =} \frac{1}{4}$, $s_y = \frac{1}{8}$, and

$$N = \begin{pmatrix} \frac{1}{4} & 0 & -\frac{1}{4} \\ 0 & \frac{1}{8} & -\frac{1}{8} \\ 0 & 0 & 1 \end{pmatrix}$$

Problem 3 Find a normalization transformation from the window whose lower left corner is at (0,0) and upper right corner is at (4,3) onto the normalized device screen so that aspect ratios are preserved.

Answer

The window aspect ratio is $a_w = \frac{4}{3}$. Unless otherwise indicated , we shall choose a viewport that is as large as possible with respect to the normalized device screen. To this end, we choose the x extent from 0 to 1 and the y extent from 0 to $\frac{3}{4}$. So

$$a_v = \frac{1}{\frac{3}{4}} = \frac{4}{3}$$

As in Prob. 1 , with parameters $wx_{min} = 0$, $wx_{max} = 4$, $wy_{max} = 0$, $wy_{min} = 3$, and $vx_{max} = 1$, $vy_{min} = 0$, $vy_{max} = \frac{3}{4}$. ¾

$$N = \begin{pmatrix} \frac{1}{4} & 0 & 0 \\ 0 & \frac{1}{4} & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

5.6 Area Clipping

An algorithm that clips a polygon must deal with many different cases. The case is particularly note worthy in that the concave polygon is clipped into two separate polygons. All in all, the task of clipping seems rather complex. Each edge of the polygon must be tested against each edge of the clip rectangle; new edges must be added, and existing edges must be discarded, retained, or divided. Multiple polygons may result from clipping a single polygon. We need an organized way to deal with all these cases.

The following example illustrate a simple case of polygon clipping (figure5.6).

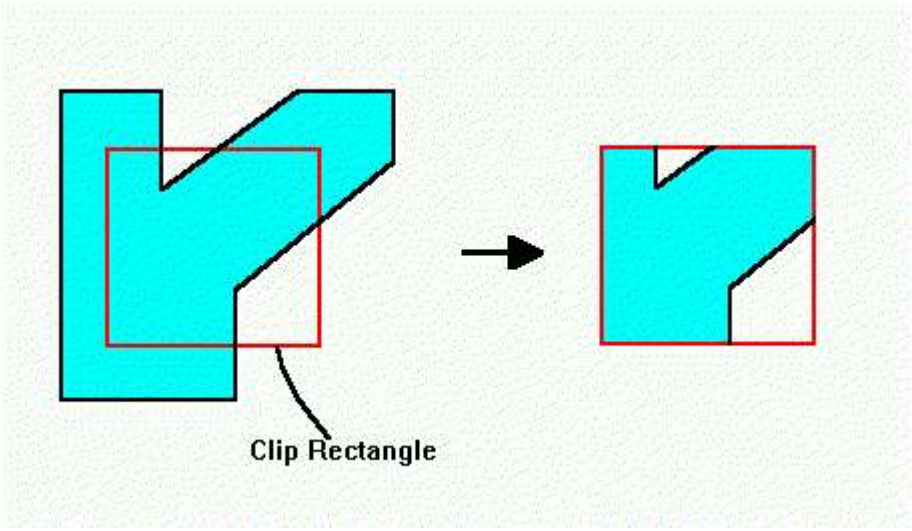


Figure 5.6

Sutherland and Hodgman's polygon-clipping algorithm uses a divide-and-conquer strategy: It solves a series of simple and identical problems that, when combined, solve the overall problem. The simple problem is to clip a polygon against a single infinite clip edge. Four clip edges, each defining one boundary of the clip rectangle, successively clip a polygon against a clip rectangle.

Note the difference between this strategy for a polygon and the Cohen-Sutherland algorithm for clipping a line: The polygon clipper clips against four edges in succession, whereas the line clipper tests the outcode to see which edge is crossed, and clips only when necessary.

5.6.1 Sutherland-Hodgman's polygon-clipping algorithm

- Polygons can be clipped against each edge of the window one at a time. Windows/edge intersections, if any, are easy to find since the X or Y coordinates are already known.
- Vertices which are kept after clipping against one window edge are saved for clipping against the remaining edges.
- Note that the number of vertices usually changes and will often increases.
- We are using the Divide and Conquer approach.

Four Cases of polygon clipping against one edge

The clip boundary determines a visible and invisible region. The edges from vertex i to vertex i+1 can be one of four types:

- Case 1 : Wholly inside visible region - save endpoint
- Case 2 : Exit visible region - save the intersection
- Case 3 : Wholly outside visible region - save nothing
- Case 4 : Enter visible region - save intersection and endpoint

Case 1 : Wholly inside the visible region, save the endpoint (Fig. 5.7).

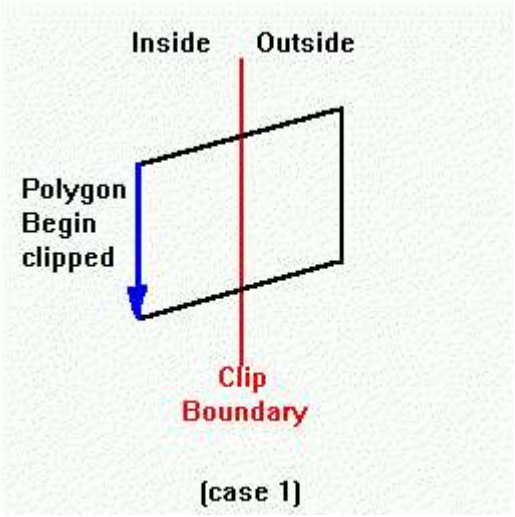


Figure 5.7

Case 2 : Exit visible region, save the intersection Figure 5.8

.

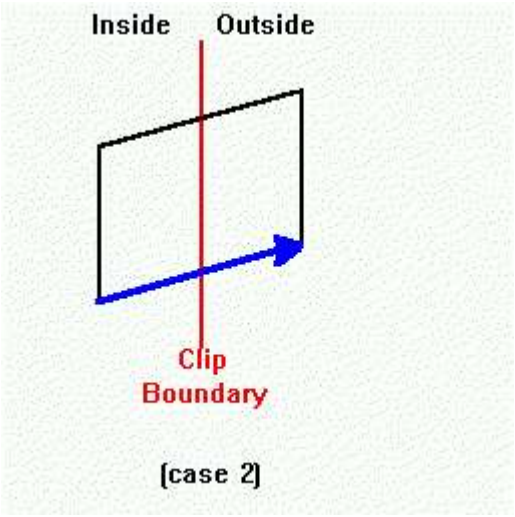


Figure 5.8

Case 3 : Wholly outside visible region, save nothing (Figure 5.9)

.

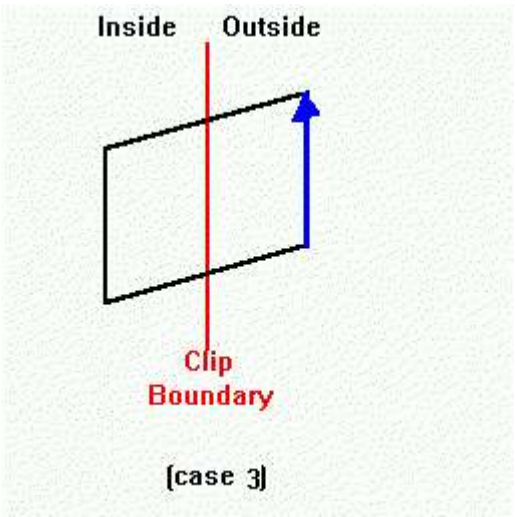


Figure 5.9

Case 4 : Enter visible region, save intersection and endpoint(Figure 5.10)

.

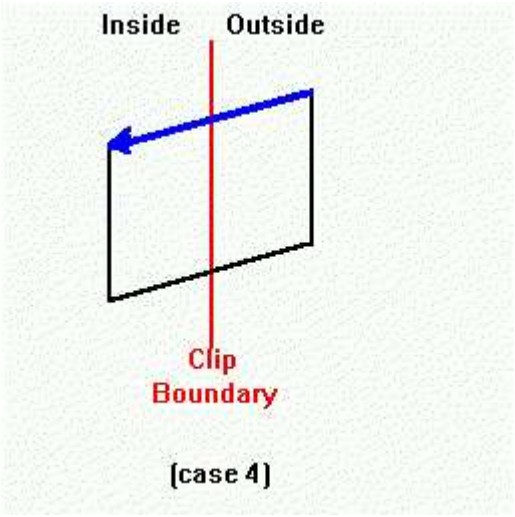


Figure 5.10

Because clipping against one edge is independent of all others, it is possible to arrange the clipping stages in a pipeline. The input polygon is clipped against one edge and any points that are kept are passed on as input to the next stage of the pipeline. This way four polygons can be at different stages of the clipping process simultaneously. This is often implemented in hardware.

5.7 Text Clipping

For text clipping various techniques are available in graphic packages. The clipping technique used will depend upon the application requirement and how characters produced. Normally we have many simple methods for clipping text. One method for processing character strings relative to a window boundary is to use the **all or none sting clipping** strategy shown in figure 5.11 below. If all of the string is discarded. This procedure is implemented by considering a boundary rectangle around the text pattern.

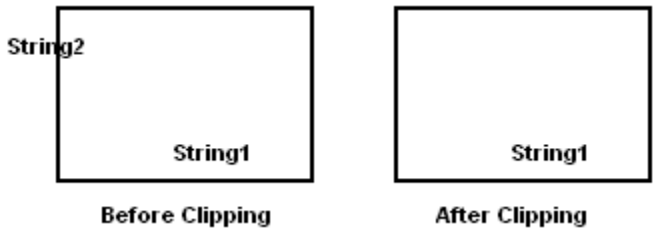


Figure 5.11

The boundary positions of the rectangle are the compared to the window boundaries, and the string is rejected if there is any overlap. This method produces fastest text clipping.

5.8 Window-To-Viewport Coordinate Transformation

Once object descriptions have been transferred to the viewing reference frame, we choose the window extents in viewing coordinates and select the viewport limits in normalized coordinates. Object descriptions are then transferred to normalized device coordinates. We do this using a transformation that maintains the same relative placement of objects in normalized space as they had in viewing coordinates. If a coordinate position is at the center of the viewing window, for instance, it will be displayed at the center of the viewport.

Figure 5.12 illustrates the window-to-viewport mapping. A point at position (xw, yw) in the window is mapped into position (xv, yv) in the associated view-port. To maintain the same relative placement in the viewport as in the window, we require that

$$\frac{xv - xv_{min}}{xv_{max} - xv_{min}} = \frac{xw - xw_{min}}{xw_{max} - xw_{min}}$$

$$\frac{yv - yv_{min}}{yv_{max} - yv_{min}} = \frac{yw - yw_{min}}{yw_{max} - yw_{min}}$$

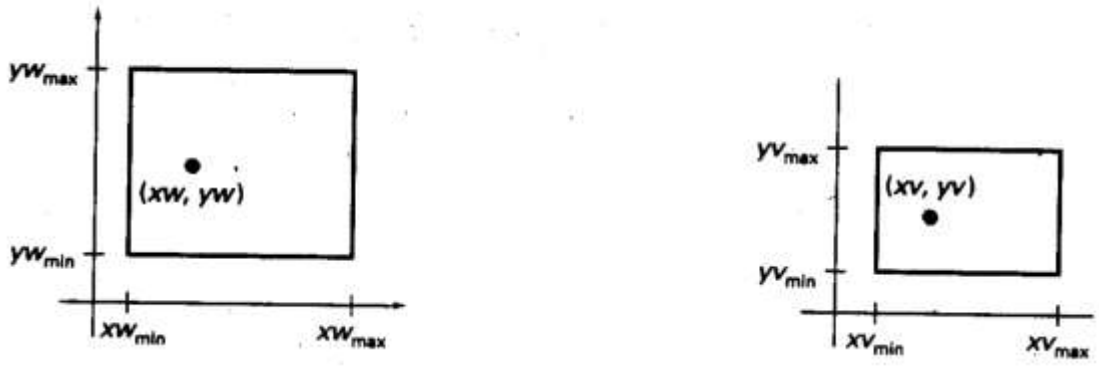


Figure 5.12: A point at position (xm, yw) in a designated window is mapped to viewport coordinates
(xv, yv) so that relative positions in the two areas are the same.

Solving these expressions for the viewport position (xv, yv), we have

$xv = xv_{min} + (xw - xw_{min})sx$

$yv = yv_{min} + (yw - yw_{min})sy$

where the scaling factors are

$$sx = \frac{xv_{max} - xv_{min}}{xw_{max} - xw_{min}}$$

$$sy = \frac{yv_{max} - yv_{min}}{yw_{max} - yw_{min}}$$

Above equations can also be derived with a set of transformations that converts the window area into the viewport area. This conversion is performed with the following sequence of transformations:

1. Perform a scaling transformation using a fixed-point position of (xw_{min}, yw_{min}) that scales the window area to the size of the viewport.
2. Translate the scaled window area to the position of the viewport.

Relative proportions of objects are maintained if the scaling factors are the same (sx = sy). Otherwise, world objects will be stretched or contracted in either x or y direction when displayed on the output device.

Character strings can be handled in two ways when they are mapped to a viewport. The simplest mapping maintains a constant character size, even though the viewport

area may be enlarged or reduced relative to the window. This method would be employed when text is formed with standard character fonts that cannot be changed. In systems that allow .for changes in character size, string definitions can be windowed the same as other primitives. For characters formed with line segments, the mapping to the viewport can be carried out as a sequence of line transformations.

From normalized coordinates, object descriptions are mapped to the viewport display devices. Any number of output devices can be open in a particular application, and another window-to-viewport transformation can be performed for each open output device. This mapping, called the workstation transformation, is accomplished by selecting a window area in normalized space and a viewport area in the coordinates of the display device. With the workstation transformation, we gain some additional control over the positioning of parts of a scene on individual output devices. As illustrated in Fig. 5.13, we can use work station transformations to partition a view so that different parts of normalized space can be displayed on different output devices.

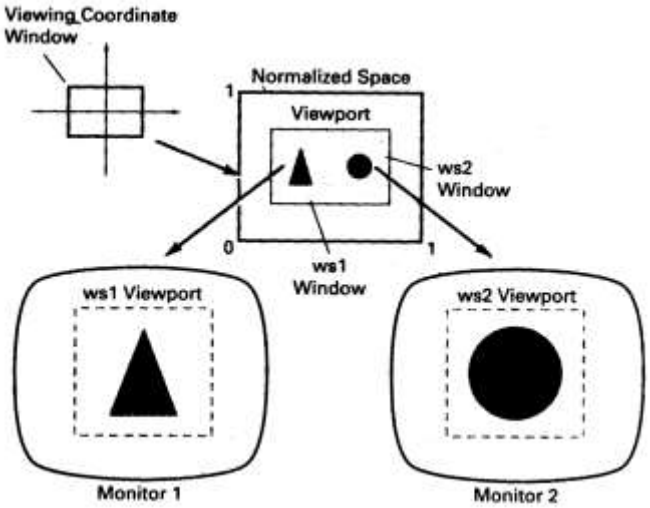


Figure 5.13: Mapping selected parts of a scene in normalized coordinated to different video monitors with workstation transformations.

5.9 Exterior and Interior Clipping

So far, we have considered only procedures for clipping a picture to the interior of a region by eliminating everything outside the clipping region. What is saved by these procedures is inside the region. In some cases, we want to do the reverse, that is, we want to clip a picture to the exterior of a specified region. The picture parts to be saved are those that are outside the region. This is referred to as exterior clipping.

A typical example of the application of exterior clipping is in multiple-window systems. To correctly display the screen windows, we often need to apply both internal and external clipping. Objects within a window are clipped to the interior of that window. When other higher-priority windows overlap these objects, the objects are also clipped to the exterior of the overlapping windows.

Exterior clipping is used also in other applications that require overlapping pictures. Examples here include the design of page layouts in advertising or publishing applications or for adding labels or design patterns to a picture. The technique can also be used for combining graphs, maps, or schematics. For these applications, we can use exterior clipping to provide a space for an insert into a larger picture.

Procedures for clipping objects to the interior of concave polygon windows can also make use of external clipping. A figure 5.14 shows a line $\overline{P_1P_2}$ that is to be clipped to the interior of a concave window with vertices $V_1V_2V_3V_4V_5$. Line $\overline{P_1P_2}$ can be clipped in two passes: (1) First, $\overline{P_1P_2}$ is clipped to the interior of the convex polygon $V_1V_2V_3V_4$ to yield the clipped segment $\overline{P'_1P'_2}$. (2) Then an external clip of $\overline{P'_1P'_2}$ is performed against the convex polygon $V_1V_5V_4$ to yield the final clipped line segment $\overline{P''_1P'_2}$.

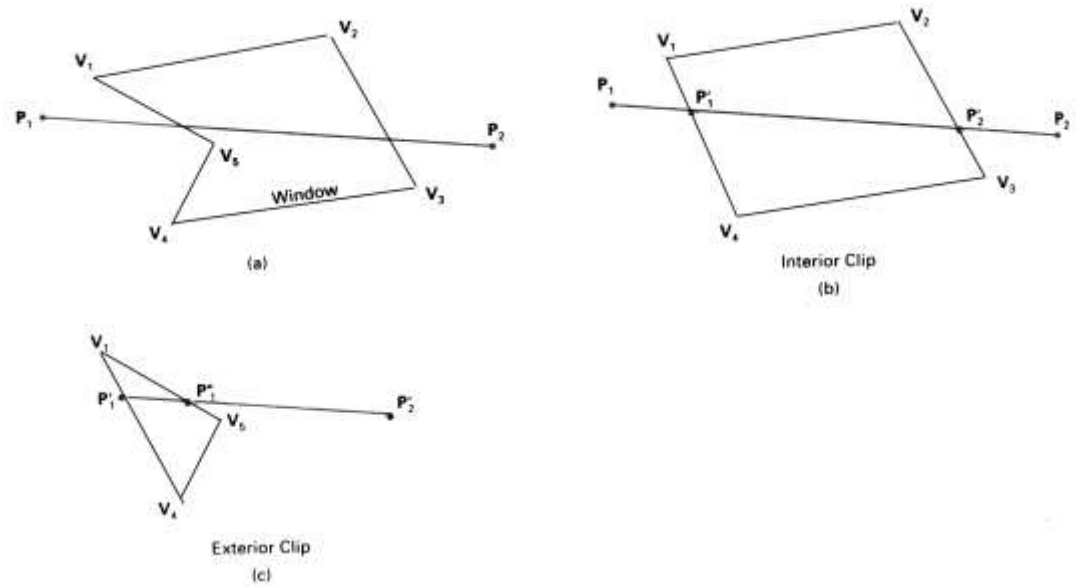


Figure 5.14: Clipping line $\overline{P_1P_2}$ to the interior of a concave polygon with vertices $V_1V_2V_3V_4V_5$ (a), using convex polygons $V_1V_2V_3V_4$ (b) and $V_1V_5V_4$ (c), to produce the clipped line P"$_1$P'$_2$

5.10 Summary

- The viewing transformation is used to transform a part of graphical scene to a portion of the screen.
- The Normalised Device Coordinates describe the coordinates system natural
- A window is a rectangular surrounding the object or a part of it that we wish to draw on the screen.
- A viewport beings the window content to the screen, so, that window coordinates are based on world coordinates system.
- Multiplying the basic matrix transformations can do complex transformations
- Shear transformation distorts an object by scaling one coordinate using the other in such a way as if the object were composed of internal layers that has been caused to slide over each other.

5.11 Key Words

Window, Viewport, Clipping

5.12 Self Assessment Questions (SAQ)

1. Explain The Cohen-Sutherland Line-Clipping Algorithm.
2. Explain some uses of clipping.
3. What is clipping?
4. Explain clipping in a raster world?
5. What do you mean by viewing transformation?
6. Find the workstation transformation that maps the normalized device screen onto a physical device whose x extent is 0 to 199 and y extent is 0 to 639 where origin is located at the lower left corner.

5.13 References/Suggested Readings

1. Computer Graphics (C Version), Donald Hearn and M. Pauline Baker, Prentice Hall,
2. Computer Graphics , Second Edition , by Pradeep K. Bhatia , I.K .International Publisher.
3. Advanced Animation and Rendering Techniques, Theory and Practice, Alan Watt and Mark Watt , ACM Press/Addison-Wesley
4. Graphics Gems I-V, various authors, Academic Press
5. Computer Graphics, Plastok, TMH
6. Principles of Interactive Computer Graphics, Newman, TMH

SUBJECT: COMPUTER GRAPHICS	
COURSE CODE: MCA-44(iv)	**AUTHOR: Abhishek Taneja**
LESSON NO. 6	
Interactive Graphics	

Unit Structure

6.0 Objectives

At the end of this chapter the reader will be able to:

- Describe the concept of pointing and positioning
- Describe various interactive graphic devices
- Describe various interactive graphic techniques including positioning, constraints, grid, gravity field, inking, painting, dragging, and rubber band techniques

6.1 Introduction

As hardware cost is plummeting, which is considered as the major bottleneck for the progress; now communication devices is more listened for better development. For that reason, techniques for developing high-quality user interfaces are moving to the forefront in computer science and are becoming the "last frontier" in providing computing to a wide variety of users—as other aspects of technology continue to improve, but the human users remain the same. Interest in the quality of user-computer interfaces is a recent part of the formal study of computers. The emphasis until the early 1980s was on optimizing two scarce hardware resources, computer time and memory. Program efficiency was the highest goal. With today's plummeting hardware costs and powerful graphics-oriented personal computing environments the focus turns to optimizing user efficiency rather than computer efficiency. Thus, although many of the ideas presented in this chapter require additional CPU cycles and memory space, the potential rewards in user productivity and satisfaction well outweigh the modest additional cost of these resources. The quality of the user interface often determines whether users enjoy or despise a system, whether the designers of the system are praised or damned, whether a system succeeds or fails in the market. Actually, a poor user interface such as in air traffic control or in nuclear power plant monitoring can lead to catastrophic consequences.

The desktop user-interface metaphor, with its windows, icons, and pull-down menus, all making heavy use of raster graphics, is popular because it is easy to learn and requires little typing skill. Most users of such systems are not computer programmers and have little sympathy for the old-style difficult-to-learn keyboard-oriented command-language interfaces that many programmers take for granted. The designer of an interactive graphics application must be sensitive to users' desire for easy-to-learn yet powerful interfaces. In this chapter, we discuss the three basic low-level

elements of user interfaces: input devices, interaction techniques, and interaction tasks. **Interaction techniques are the primitive building blocks from which a user interface is crafted.**

We focus in this chapter on input devices—those pieces of hardware by which a user enters information into a computer system. Input devices for the earliest computers were switches and knobs, jumper wires placed in patch boards, and punched cards. These were followed by the teletype, the text-only forerunner of today's interactive terminals. The mouse and keyboard now predominate, but a wide variety of input devices can be used. **An interaction task is the entry of a unit of information by the user. Basic interaction tasks are *position, text, select,* and *quantify*.** The unit of information that is input in a position interaction task is of course a position; the text task yields a text string; the select task yields an object identification; and the quantify task yields a numeric value. A designer begins with the interaction tasks necessary for a particular application. For each such task, the designer chooses an appropriate interaction device and interaction technique. Many different *interaction techniques* can be used for a given interaction task, and there may be several different ways of using the same device to perform the same task. For instance, a selection task can be carried out by using a mouse to select items from a menu, using a keyboard to enter the name of the selection, pressing a function key, circling the desired command with the mouse, or even writing the name of the command with the mouse. Similarly, a single device can be used for different tasks: A mouse is often used for both positioning and selecting.

Interaction tasks are defined by *what* the user accomplishes, whereas logical input devices categorize *how* that task is accomplished by the application program and the graphics system. Interaction tasks are user-centered, whereas logical input devices are a programmer and graphics-system concept. By analogy with a natural language, single actions with input devices are similar to the individual letters of the alphabet from which words are formed. The sequence of input-device actions that makes up an interaction technique is analogous to the sequence of letters that makes up a word. A word is a unit of meaning; just as several interaction techniques can be used to carry out the same interaction task, so too words that are synonyms convey the same meaning. An interactive dialogue is made up of interaction-task sequences, just as a sentence is constructed from word sequences.

6.2 Concept of Positioning and Pointing

Most display terminals provide the user with an alphanumeric keyboard with which to type commands and enter data for the program. For some applications, however, the keyboard is inconvenient or inadequate. For example, the user may wish to indicate one of a number of symbols on the screen, in order to erase the symbol. If each symbol is labeled, he can do so by typing the symbol's name; by pointing at the symbol, however, he may be able to erase more rapidly, and the extra clutter of labels can be avoided.

Another problem arises if the user has to add lines or symbols to the picture on the screen. Although he can identify an items's position by typing coordinates he can do so even better by pointing at the screen, particularly if what matters most is the items's position relative to the rest of the picture.

These two examples illustrate the two basic types of graphical interaction: pointing at items already on the screen and positioning new items. The need to interact in these ways has stimulated the developed of a number of different types of graphical input device, some of which are described in this chapter.

Ideally a graphical input device should lend itself both to pointing and to positioning. In reality there are no devices with this versatility. Most devices are much better at positioning than at pointing; one device, the light pen, is the exact opposite. Fortunately, however we can supplement the deficiencies of these devices by software and in this way produce hardware-software system that has both capabilities. Nevertheless the distinction between pointing and positioning capability is extremely important.

Another important distinction is between devices that can be used directly on the screen surface and devices that cannot. The latter might appear to be less useful, but this is far from true. Radar operators and air-traffic controllers have for years used devices like the joystick and the tracker ball neither of which can be pointed at the screen. The effectiveness of these input devices depends on the use of visual feedback: the x and y outputs of the device control the movement of a small cross, or cursor, displayed on the screen. The user of the device steers the cursor around the screen as if it were a toy boat on the surface of a pond. Although this operation sounds as if it requires a lot of skill, it is in fact very easy.

The use of visual feedback has an additional advantage: just as in any control system, it compensates for any lack of linearity in the device. A linear input device is one that faithfully increases or decreases the input coordinate value in exact proportion to the user's hand movement. If the device is being used to trace a graph or a map. Linearity is important. A cursor, however, can be controlled quite easily even if the device behaves in a fairly nonlinear fashion. For example, the device may be much less sensitive near the left – hand region of its travel: a 1 – inch hand movement may change the x value by only 50 units, whereas the same movement elsewhere may change x by 60 units. The user will simply change his hand movement to compensate, often without even noticing the no linearity. This phenomenon has allowed simple, inexpensive devices like the mouse to be used very successfully for graphical input.

6.3 Interactive Graphic 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.

6.3.1 Keyboards

The well-known QWERTY keyboard has been with us for many years. It is ironic that this keyboard was originally designed to *slow down* typists, so that the typewriter hammers would not be so likely to jam. Studies have shown that the newer Dvorak keyboard , which places vowels and other high-frequency characters under the home positions of the fingers, is somewhat faster than is the QWERTY design. It has not been widely accepted. Alphabetically organized keyboards are sometimes used when many of the users are non typists. But more and more people are being exposed to QWERTY keyboards, and experiments have shown no advantage of alphabetic over QWERTY keyboards .In recent years, the chief force serving to displace the keyboard has been the shrinking size of computers, with laptops, notebooks, palmtops, and personal digital assistants. The typewriter keyboard is becoming the largest component of such pocket-sized devices, and often the main component standing in the way of reducing its overall size. The *chord keyboard* has five keys similar to piano keys, and is operated with one hand, by pressing one or more keys simultaneously to

"play a chord." With five keys, 31 different chords can be played. Learning to use a chord keyboard (and other similar stenographer style keyboards) takes longer than learning the QWERTY keyboard, but skilled users can type quite rapidly, leaving the second hand free for other tasks. This increased training time means, however, that such keyboards are not suitable substitutes for general use of the standard alphanumeric keyboard. Again, as computers become smaller, the benefit of a keyboard that allows touch typing with only five keys may come to outweigh the additional difficulty of learning the chords. Other keyboard-oriented considerations, involving not hardware but software design, are arranging for a user to enter frequently used punctuation or correction characters without needing simultaneously to press the control or shift keys, and assigning dangerous actions (such as delete) to keys that are distant from other frequently used keys.

6.3.2 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. 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 beams that are interrupted identify the horizontal and vertical coordinates of the screen position selected. Positions can be selected with an accuracy of about 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. 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.

6.3.3 Light pens

The pencil-shaped devices 's are used to select screen positions by detecting the light coming from point 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 implementation 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 sometime give false readings due to background lighting in a room.

6.3.4 Graphics Tablets

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 positions on the

tablet. 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 4 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 wire embedded in the tablet surface. Electromagnetic pulses are generated in sequence 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, a their signal strength, coded pulses, or phase shifts can be used to determine the position on the tablet.

6.3.5 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. 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.

6.3.6 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 direction 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.

6.3.7 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 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. 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.

6.3.8 Other Devices

Here we discuss some of the less common, and in some cases experimental, 2D interaction devices. Voice recognizers, which are useful because they free the user’s hands for other uses, apply a pattern-recognition approach to the waveforms created when we speak a word. The waveform is typically separated into a number of different frequency bands, and the variation over time of the magnitude of the waveform. in each band forms the basis for the pattern matching. However, mistakes can occur in the pattern matching, so it is especially important that an application using a recognizer provide convenient correction capabilities. Voice recognizers differ in whether or not they must be trained to recognize the waveforms of a particular speaker, and whether they can recognize connected speech as opposed to single words or phrases. Speaker-independent recognizers have very limited vocabularies—typically, they include only the ten digits and 50 to 100 words. Some discrete word recognizers can recognize vocabularies of thousands of different words after appropriate training. But if the user has a cold, the recognizer must be retrained. The

user of a discrete word recognizer must pause for a fraction of a second after each word to cue the system that a word end has occurred. The more difficult task of recognizing connected speech from a limited vocabulary can now be performed by off-the-shelf hardware and software, but with somewhat less accuracy. As the vocabulary becomes larger, however, artificial-intelligence techniques are needed to exploit the context and meaning of a sequence of sentences to remove ambiguity. A few systems with vocabularies of 20,000 or more words can recognize sentences such as "Write Mrs. Wright a letter right now!" Voice synthesizers create waveforms that approximate, with varying degrees of realism, spoken words. The simplest synthesizers use *phonemes*, the basic sound units that form words. This approach creates an artificial-sounding, inflection-free voice. More sophisticated phoneme-based systems add inflections. Other systems actually play back digitized spoken words or phrases. They sound realistic, but require more memory to store the digitized speech. Speech is best used to augment rather than to replace visual feedback, and is most effective when used sparingly. For instance, a training application could show a student a graphic animation of some process, along with a voice narration describing what is being seen. See for additional guidelines for the effective application of speech recognition and generation in user-computer interfaces, and for an introduction to speech interfaces, and for speech recognition technology. The data tablet has been extended in several ways. Many years ago, Herot and Negroponte used an experimental pressure-sensitive stylus : High pressure and a slow drawing speed implied that the user was drawing a line with deliberation, in which case the line was recorded exactly as drawn; low pressure and fast speed implied that the line was being drawn quickly, in which case a straight line connecting the endpoints was recorded. Some commercially available tablets sense not only stylus pressure but orientation as well. The resulting 5 degrees of freedom reported by the tablet can be used in various creative ways. For example, Bleser, Sibert, and McGee implemented the GWPaint system to simulate various artist’s tools, such as an italic pen, that are sensitive to pressure and orientation. An experimental touch tablet, developed by Buxton and colleagues, can sense multiple finger positions simultaneously, and can also sense the area covered at each point of contact The device is essentially a type of touch panel, but is used as a tablet on the work surface, not as a touch panel mounted over the screen. The device can be used in a rich variety of ways . Different finger pressures correlate with the area covered at a point of contact, and are used to signal user

commands: a light pressure causes a cursor to appear and to track finger movement; increased pressure is used, like a button-push on a mouse or puck, to begin feedback such as dragging of an object; decreased pressure causes the dragging to stop.

6.4 Interactive Graphical Techniques

There are several techniques that are incorporated into graphics packages to aid the interactive construction of pictures. Various input options can be provided, so that coordinate information entered with locator and stroke devices can be adjusted or interpreted according to a selected option. For example, we can restrict all lines to be either horizontal or vertical. Input coordinates can establish the position or boundaries for objects to be drawn, or they can be used to rearrange previously displayed objects.

6.4.1 Basic Positioning Methods

Coordinate values supplied by locator input are often used with positioning methods to specify a location for displaying an object or a character string. We interactively select coordinate positions with a pointing device, usually by positioning the screen cursor. Just how the object or text-string positioning is performed depends on the selected options. With a text string, for example, the screen point could be taken as the center string position, or the start or end position of the string, or any of the other string-positioning options. For lines, straight line segments can be displayed between two selected screen positions:

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

6.4.2 Constraints

With some applications, certain types of prescribed orientations or object alignments are useful. A constraint is a rule for altering input-coordinate values to produce a specified orientation or alignment of the displayed coordinates. There are many kinds of constraint functions that can be specified, but the most common constraint is a horizontal and vertical alignment of straight lines. This type of constraint, shown in Figs. 6.1 and 6.2, is useful in forming network layouts. With this constraint, we can

create horizontal and vertical lines without worrying a-bout precise specification of endpoint coordinates.

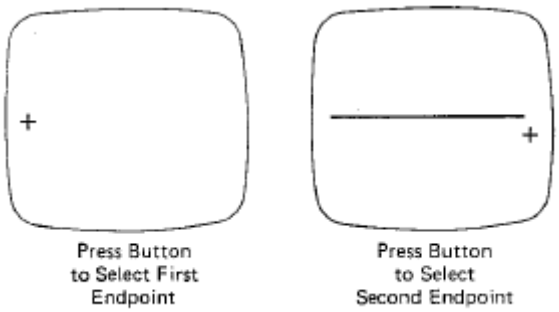


Figure 6.1: Horizontal line constraint

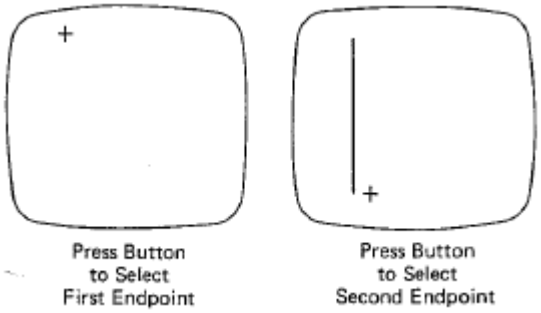


Figure 6.2: Vertical line constraint

A horizontal or vertical constraint is implemented by determining whether any two input coordinate endpoints are more nearly horizontal or more near 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.

6.4.3 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 intersection of two grid lines. Figure 6.3 illustrates line drawing with 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.

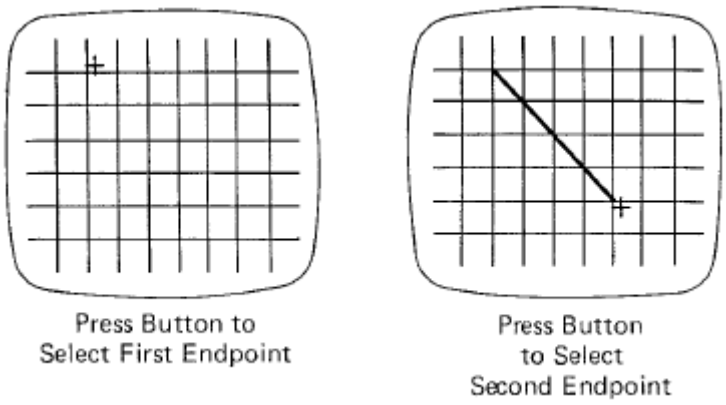


Figure 6.3: Line drawing using a grid

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.

6.4.4 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. 6.4. 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.

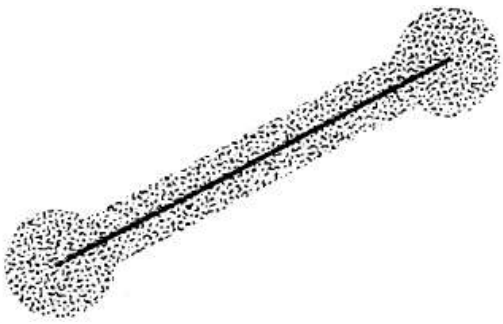


Figure 6.4: Gravity field around a line. Any selected point in the shaded area is shifted to a position on the line

6.4.5 Rubber-Band Methods

Straight lines can be constructed and positioned using rubber-band method which stretch out a line from a starting position as the screen cursor is move Figure 6.5 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 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.

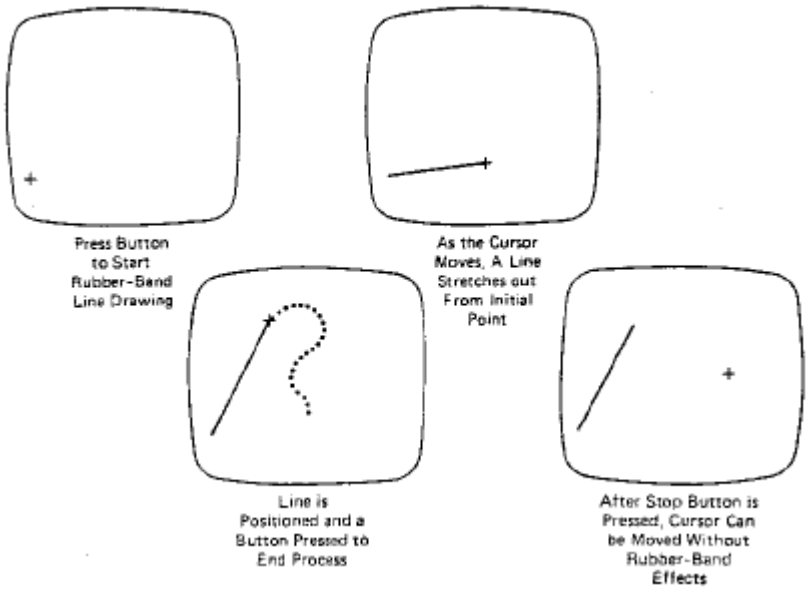


Figure 6.5: Rubber-band method for drawing and positioning a straight line segment

Rubber-band methods are used to construct and position other objects besides straight lines. Figure 6.6 demonstrates rubber-band construction of a rectangle, and Fig. 6.7 shows a rubber-band circle construction.

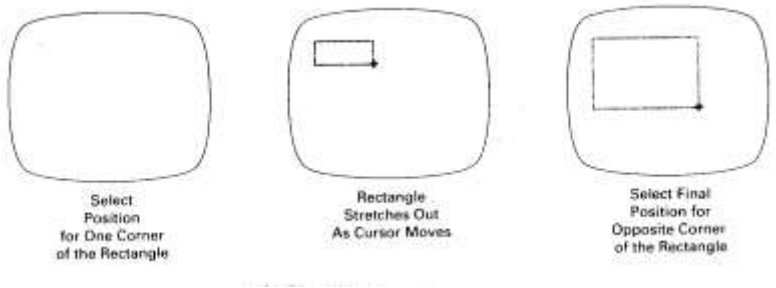


Figure 6.6: Rubber-band method for constructing a rectangle

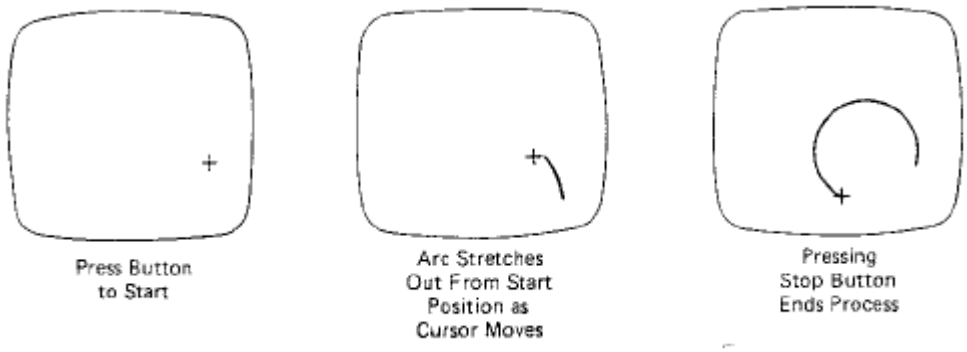


Figure 6.7: Constructing a circle using a rubber-band method

6.4.6 Sketching

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.

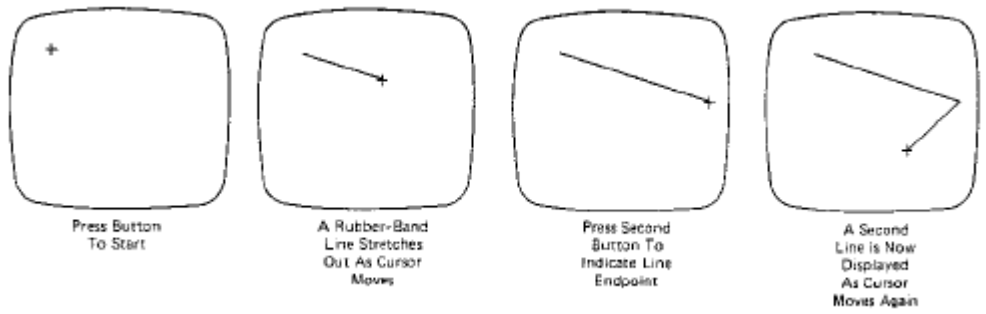


Figure 6.8 Uses rubber band methods to create objects consisting of connected line segments

Line widths, line styles, and other attribute options are also commonly found in painting and drawing packages. 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.

6.4.7 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 scene is useful in applications where we might want to explore different possibilities before selecting a final location.

6.4.8 Inking and Painting

If we sample the position of a graphical input device at regular intervals and display a dot at each sampled position, a trail will be displayed of the movement of the device. This technique, which closely simulates the effect of drawing on paper, is called inking. For many years the main use of inking has been in conjunction with on-line character-recognition programs. With the advent of high-quality raster displays the technique has found wider use for painting purposes.

6.4.9 Painting

A raster display incorporating a random-access frame buffer, can be treated as a painting surface for interactive purposes. As the user moves the cursor around, a trace of its path can be left on the screen. The user can build up freehand drawings of surprisingly good quality.

It is possible to provide a range of tools for painting on a raster display: these tools take the form of brushes that lay down trails of different thick nesses and colors. For example, instead of depositing a single dot at each sampled input position, the program can insert a group of dots so as to fill in a square or circle: the result will be a much thicker trace. On a "black-and-white display the user needs brushes that paint in both black and white, so that information can be both added and removed (Figure 6.9). When a color display is used for painting, a menu of different colors can be provided.

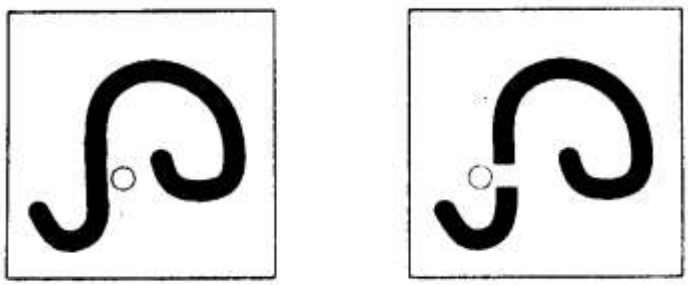


Figure 6.9: Erasing with a white brush

6.5 Summary

- An interaction technique is a way of using a physical input/output device to perform a generic interaction task in a human-computer dialogue. It represents an abstraction of some common class of interactive task, for example, choosing one of several objects shown on a display screen, so it is not bound to a single application.
- The basic interaction tasks for interactive graphics are positioning, selecting, entering text, and entering numeric quantities.
- Input functions available in a graphics package can be defined in three input modes. Request mode places input under the control of the application program. Sample mode allows the input devices and program to operate concurrently. Event

mode allows input devices to initiate data entry and control processing of data. Once a mode has been chosen for a logical device class and the particular physical device to be used to enter this class of data, input functions the program are used to enter data values into the program. An application program can make simultaneous use of several physical input devices operating in different modes.

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

6.6 Key Words

Positioning, Pointing, Interactive Graphics, Inking, Painting, Dragging

6.7 Self Assessment Questions (SAQ)

1. Discuss some hardware interaction devices with their advantages and disadvantages.
2. Define in precise the various interactive techniques?
3. Discuss how pointing techniques are different are different form positioning techniques.
4. Discuss the functional characteristics of light pen and mouse.
5. Write a short note on Interactive computer Graphic device?
6. Discuss the following in details
    a) Rubber Band methods
    b) Inking and painting
    c) Dragging
    d) Constrained painting

6.8 References/Suggested Readings

1. High Resolution Computer Graphics using Pascal/C, by Ian O. Angell and Gareth Griffith, John Wiley & Sons
2. Computer Graphics , Second Edition , by Pradeep K. Bhatia , I.K .International Publisher.

3. Computer Graphics (C Version), Donald Hearn and M. Pauline Baker, Prentice Hall

SUBJECT: COMPUTER GRAPHICS

COURSE CODE: MCA-44(iv)	AUTHOR: Abhishek Taneja

LESSON NO. 7

Three Dimensional Transformations

UNIT STRUCTURE

7.0 Objectives

At the end of this chapter the reader will be able to:

- Describe three dimensional transformations and viewing and clipping
- Describe parallel and perspective projections
- Describe concept of hidden line and surface elimination methods like Z-Buffer, Scan line, Painter's Subdivision
- Describe Wireframe model
- Describe Bezier curves and surfaces

7.1 Introduction

We now turn to transformations in three dimensions. In most cases, the mathematics of linear transformations is easy to extend from two dimensions to three, but the discussion here demonstrates that certain transformations, most notably rotations, are more complex in three dimensions because there are more directions about which to rotate and because the simple terms clockwise and counterclockwise no longer apply. We start with a short discussion of coordinate systems in three dimensions.

In two dimensions, there is only one Cartesian coordinate system, with two perpendicular axes labeled *x* and *y* (actually, the axes don't have to be perpendicular, but this is irrelevant for our discussion of transformations). A coordinate system in three dimensions consists similarly of three perpendicular axes labeled *x*, *y*, and *z*, but there are two such systems, a left-handed and a right-handed (Figure 7.1a), and they are different. A right-handed coordinate system is constructed by the following rule. Align your right thumb with the positive *x* axis and your right index finger with the positive *y* axis. Your right middle finger will then point in the direction of positive *z*. The rule for a left-handed system uses the left hand in a similar manner. It is also possible to define a left-handed coordinate system as the mirror image (reflection) of a right-handed one. Notice that one coordinate system cannot be transformed into the other by translating or rotating it. The difference between left-handed and right-handed coordinate systems becomes important when a three-dimensional object is projected on a two-dimensional screen. We assume that the screen is positioned at the *xy* plane with its origin (i.e., its bottom left corner) at the origin of the three-dimensional system. We also assume that the object to be projected is located on the positive side of the *z* axis and the viewer is located on the negative side, looking at the projection of the image on the screen. Figure 7.1b shows that in a left-handed three-dimensional coordinate system, the directions of the positive *x* and *y* axes on the

screen coincide with those of the three-dimensional *x* and *y* axes. In a right-handed system (Figure 7.1c), though, the two-dimensional *x* axis (on the screen) and the three-dimensional *x* axis point in opposite directions.

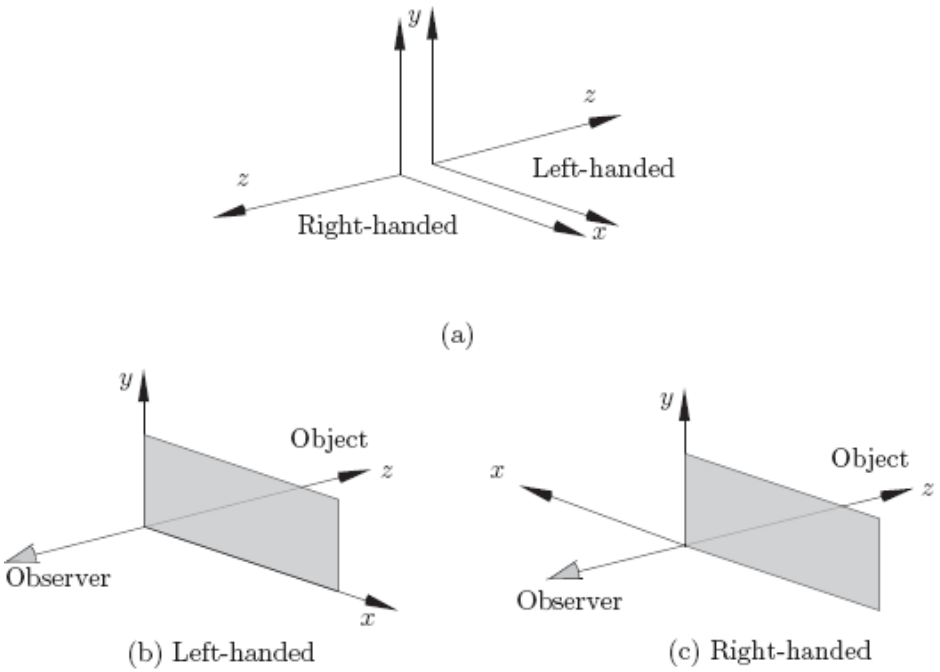


Figure 7.1 Three Dimensional Coordinate System

7.2 Three–Dimensional Transformations

The ability to represent or display a three-dimensional object is fundamental to the understanding of the shape of that object. Furthermore, the ability to rotate, translate, and project views of that object is also, in many cases, fundamental to the understanding of its shape. Manipulation, viewing, and construction of three-dimensional graphic images require the use of three-dimensional geometric and coordinate transformations. In geometric transformation, the coordinate system is fixed, and the desired transformation of the object is done with respect to the coordinate system. In coordinate transformation, the object is fixed and the desired transformation of the object is done on the coordinate system itself. These transformations are formed by composing the basic transformations of translation, scaling, and rotation. Each of these transformations can be represented as a matrix transformation. This permits more complex transformations to be built up by use of matrix multiplication or concatenation. We can construct the complex objects/pictures, by instant transformations. In order to represent all these transformations, we need to use homogeneous coordinates.

As with two – dimensional transformations, two complementary points of view are adopted: either the object is manipulated directly through the use of geometric transformation, or the object remains stationary and the viewer's coordinate system is changed by using coordinate transformations. In addition, the construction of complex objects and scene is facilitated by the use of instance transformations. The concepts and transformations introduced here are direct generalizations of these introduced in for two-dimensional transformations.

7.2.1 Geometric Transformations

With respect to some three- dimensional coordinate system, an object Obj is considered as a set of points.

Obj = { P (x,y,z)}

If the object moved to a new position, we can regard it as a new object Obj', all of whose coordinate points P' (x' , y', z') can be obtained from the original coordinate points P(x,y,z) of Obj through the application of a geometric transformation.

Translation

An Object is displaced a given distance and direction from its original position. The direction and displacement of the translation is prescribed by a vector

V = aI + bJ + cK

The new coordinates of a translated point can be calculated by using the transformation

$$T_v : \begin{cases} x' = x + a \\ y' = y + a \\ z' = z + a \end{cases}$$

(see Fig. 7.2). In order to represent this transformation as a matrix transformation, we need to use homogeneous coordinate. The required homogeneous matrix transformation can then be expressed as

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & a \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

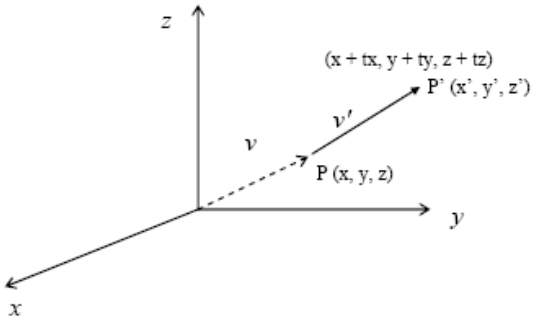


Figure 7.2

Scaling

The process of scaling changes the dimensions of an object. The scale factor s determines whether the scaling is a magnification, s> I , or a reduction , s< 1.

Scaling with respect to the origin, where the origin remains fixed, is offered by the transformation

$$S_{s_x, sy, s_z} : \begin{cases} x' = S_x . x \\ y' = S_y . y \\ z' = S_x . z \end{cases}$$

In matrix form this is

$$S_{s_x, sy, s_z} = \begin{pmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & S_z \end{pmatrix}$$

Rotation

Rotation in three dimensions is considerably more complex then rotation in two dimensions. In 2-D, a rotation is prescribed by an angle of rotation θ and a centre of rotation, say P.

In two dimensions, a rotation is prescribed by an angle of rotations require the prescription of an angle of rotation and an axis of rotation. The canonical rotations are defined when one of the positive x, y or z coordinate axes is chosen as the axis of rotation. Then the construction of the rotation transformation proceeds just like that of a rotation in two dimensions about the origin (see. Fig. 7.3).

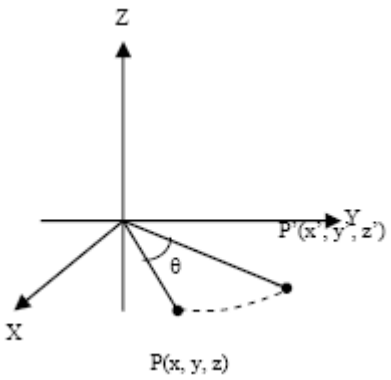


Figure 7.3

Rotation about the z Axis

$$R_{\theta.K}: \begin{cases} x' = x\cos\theta - y\sin\theta \\ y' = x\sin\theta + y\cos\theta \\ z' = z \end{cases}$$

Rotation about the y Axis

An analogous derivation leads to

$$R_{\theta.J}: \begin{cases} x' = x\cos\theta + z\sin\theta \\ y' = y \\ z' = y\sin\theta + z\cos\theta \end{cases}$$

Rotation about the x Axis

Similarly:

$$R_{\theta.I}: \begin{cases} x' = x \\ y' = y\cos\theta - z\sin\theta \\ z' = y\sin\theta + z\cos\theta \end{cases}$$

Note that the direction of a positive angle of rotation is chosen in accordance to the right-hand rule with respect to the axis of rotation.

The Corresponding matrix transformations are

$$R_{\theta,K} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$R_{\theta,J} = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix}$$

$$R_{\theta,I} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix}$$

The general use of rotation about an axis L can be built up from these canonical using matrix multiplication.

7.2.2 Rotation About an Arbitrary Axis in Space

The rotation about an arbitrary axis in space occurs in robotics animation and simulation. This is accomplished using the procedure of translations and simple rotation about the coordinate axes. (See figure 7.4) We know the technique for rotation about a coordinate axis. The procedural idea is to make the arbitrary rotation axis coincident with one of the coordinate axes.

Let use assume that an arbitrary axis in space passing through the point (x_0, y_0, z_0) with direction cosine (Cx, Cy, Cz). Rotation about this axis by some angle is accomplished using the following procedure i.e.

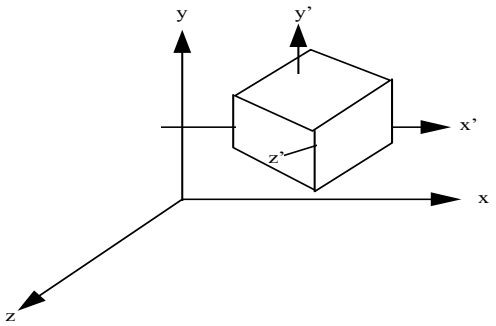


Figure 7.4

1. Translate the body so that the point (x_0, y_0, z_0) is reached at the origin of the coordinate system.

2. Do appropriate rotations to make the axis of rotation coincident with the z-axis (the selection of z-axis is arbitrary we can choose any axis).

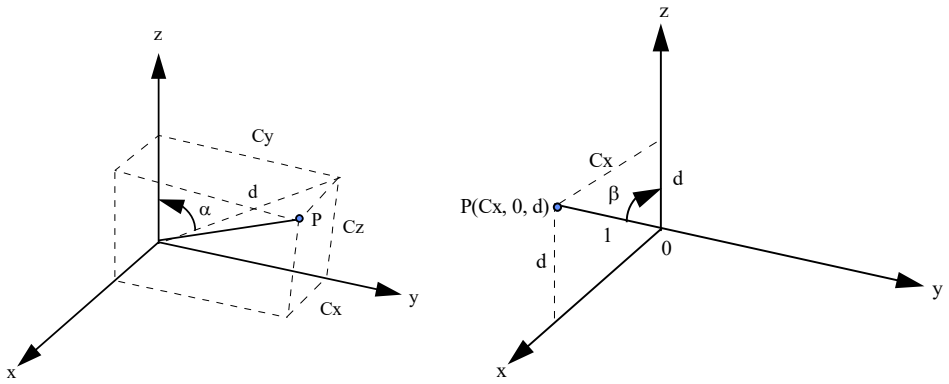
3. Rotate the body by the angle □about the z-axis.

4. Do the inverse of the combined rotation transformation.

5. Now perform the inverse of the translation.

Now our aim to make the arbitrary rotation axis coincident with the z-axis (It is an arbitrary choice we can choose any axis), For this first rotate about the x-axis and them about the y-axis.

To find the value of rotation angle about x-axis used to place the arbitrary axis in the xz-plane, now first project the unit vector along the axis onto the yz-plane as shown in fig 7.5 (a).



(a) **(b)**

Figure 7.5 (a) Rotation about x-axis), (b) Rotation about y-axis

The projected component of the y and z vectors are C_y and C_z and the direction cosine of the unit vector along the arbitrary axis. from the fig (a) we note that -

$$d = \sqrt{C_y^2 + C_z^2} \qquad \text{- - - (1)}$$

and

$$\cos\alpha = \frac{C_z}{d} \qquad \sin\alpha = \frac{C_y}{d} \qquad \text{- - - (2)}$$

After completion of rotation about the x-axis into the xz-plane the z-component of the unit vector is d and the x-component is Cx and the direction cosine in the x-direction i.e. shown in fig 7.5(b). We know the length of the unit vector is 1 (one). The rotation angle b about the y-axis required to make the arbitrary axis coincident with z-axis is -

$$\cos\beta = d \qquad \text{and} \qquad \sin\beta = C_x \qquad \text{- - - (3)}$$

Now the total transformation or composite transformation is then.

$$[T] = [Tr]\,[Rx]\,[Ry]\,[Rd]\,[Ry]^{-1}\,[Rx]^{-1}\,[Tr]^{-1} \qquad \text{- - - (4)}$$

where

$$[Tr] = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x_0 & -y_0 & -z_0 & 1 \end{pmatrix} \qquad \text{- - - (5)}$$

Tr is the translation matrix

Rx is the rotation matrix about x-axis

$$[Rx] = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & Cx/d & Cy/d & 0 \\ 0 & -Cy/d & Cz/d & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad -(7)$$

Ry is the rotation matrix about y-axis is -

$$[Ry] = \begin{pmatrix} \cos(-\beta) & 0 & -\sin(-\beta) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(-\beta) & 0 & \cos(-\beta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} d & 0 & Cx & 0 \\ 0 & 1 & 0 & 0 \\ -Cx & 0 & d & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (7)$$

and

$$Rd = \begin{pmatrix} \cos d & \sin d & 0 & 0 \\ -\sin d & \cos d & 0 & 0 \end{pmatrix} \qquad \text{- - - (8)}$$

0	0	1	0
0	0	0	1

Rd is the rotation about the arbitrary axis is given by z-axis.

Now we take a second point on the axis i.e. $(x_1\ y_1\ z_1)$ to find the direction cosine i.e.

$$[c_x \quad c_y \quad c_z] = \frac{[(x_1 - x_0) \quad (y_1 - y_0) \quad (z_1 - z_0)]}{[(x_1 - x_0)^2 + (y_1 - y_0)^2 + (z_1 - z_0)^2]^{1/2}}$$

7.2.3 Coordinate Transformations

We can also achieve the effects of translation, scaling, and rotation by moving the observer who views the object and by keeping the object stationary. This type of transformation is called a coordinate transformation. We first attach a coordinate system to the observer and then move the observer and the attached coordinate system. Next, we recalculate the coordinates of the observed object with respect to this new observer coordinate system. The new coordinate values will be exactly the same as if the observer had remained stationary and the object had moved, corresponding to a geometric transformation (see Fig. 7.6).

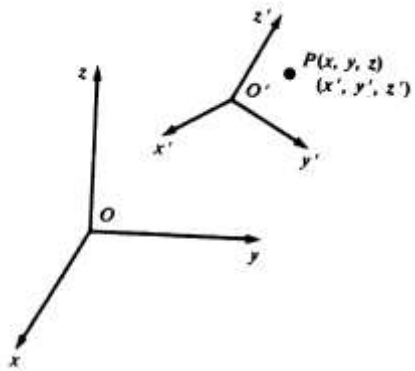


Figure 7.6

If the displacement of the observer coordinate system to a new position is prescribed by a vector V = a1 + bj + cK , a point P9x,y,z) in the original coordinate system has coordinate P(x',y',z') in the new coordinate system, and

$$\bar{T}_v : \begin{cases} x' = x - a \\ y' = y - b \\ z' = z - c \end{cases}$$

The derivation of this transformation is completely analogous to that of the two-dimensional transformation.

Similar derivations hold for coordinate scaling and coordinate rotation transformations.

As in the two-dimensional case , we summarize the relationships between the matrix forms of the coordinate transformations and the geometric transformations:

	Coordinate Transformations	Geometric Transformations
Translation	$\bar{T}_v$	T_{-v}
Rotation	$\bar{R}_\theta$	$R_{-\square}$
Scaling	$\bar{S}_{s_x, s_x, s_z}$	$S_{1/sx,1/sy,1/sz}$

Inverse geometric and coordinate transformations are constructed by performing the reverse operation. Thus, for coordinate transformations (and similarly for geometric transformation):

$$\bar{T}_v^{-1} = \bar{T}_{-v} \qquad \bar{R}_\theta^{-1} = \bar{R}_{-\theta} \qquad \bar{S}_{s_x, s_y, sz}^{-1} = \bar{S}_{1/x_x, 1/s_y, 1/s_z}$$

7.2.4 Composite Transformations

More complex geometric and coordinate transformations are formed through the process of composition of functions. For matrix function, however, the process of composition is equivalent to matrix multiplication or concatenation. The following more complex transformations can be constructed:

1. $A_{V,N}$ = Aligning a vector V with a vector N.
2. $R_{\theta,L}$ = rotation about an axis is prescribed by giving a direction vector V and a point P through which the axis passes.
3. $S_{Sx,Sy,Sz,P}$ = scaling with respect to an arbitrary point P.

In order to build these more complex transformations through matrix connection, we must be able to multiply translation matrices with rotation and sealing matrices. This necessitates the use of homogeneous coordinates and 4 x 4 matrices. This necessitates the use of homogeneous coordinates and 4 x 4 matrices. The standard 3 x 3 matrices of rotation and scaling can be represented as 4 x 4 homogeneous matrices by adjoining an extra row and column as follows:

$$\begin{pmatrix} a & b & c & 0 \\ d & e & f & 0 \\ g & h & i & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

These transformations are then applied to points P (x, y., z,) having the homogeneous form:

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

7.3 Projections

Once world-coordinate descriptions of the objects in a scene are converted to viewing coordinates, we can project the three-dimensional objects onto the two-dimensional view plane. There are two basic projection methods. In a parallel projection, coordinate positions are transformed to the view plane along parallel lines, as shown in the example of Fig. 7.7. For a perspective projection (Fig. 7.8), object positions are transformed to the view plane along lines that converge to a point called the projection reference point (or center of projection). The projected view of an object is determined by calculating the intersection of the projection lines with the view plane.

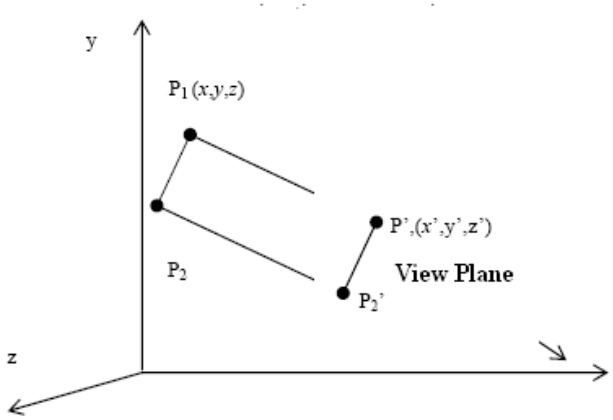


Figure 7.7: Parallel projection of an object to the view plane

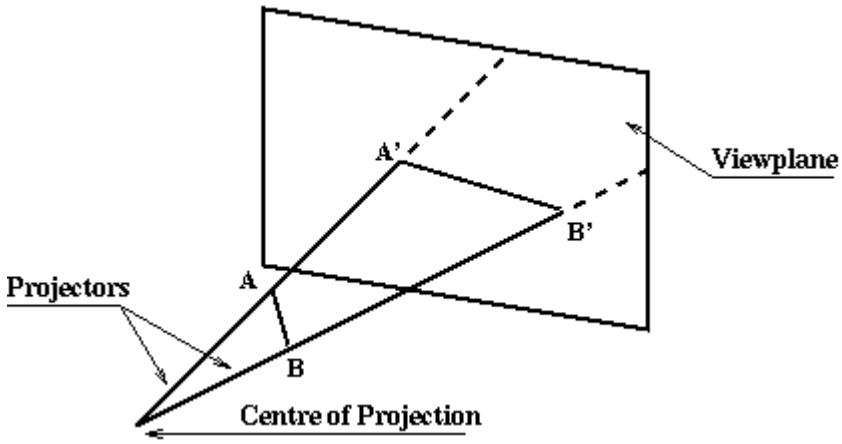


Figure 7.8: Perspective projection of an object to the view plane

A parallel projection preserves relative proportions of objects, and this is the method used in drafting to produce scale drawings of three-dimensional objects. Accurate views of the various sides of an object are obtained with a parallel projection, but this does not give us a realistic representation of the appearance of a three-dimensional object. A perspective projection, on the other hand, produces realistic views but does not preserve relative proportions. Projections of distant objects are smaller than the projections of objects of the same size that are closer to the projection plane.

7.3.1 Parallel Projections

1. **Orthographic**. The projection of a point by a projector parallel to the ith co-ordinate axis onto a viewplane containing the other two axes is obtained by setting the ith co-ordinate to zero.

Hence to project in the direction of the z-axis onto the xy plane can be carried out by:

scale(1.0.1.0,0.0)

2.**Axonometric**. The required transformation is produced in two stages, firstly the direction of projection is rotated until it aligns with one of the co-ordinate axes and then an orthographic projection along that axis is carried out.

For example in the isometric case the direction of projection must be symmetric with respect to the three co-ordinate directions to allow equal foreshortening on each axis. This is obviously achieved by using a direction of projection *(1,1,1).*

3.**Oblique**. It is usual to take the viewplane parallel to a face of the object. This face (and these parallel to it) will remain undistorted(figure 7.9).

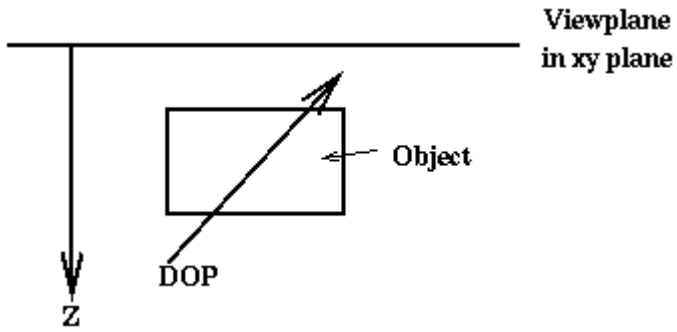


Figure 7.9

Assume that the viewplane is the xy plane and that in the required image the foreshortening factor in the z direction is *l* and lines parallel to the z-axis make an angle of *alpha* with the x-axis. Thus a cube of side *l* would appear in the image as follows(figure 7.10):

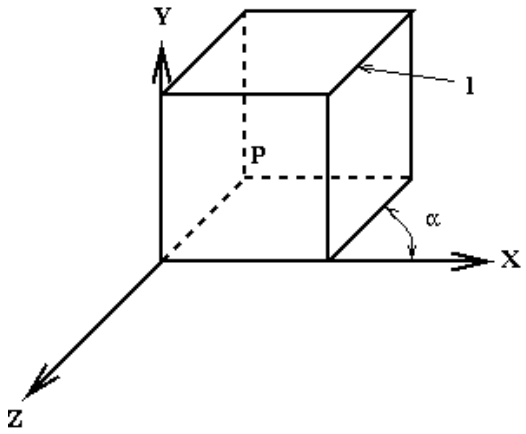


Figure 7.10

In the image the point P is *(lcos(alpha),lsin(alpha))*, but this corresponds to the point *(0,0,-1)* in three-dimensions hence the required transformation in matrix form using homogeneous co-ordinates is:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -l\cos\alpha & 0 \\ 0 & 1 & -l\sin\alpha & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

which is a **shear transformation**. If *l = 1* then it is called a Cavalier projection and if *l = ½* then it is called a Cabinet projection.

7.3.2 Perspective Projections

In the following it is assumed that the Centre of Projection is at the origin and that the viewplane is normal to the z-axis at a distance d from the origin(figure 7.11). This is the situation that holds after view orientation and view mapping.

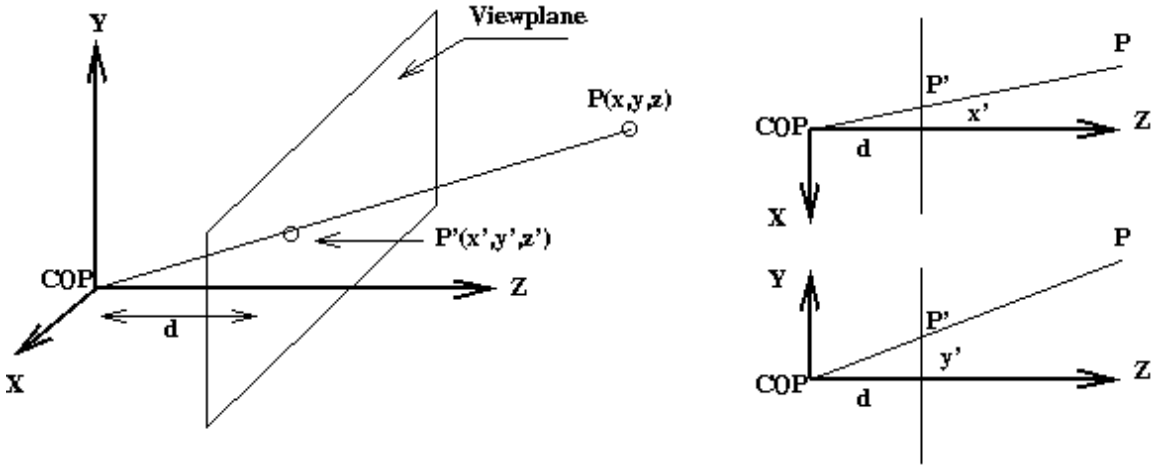


Figure 7.11

Using similar triangles gives:

$$x' = \frac{x}{z/d} \quad , \quad y' = \frac{y}{z/d}$$

Using homogeneous co-ordinates this can be expressed as follows:

$$[X\ Y\ Z\ W]^T = Mper[x\ y\ z\ 1]^T$$

where

$$Mper = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix}$$

that is:

$$[X\ Y\ Z\ W]^T = [x\ y\ z\ \ \frac{z}{d}]^T$$

hence dividing by the homogeneous term W gives:

$$[x'\ y'\ z'\ 1]^T = [\frac{x}{z/d}\ \ \frac{y}{z/d}\ \ d\ \ 1]^T$$

which gives the projected point as previously and also gives $z' = d$ as required.

From

$$x' = \frac{x}{z/d} \quad y' = \frac{y}{z/d} \quad z' = d$$

it is seen that if *x* and *y* remain constant *x'-> 0* and *y'->0* while *z->infinity,* i.e. lines parallel to the z-axis converge to the *vanishing point (0,0)* in the viewplane. If the viewplane cuts two axes then two vanishing points result while if it cuts three axes then three vanishing points will result.

7.4 Three – Dimensional Viewing and Clipping

An important step in photography is to position and aim the camera at the scene in order to compose a picture. This parallel the specification of 3D viewing parameters in computer graphics that prescribe the projector (the center of projection for perspective projection or the direction of projection for parallel projection) along with the position and orientation of the projection / view plane.

In addition, a view volume defines the spatial extent that is visible through a rectangular window in the view plane. The bounding surfaces of this view volume is used to tailor / clip the objects that have been placed in the scene via modeling

transformations prior to viewing . The clipped objects are then projected into the window area, resulting in a specific view of the 3D scene that can be further mapped to the view-port in the NDCS.

In this chapter we are concerned with the specification of 3D viewing parameters, including a viewing coordinate system for defining the view plane window, and the formation of the corresponding view volume. We also discuss 3D clipping strategies and algorithms. We then summarize the three-dimensional viewing process. Finally, we examine the operational the organization of a typical 3D graphics pipeline.

7.4.1 Three – Dimensional Viewing

Three – dimensional viewing of objects requires the specification of a projection plane (called the view plane), a center of projection, and a view volume in world coordinates.

Specifying the View Plane

We specify the view plane by prescribing (1) a reference point R_0 (x_0, y_0, z_0) in world coordinates and (2) a unit normal vector $N = n_1I + n_2J + n_3K$, $|N|= 1$, to the view plane. From this information we can construct the projections used in presenting the required view with respect to the given viewpoint or direction of projection.

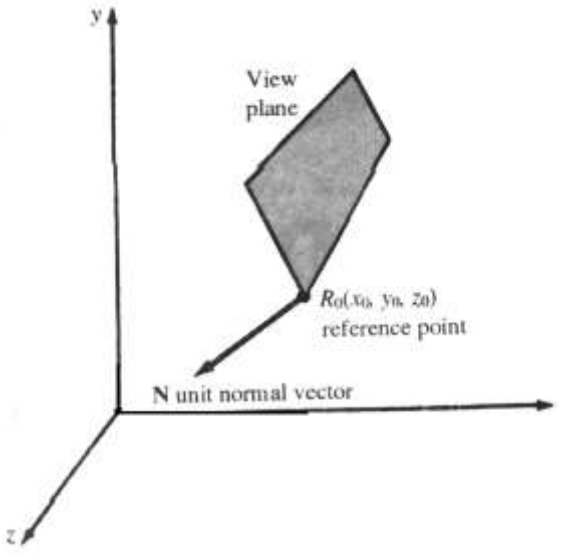


Figure: 7.12

View Plane Coordinates

The view plane coordinates system or viewing coordinate system can be specified as follows; (1) let the reference point R_0 (x_0, y_0, z_0) be the origin of the coordinates system and (2) determine the coordinate axes To do this we first choose a reference vector U called the up vector. A unit vector J_q define the direction of the positive q axis for the view plane coordinate system . To calculate J_q we proceed as follows: with N being the view plane unit normal vector , let $U_q = U - (N - U) N$ Then

$$J_q = \frac{U_q}{|U_q|}$$

Is the unit vector that defines the direction of the positive q axis (see Fig. 7.13)

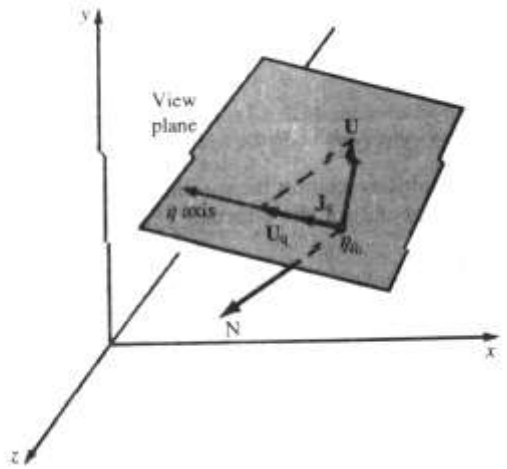


Figure 7.13

Finally, the direction vector I_p of the positive p axis is chosen so that it is perpendicular to J_q and by convention, so that the triad I_p,J_q, and N form a left – handed coordinate system . That is :

$$I_p = \frac{N \times J_p}{|N \times J_p|}$$

This coordinate system is called the view plane coordinate system or viewing coordinate system. A left-handed system is traditionally chosen so that, if one thinks of the view plane as the face of a display device, then with the p and q coordinate axes superimposed on the display device, the normal vector N will point away from an

observer facing the display. Thus the direction of increasing distance away from the observer is measured along N [see Fig. 7.14(a)].

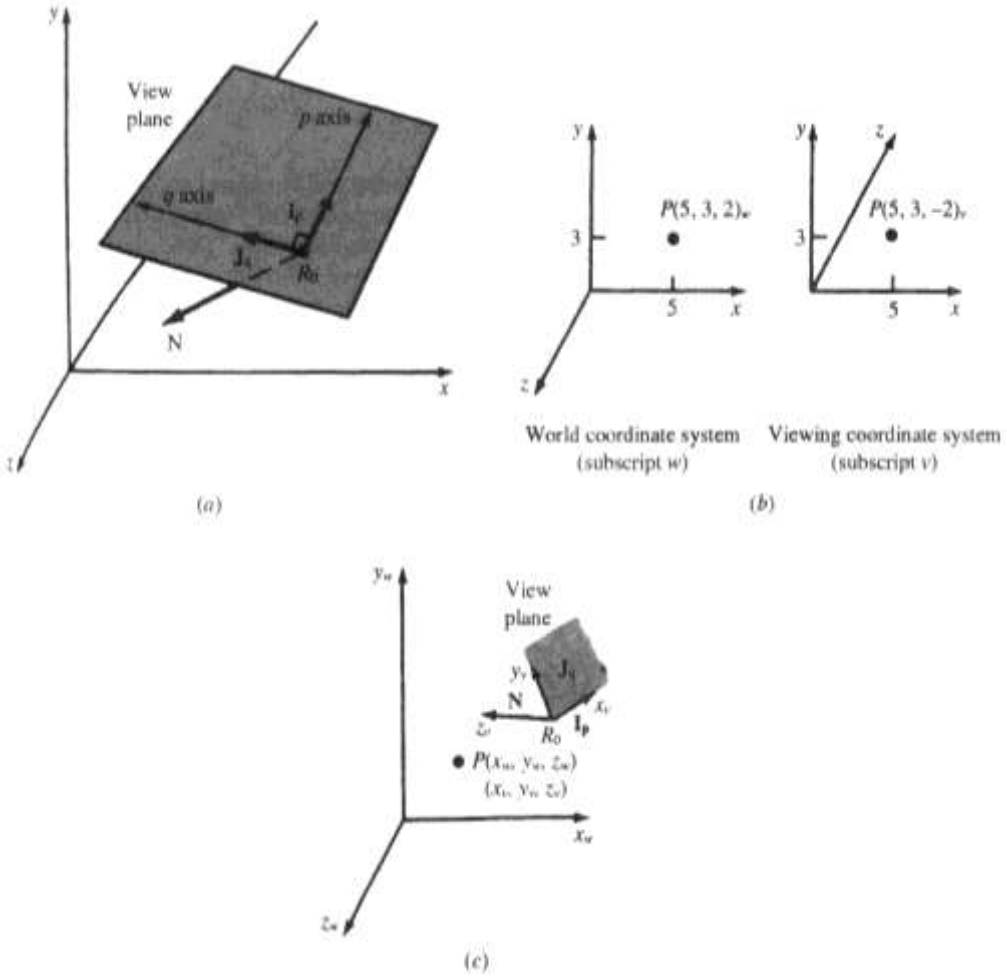


Figure 7.14

Specifying the View Volume

The view volume bounds a region in world coordinate space that will be clipped and projected onto the view plane. To define a view volume that projects onto a specified rectangular window defined in the view plane, we use view plane coordinates $(p, q)_v$ to locate points on the view plane. Then a rectangular view plane window is defined by prescribing the coordinates of the lower left-hand corner $L(p_{min}, q_{min})_v$ and upper right hand corner $R(p_{max}, q_{max})_v$ (see Fig. 7.15). We can use the vectors I_p and J_q to find the equivalent world coordinates of L and R.

For a perspective , view the view volume, corresponding , to the given window , is a semi-infinite pyramid, with apex at the viewpoint (Fig. 7.16). For views created using

parallel projections (Fig. 7.17) the view volume is an infinite parallelepiped with sides parallel to the direction of projection.

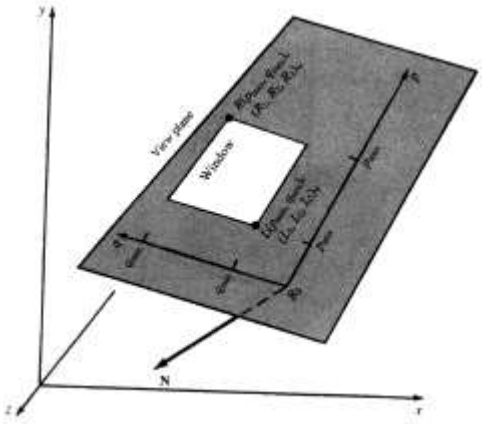


Figure 7.15

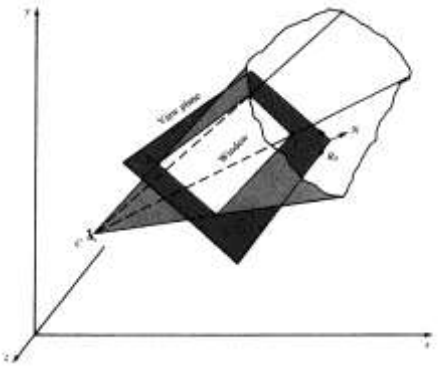


Figure 7.16

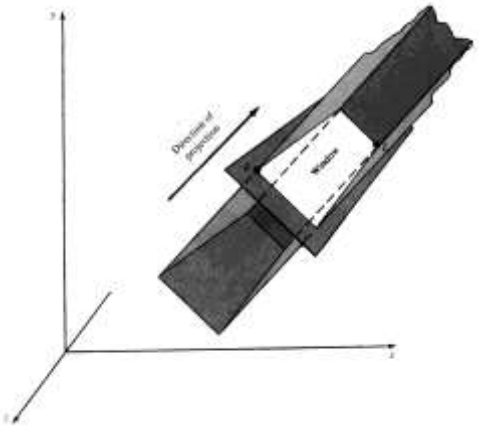


Figure 7.17

Clipping against a Finite View Volume

The view volumes created above are infinite in extent. In practice, we prefer to use a finite volume to limit the number of points to be projected. In addition, for perspective views, very distant objects from the view plane, when projected, appear as indistinguisliable spots, while objects very close to the center of projection appear to have disjointed structure. This is another reason for using a finite view volume.

A finite volume is deliminated by using front (near) and back (far) clipping planes parallel to the view plane. These planes are specified by giving the front distance f and back distance b relative to the view plane reference point Ro and measured along the normal vector N. The signed distance b and f can be positive or negative (Figs. 7.18 and 7.19).

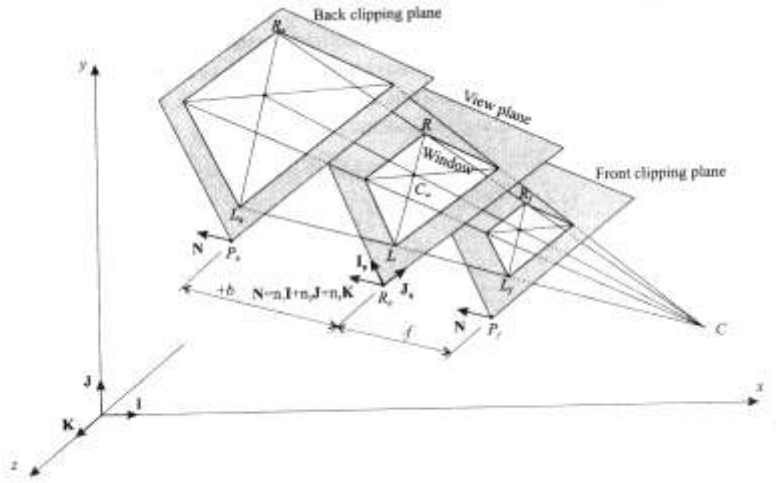


Figure 7.18

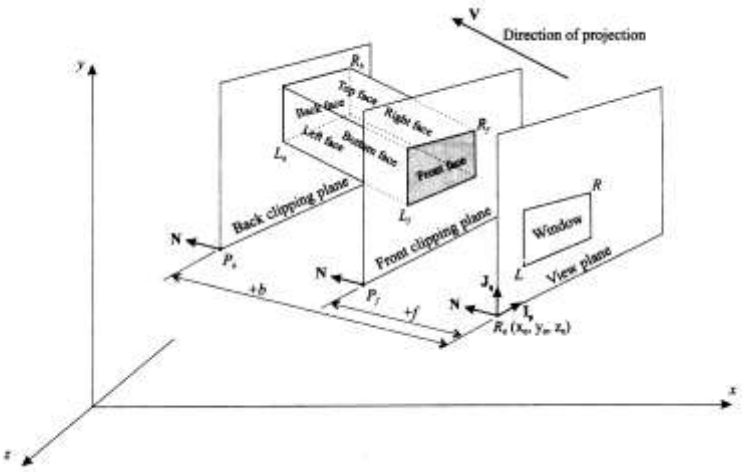


Figure 7.19

7.4.2 Three Dimensional-Clipping

Two differing strategies have been devised to deal with the extraordinary computational effort required for three-dimensional clipping:

1. Direct clipping: In this method, as the name suggests, clipping is done directly against the view volume.
2. Canonical clipping: In this method, normalizing transformations are applied which transform the original view volume into a so-called canonical view volume. Clipping is then performed against the canonical view volume.

The canonical view volume for parallel projection is the unit cube whose faces are defined by the planes x = 0, x = 1, y = 1, z = 0, and z = 1. The corresponding normalization transformation N_{par} is constructed (See Figure 7.20).

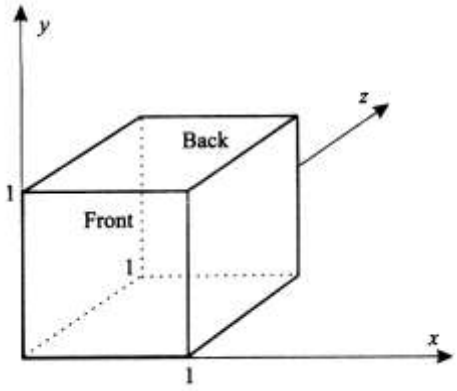


Figure 7.20

The canonical view volume for perspective projection is the truncated pyramid whose faces are defined by the planes x = z, x = -z, y = z , y = -z, z = z_f, and z = 1 (where z_f is to be calculated) (Fig. 7.21).

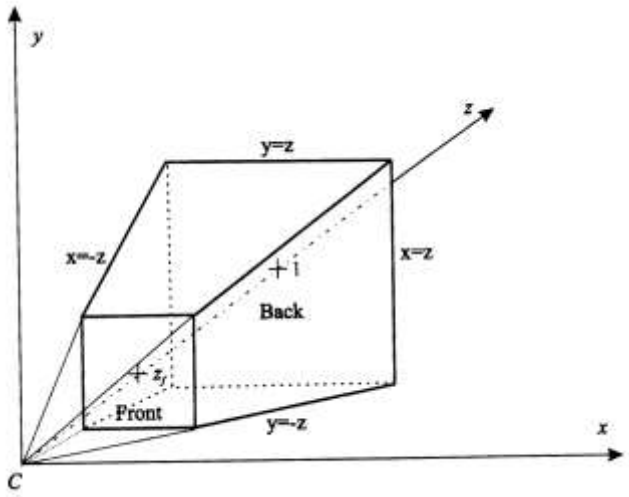


Figure 7.21

The basis of the canonical clipping strategy is the fact that the computations involved such operations as finding the intersections of a line segment with the planes forming the faces of the canonical view volume are minimal. This is balanced by the overhead involved in transforming points many of which will be subsequently clipped.

For perspective views, additional clipping may be required to avoid the perspective anomalies produced by projecting objects that are behind the viewpoint.

Clipping Algorithms

Three – dimensional clipping algorithms are often direct adaptations of their two-dimensional counter parts. The modifications necessary arise from the fact that we are now clipping against the six. Faces of the view volume, which are planes, as opposed to the four edges of the two – dimensional window, which are lines

1. Finding the intersection of a line and a plane.
2. Assigning region codes to the endpoints of line segments for the Cohen-Sutherland algorithm.
3. Deciding when a point is to the right (also said to be outside) or to the left (inside) of a plane for the Sutherland – Hodgman algorithm.
4. Determining the inequalities for points inside the view volume.

7.5 Wireframe Models

A wireframe model consists of edges, vertices, and polygons. Here vertices are connected by edges, and polygons are sequences of vertices or edges. The edges may be curved or straight line segments. In the latter case, the wireframe model is called a polygonal net or polygonal mesh (Fig. 7.22).

Figure 7.22

7.5.1 Representing a Polygonal Net Model

There are several different ways of representing a polygonal net model.

1. Explicit vertex list V = $\{P_0, P_1, P_2, \ldots..P_N)$. The points $P_i(x_i, y_i, z_i)$ are the vertices of the polygonal net, stored in the order in which they would be encountered by traveling around the model . Although this form of representation is useful for single polygons, it is quite inefficient for a complete polygonal net, in that shared vertices are repeated several times. In addition when displaying the model by drawing the edges, shared edges are drawn several times.

2. Polygon listing . In this form of representation , each vertex is stored exactly once in a vertex list V = (P_0 ….P_N), and each polygon is defined by pointing or indexing into this vertex list. Again , shared edges are drawn several times in displaying the model.

3. Explicit edge listing . In this form of representation, we keep a vertex list in which each vertex is stored exactly once and an edge list in which each edge is stored exactly once. Each edge in the edge list points to the two vertices in the vertex list which define that edge. A polygon is now represented as a list of pointers or indices into the edge list. Additional information, such as those polygon sharing a given edge, can also be stored in the edge list. Explicit edge listing can be used to represent the more general wireframe model. The wireframe model is displayed by drawing all the edges, and each edge is drawn only once.

7.5.2 Polyhedron

A polyhedron is a closed polygonal net (i.e. one which encloses a define volume) in which each polygon is planar. The polygon are called the faces of the polyhedron . In modeling, polyhedrons are quite often treated as solid (i.e. block) objects as opposed to wireframes or two –dimensional surfaces.

7.5.3 Advantages and Disadvantages of Wireframe Models

Wireframe models are used in engineering applications. They are easy to construct and , if they are compound of straight lines, easy to clip and manipulate through the use of geometric and coordinate transformations. However, for building realistic models, especially of highly curved objects, we must use a very large number of polygons to achieve the illusions of roundness and smoothness.

7.6 Bezier Curves and Surfaces

This spline approximation method was developed by the French engineer Pierre Bezier for use in the design of Renault automobile bodies. Bezier splines have a number of properties that make them highly useful and convenient for curve and surface design. They are also easy to implement. For these reasons, Bezier splines are widely available in various cad systems, in general graphics packages (such as GL on

silicon graphics systems), and in assorted drawing and painting pack-ages (such as Aldus superPaint and cricket draw).

7.6.1 Bezier Curves

Bezier curves are used in computer graphics to produce curves which appear reasonably smooth at all scales. This spline approximation method was developed by French engineer Pierre Bezier for automobile body design. Bezier spline was designed in such a manner that they are very useful and convenient for curve and surface design, and are easy to implement Curves are trajectories of moving points. We will specify them as functions assigning a location of that moving point (in 2D or 3D) to a parameter t, i.e., parametric curves.

Curves are useful in geometric modeling and they should have a shape which has a clear and intuitive relation to the path of the sequence of control points. One family of curves satisfying this requirement are Bezier curve.

The Bezier curve require only two end points and other points that control the endpoint tangent vector.

Bezier curve is defined by a sequence of N + 1 control points, P_0, P_1,. . . , P_n. We defined the Bezier curve using the algorithm (invented by DeCasteljeau), based on recursive splitting of the intervals joining the consecutive control points.

A purely geometric construction for Bezier splines which does not rely on any polynomial formulation, and is extremely easy to understand. The DeCasteljeau method is an algorithm which performs repeated bi-linear interpolation to compute splines of any order.

De Casteljeau algorithm: The control points P0, P1, P2 and P3are joined with line segments called ‘control polygon’, even though they are not really a polygon but rather a polygonal curve.

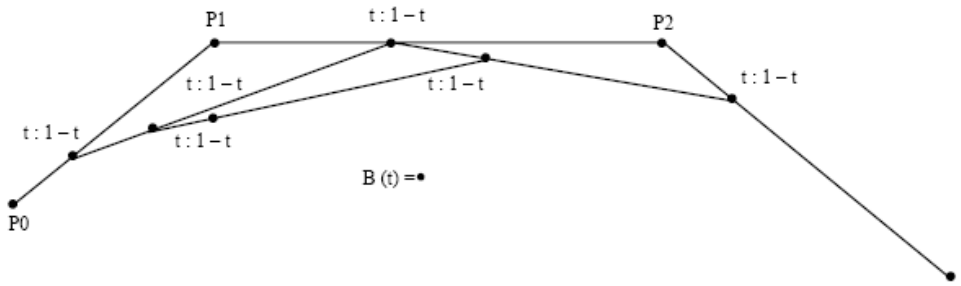


Figure 23

Each of them is then divided in the same ratio t : 1- t, giving rise to the another points. Again, each consecutive two are joined with line segments, which are subdivided and so on, until only one point is left. This is the location of our moving point at time t. The trajectory of that point for times between 0 and 1 is the Bezier curve.

A simple method for constructing a smooth curve that followed a control polygon p with m-1 vertices for small value of m, the Bezier techniques work well. However, as m grows large (m>20) Bezier curves exhibit some undesirable properties.

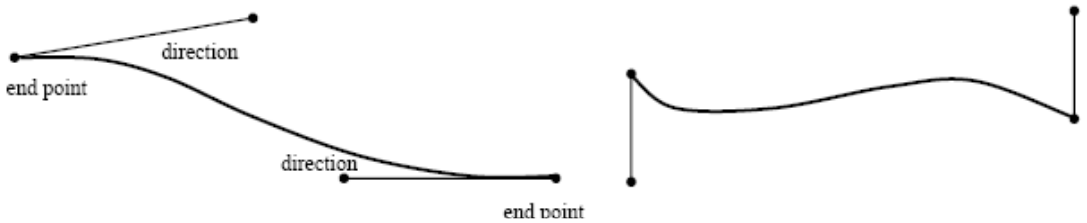


Figure 24(a) Bezier curve defined by its endpoint vector (b) All sort of curves can be specified with different direction vectors at end points

Figure 24(c) Reflex curves appear when you set the vectors in different directions

In general, a Bezier curve section can be fitted to any number of control points. The number of control points to be approximated and their relative positions determine the degree of the Bezier polynomial. As with the interpolation splines, a Bezier curve can be specified with boundary conditions, with a characterizing matrix, or with blending function. For general Bezier curves, the blending-function specification is the most convenient.

7.6.2 Bezier Surfaces

Two sets of orthogonal Bezier curves can be used to design an object surface by specifying by an input mesh of control points. The parametric vector function for the Bezier surface is formed as the Cartesian product of Bezier blending functions:

$$P(u,v) = \sum_{j=o}^{m} \sum_{k=o}^{n} P_{j,k} BEZ_{j,m} BEZj,m(v) BEZk,n(u)$$

With $P_{j,k}$ specifying the location of the (m + 1) by (n + 1) control points.

Figure 7.25 illustrates two Bezier surface plots. The control points are connected by dashed lines, and the solid lines show curves of constant a and constant v. Each curve of constant a is plotted by varying v over the interval from 0 to 1, with u fixed at one of the values in this unit interval. Curves of constant v are plotted similarly.

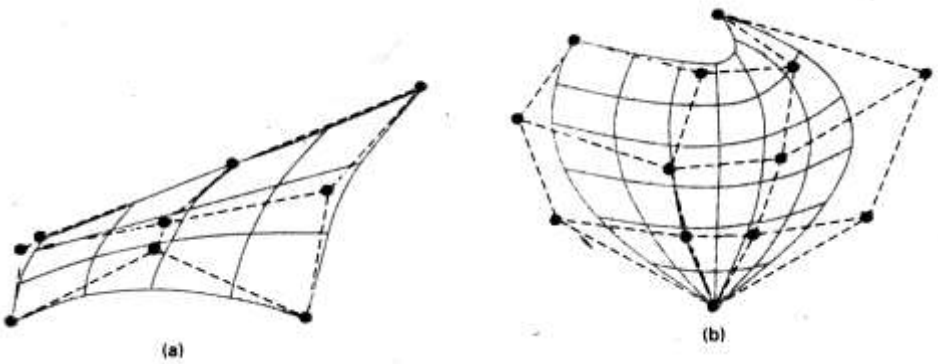


Figure 7.25: Bezier surfaces constructed for (a) m = 3, n = 3, and (b) m = 4, n = 4. Dashed lines connect the control points

Bezier surfaces have the same properties as Bezier curves, and they provide a convenient method for interactive design applications. For each surface patch, we can select a mesh of control points in the xy "ground" plane, then we choose elevations above the ground plane for the z-coordinate values of the control points. Patches can then be pieced together using the boundary constraints.

Figure 7.26 illustrates a surface formed with two Bezier sections. As with curves, a smooth transition from one section to the other is assured by establishing both zero-order and first-order continuity at the boundary line. Zero-order continuity is obtained by matching control points at the boundary. First-order continuity is obtained by choosing control points along a straight line across the boundary and by maintaining a constant ratio of collinear line segments for each set of specified control points across section boundaries.

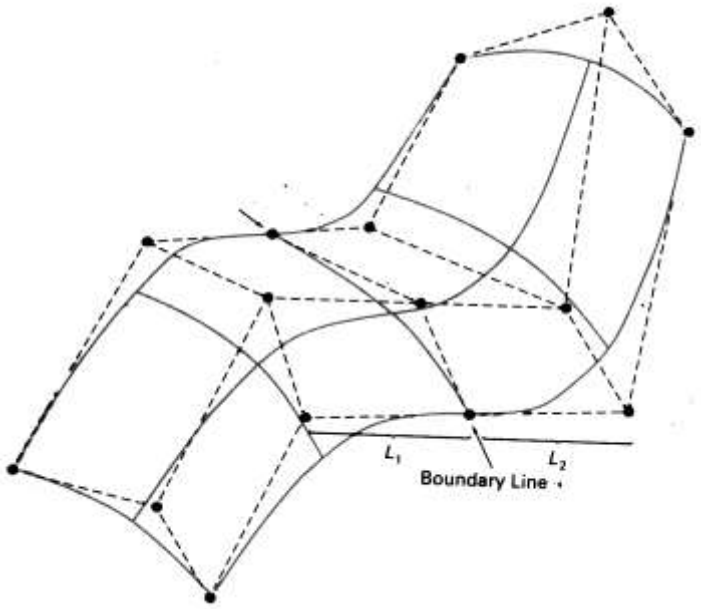


Figure 7.26: A composite Bezier surface constructed with two Bezier sections, joined at the indicated boundary line. The dashed lines connect specified control points. First-order continuity is established by making the ratio of length L_1 to length L_2 constant for each collinear Line of control points across the boundary between the surface sections.

7.7 Concept of Hidden Line and Surfaces

7.7.1 Hidden Surfaces

Opaque objects that are closer to the eye and in the line of sight of other objects will block those objects those objects from view. In fact, some surfaces of these opaque objects themselves are not

Visible because they are eclipsed by the objects' visible parts. The surfaces that are blocked or hidden from view must be "removed" in order to construct a realistic view of the 3D scene The identification and removal of these surfaces is called the hidden – surface problem. The solution involves the determination of the closest visible surface along each projection line.

There are many different hidden –surface algorithms. Each can be characterized as either an image-space method, in which the pixel grid is used to guide the computational activities that determine visibility at the pixel level, or an object-space method, in which surface visibility is determined using continuous models in the object space (or its transformation) without involving pixel based operations.

Notice that the hidden – surface problem has ties to the calculation of shadows. If we place a light source, such as a bulb, at the viewpoint, all surfaces that are visible from the viewpoint are lit directly by the light source and all surfaces that are hidden from the viewpoint are in the shadow of some opaque objects blocking the light.

Depth Comparisons

We assume that all coordinates (x,y,z,) are described in the normalized viewing coordinates system.

Any hidden-surface algorithm must determine which edges and surfaces are visible either from the center of projection for perspective projections or along the direction of projection for parallel projections.

The question of visibility reduces to this : given two points P_1 (x_1, y_1, z_1) and P_2 (x_2, y_2, z_2), does either point obscure the other? This is answered in two steps:

1 Are P_1 and P_2 on the same projection line?

2 If not, neither point obscures the other. If so, a depth comparison tells us which point is in front of the other.

For an orthographic parallel projection onto the xy plane, P_1 and P_2 are on the same projector if $x_1 = x_2$ and $y_1 = y_2$. In this case, depth comparison reduces to comparing z_1 and z_2. If $z_1 < z_2$, then P_1 obscures P_2 [see Fig. 7.27(a)].

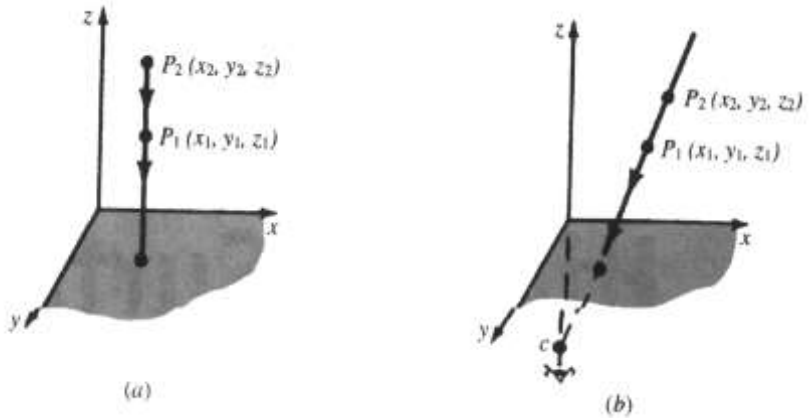


Figure 7.27

For a perspective projection [see. Fig. 7.27 (b)], the calculations are more complex. However, this complication can be avoided by transforming all-three dimensional

objects so that parallel projection of the transformed object is equivalent to a perspective projection of the original object (see Fig. 7.28). This is done with the use of the perspective to parallel transform Tp.

If the original object lies in the normalized perspective view volume, the normalized perspective to parallel transform

$$NT_p = \begin{pmatrix} \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{1-z_f} & \frac{-z}{1-z_f} \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

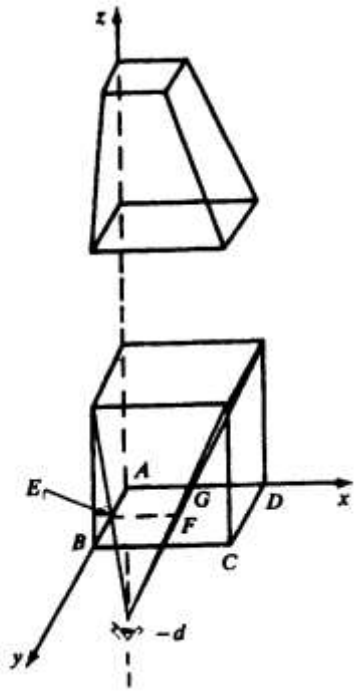


Figure 7.28

(where z_f is the location of the front clipping plane of the normalized perspective view volume) transforms the normalized perspective view volume into the unit cube bounded by $0 \leq x \leq 1$, $0 \leq y \leq 1$, $0 \leq z \leq 1$. We call this cube the normalized display space. A critical fact is that the normalized perspective to parallel transform preserves lines, planes, and depth relationships.

If our display device has display coordinates H x V, application of the scaling matrix

$$S_{H,V,1} = \begin{pmatrix} H & 0 & 0 & 0 \\ 0 & V & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

transforms the normalized display space $0 \leq x \leq 1$, $0 \leq y \leq 1$, $0 \leq z \leq 1$ onto the region $0 \leq x \leq H$, $0 \leq y \leq V$, $0 \leq z \leq 1$. We call this region the display space. The display transform DT_p.

$$DT_p = S_{H,V,1} \,.\, NT_p$$

transforms the normalized perspective view volume onto the display space.

Clipping must be done against the normalized perspective view volume prior to applying the transform NT_p. An alternative to this is to combine NT_p with normalizing transformation N_{per}, forming the single transformation $NT'_p = NT_p.N_{per}$. Then clipping is done in homogeneous coordinate space.

We now describe several algorithms for removing hidden surfaces from scenes containing objects defined with planar (i.e., flat), polygonal faces. We assumed that the displays transform DT_p has been applied (if a perspective projection is being used), so that we always deal with parallel projections in display space.

7.7.2 Hidden Line/ Surface Elimination Methods

Although there are special-purpose hidden –line algorithms, each of the above algorithms can be modified to eliminate hidden lines or edges. This is especially useful for wire frame polygonal models where the polygons are unfilled. The idea is to use a color rule, which fills all the polygons with the background color – say the edges and lines a different color – say white. The use of a hidden surface algorithm now becomes a hidden –line algorithm.

Z – Buffer Algorithm

We say that a point in display space is " seen" from pixel (x,y) if the projection of the point is scan converted to this pixel. The Z-buffer algorithm essentially keeps track of the smaller z coordinate (also called the depth value) of those point which are seen from pixel (x,y). These Z values are stored in what is called the Z buffer.

Let Z_{buf} (x, y) denote the current depth value that is stored in the Z buffer at pixel (x, y). We work with the (already) projected polygons P of the scene to be rendered.

The Z-buffer algorithm consists of the following steps.

1. Initialize the screen to a background color. Initialize the Z buffer to the depth of the back clipping plane. That is, set

Z_{buf} (x, y) = $Z_{back,}$ for every pixel (x, y)

2. Scan-convert each (projected) polygon P in the scene and during this scan–conversion process, for each pixel (x, y) that lies inside the polygon.

   a. Calculate Z (x,y) the depth of the polygon at pixel (x, y).

   b. If Z (x,y) < Z_{buf} (x, y), set Z_{buf}(x,y) = Z(x,y), and set the pixel value at (x, y) to the color of the polygon P at (x,y). In Fig. 7.40, points P_1 and P_2 are both scan-converted to pixel (x, y); however, since $z_1 < z_2$, P1 will obscure P_2 and the P_1 z value z_1, will be stored in the Z buffer.

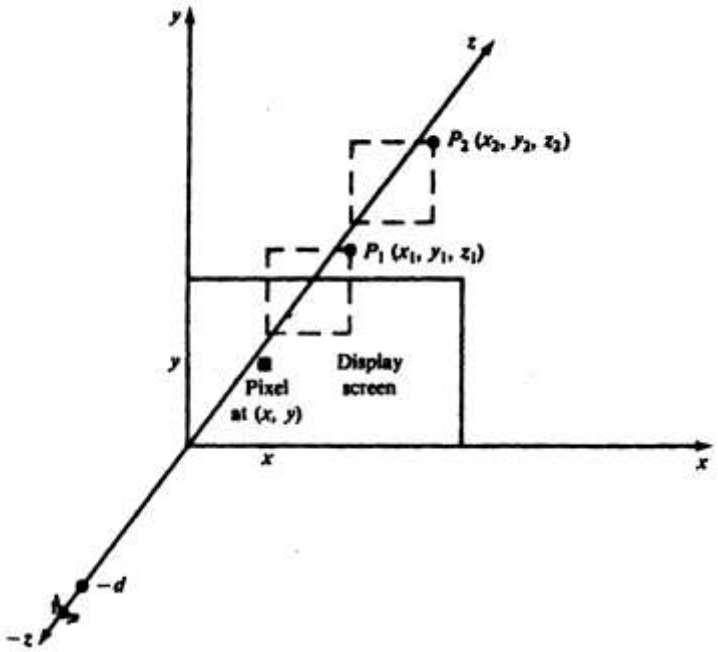


Figure 7.29

Although the Z-buffer algorithm requires Z-buffer storage proportional to the number of pixels on the screen, it does not require additional memory for storing all the objects comprising the scene. In fact, since the algorithm process polygons one at a time, the total number of objects in a scene can be arbitrarily large.

Back-Face Removal

Object surfaces that are oriented away from the viewer are called back-faces. The back-faces of an opaque polyhedron are completely blocked by the polyhedron itself and hidden from view. We can therefore identify and remove these back-faces based solely o their orientation without further processing (projection and scan-conversion) and without regard to other surfaces and objects in the scene.

Let N = (A,B,C) be the normal vector of a planar polygonal face, with N pointing in the direction the polygon is facing. Since the direction of viewing is the direction of the positive z axis (see Fig. 7.30), the polygon is facing away from the viewer when C > 0, (the angle between N and the z axis is less than 90°). The polygon is also classified as a back-face when C =0 since in this case it is parallel to the line of sight and its projection is either hidden as a back-face when C = 0, since in this case it is parallel to the line of sight and its projection is either hidden or overlapped by the edge(s) of some visible polygon(s).

Although this method identifies or removes back-faces quickly it does not handle polygons that face the viewer but are hidden (partially or completely) behind other surfaces. It can be used as a preprocessing step for other algorithms.

The Painter's Algorithm

Also called the depth sort or priority algorithm, the painter`s algorithm processes polygons as if they were been painted onto the view plane in the order of their distance from the viewer. More distance polygons are painted first. Nearer polygons are painted on or over more distance polygons, partially or totally obscuring them from view. The key to implementing this concept is to find a priority ordering of the polygons in order to determine which polygons are to be painted (i.e. scan-converted) first.

Any attempt at a priority ordering based on depth sorting alone results in ambiguities that must be resolved in order to correctly assign priorities. For example, when two polygons overlap, how do we decide which one obscures the other? (See. Fig. 7.30).

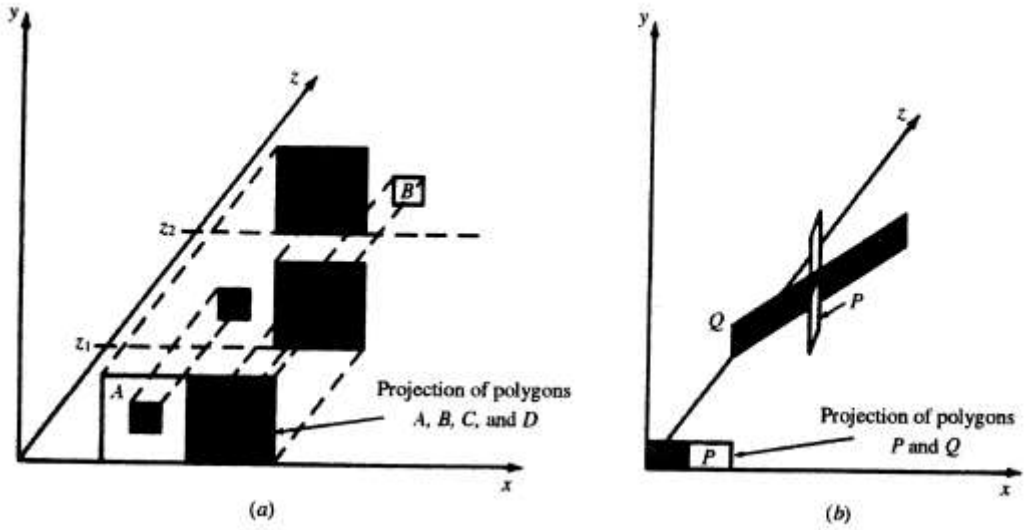


Figure 7.30

Assigning Priorities

We assign priorities to polygons by determining if a given polygon P obscures other polygon. If the answer is no, then P should be painted first. Hence the key test is to determine whether polygon P does not obscure polygon Q.

The z extent of a polygon is the region between the planes $z = z_{min}$ and $z = z_{max}$ (Fig. 7.31).Here

z_{min} is the smallest of the z coordinates of all the polygon`s vertices, and z_{max} is the largest.

Similar definitions hold for the x and y extents of a polygon. The intersection of the x, y, and z, extents is called the extent, or bounding box, of the polygon

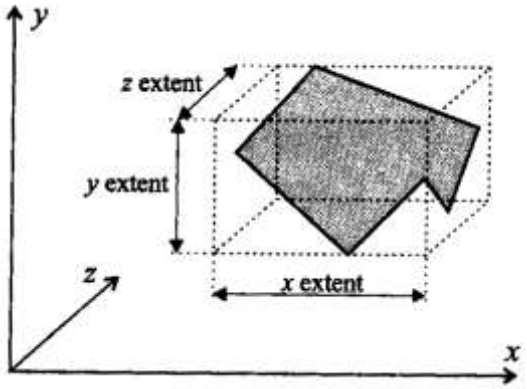


Figure 7.31

Testing Whether P Obscures Q

Polygon P does not obscure polygon Q if any one of the following tests, applied in sequential order, is true.

Test 0 : the z extents of P and Q do not overlap and $z_{Q\,max}$ of Q is smaller than $z_{p\,max}$ of P. Refer to Fig. 7.32.

Test 1 : The y extents of P and Q do not overlap, Refer to Fig. 7.33.

Test 2 : The x extents of P and Q do not overlap.

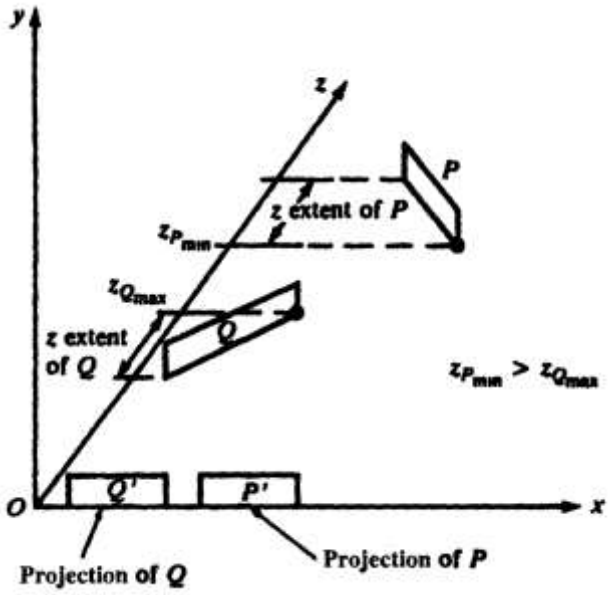


Figure 7.32

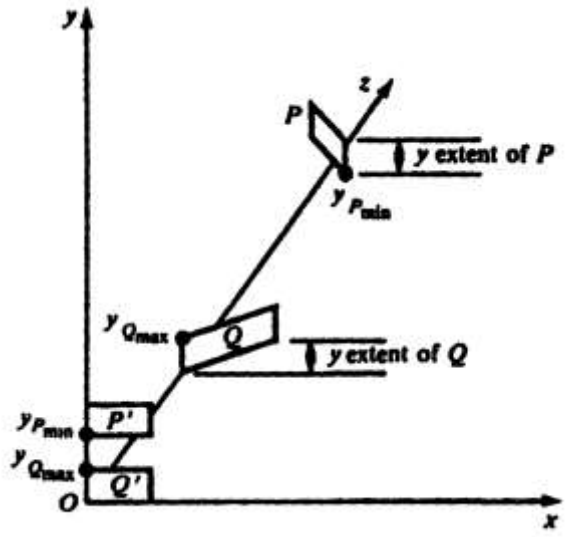


Figure 7.33

Test 3 : All the vertices of P lie on that side of the plane containing Q which is farthest from the viewpoint. Refer to Fig. 7.34.

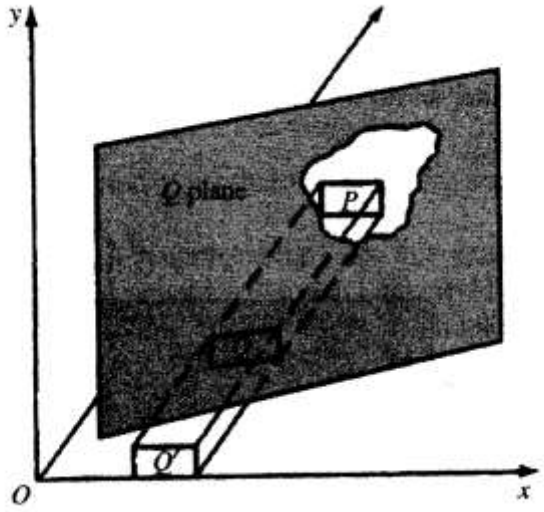


Figure 7.34

Test 4: All the vertices of Q lie on that side of the plane containing P which is closest to the viewpoint. Refer to Fig. 7.35.

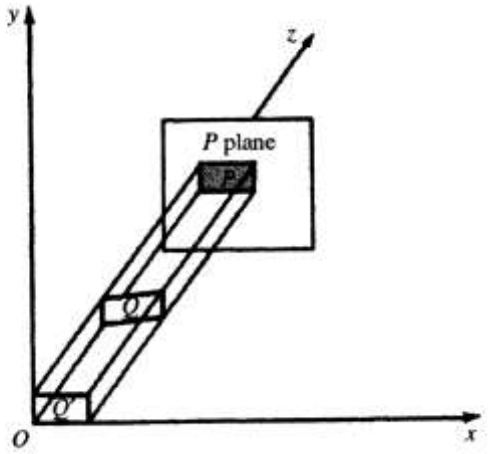


Figure 7.35

Test 5 : The projections of the polygons P and Q onto the view plane do not overlap. This is checked by comparing each edge of one polygon against each edge of the other polygon to search for intersections.

The Algorithm

1. Sort all polygons into a polygon list according to z max (the largest z coordinate of each polygon`s vertices). Starting from the end of the list, assign priorities for each polygon P, in order, as describes in steps 2 and 3 (below).

2. Find all polygons Q (proceeding P) in the polygon list whose z extents overlap that of P (test 0).

3. For each Q, perform tests 1 through 5 until true.

    a. If every Q passes, scan-convert polygon P.

    b. If false for some Q, swap P and Q on the list. Tag Q as swapped. If Q has already been tagged, use the plane containing polygon P to divide polygon Q into two polygons. Q_1 and Q_2. Remove Q from the list and place Q_1 and Q_2 on the list, in sorted order.

Sometimes the polygons are subdivided into triangles before processing, thus reducing the computational effort for polygon subdivision in step 3.

Scan – Line Algorithm

A scan-line algorithm consists essentially of two nested loops, an x–scan loop nested within a y-scan loop.

y-Scan

For each y value, say y = α, interested the polygons to be rendered with the scan plane y = α. This scan plane is parallel to the xz plane, and the resulting intersections are line segments in this plane (see Fig. 7.36).

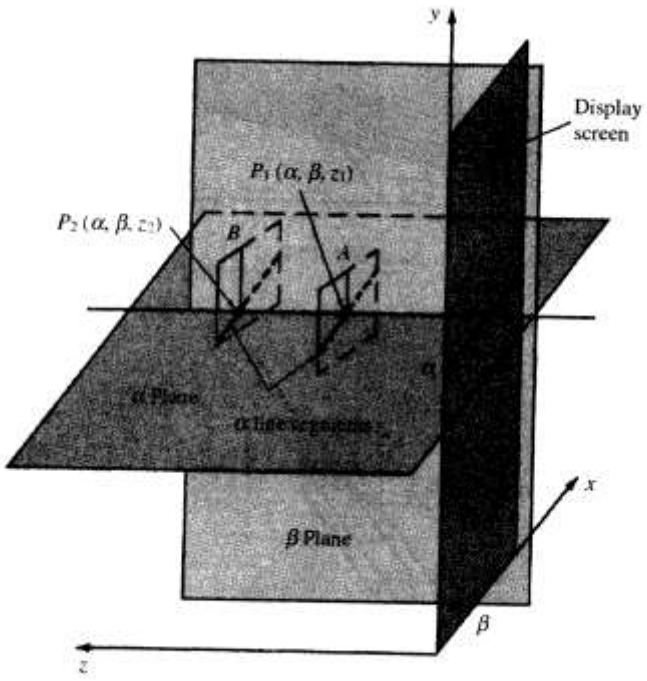


Figure 7.36

x-scan

1. For each x value, say x = f, intersect the line segments found above with the x scan line x = β lying on the y–scan plane. This intersection results in a set of points that lies on the x–scan line.

2. Sort these points with respect to their z coordinates. The point (x,y,z) with the smallest z value is visible, and the color of the polygon containing this point is the color set at the pixel corresponding to this point.

In order to reduce the amount of calculation in each scan –line loop, we try to take advantage of relationships and dependencies, called coherences, between different elements that comprise a scene.

Types of Coherence

1. Scan-line coherence: If a pixel on a scan line lies within a polygon, near it will most likely lie within the polygon.

2. Edge coherence: If an edge of a polygon intersects a given scan line, it will not likely intersect scan lines near the given one.

3. Area coherence: A small area of an image will most likely lie within a single polygon.

4. Spatial coherence: Certain properties of an object can be determined by examining the extent of the object, that is, a geometric figure which circumscribes the given object. Usually the extent is a rectangle or rectangle solid (also called a bounding box).

Scan-line coherence and edge coherence are both used to advantage in scan–converting polygons.

Spatial coherence is often used as a preprocessing step. For example, when determining whether polygons intersect, we can eliminate those polygons that don`t intersect by finding the rectangular extent of each polygon and checking whether the extents intersect – a much simpler problem (see Fig. 7.36). Note in Fig. 7.37) objects A and B do not intersect; however, objects A and C and B and C, do intersection. For example, the edge of object A is at coordinate P_3 = (7,4) and the edge of object B is at coordinate P_4 = (7,3)]. Of course, even if the extents intersect, this does not guarantee that the polygons intersect. See Fig. 7.38 and note that the extents A` and B` overlap even though the polygon do not.

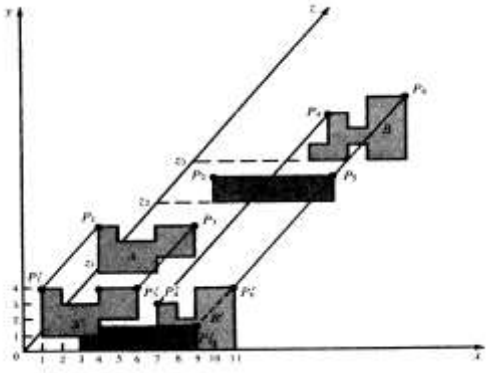


Figure 7.37

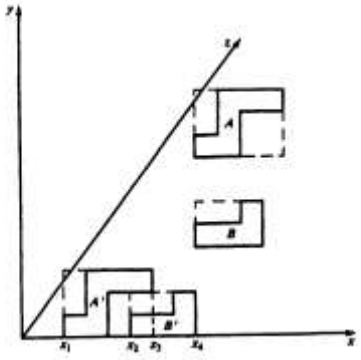


Figure: 7.38

Coherences can simplify calculations by making them incremental, as opposed to absolute.

A Scan – line Algorithm

In the following algorithm, scan line and edge coherence are used to enhance the processing done in the y-scan as follows. Since the y-scan loop constructs a list of potentially visible line segments, instead of reconstructing this list each time the y–scan line changes (absolute calculation), we keep the list and update it according to how it has changed (incremental calculation). This processing is facilitated by the use of what is called the edge list, and its efficient construction and maintenance is at the heart of the algorithm, under " a scan-line Algorithm").

The following data structure are created:

1. The edge list – contains all non horizontal edges (horizontal edges are automatically displayed) or the projections of the polygons in the scene. The

edges are sorted by the edge`s smaller y coordinate (y_{min}). Each edge entry in the edge list also contain.

a. The x coordinates of the end of the edge with the smaller y coordinate.

b. The y coordinate of the edge`s other end (y_{max}).

c. The increment $\Delta x = 1/m$.

d. A pointer indicating the polygon to which the edge belongs.

2. The polygon list – for each polygon, contains

a. The equation of the plane within which the polygon lies – used for depth determination, i.e., to find the z value at pixel (x, y,).

b. An IN/OUT flag, initialized to OUT (this flag is set depending on whether a given scan line is in or out of the polygon).

c. Color information for the polygon.

The algorithm proceeds as follows.

I. Initialization

a. Initialize each screen pixel to a background color.

b. Set y to the smallest y min value in the edge list.

Repeat steps II and III (below) until further processing can be performed.

II. y-scan loop. Activate edges whose y min is equal to y. Sort active edges in order of increasing x.

III. x – scan loop. Process, from left to right, each active edge as follows:

a. Invert the IN / OUT flag of the polygon in the polygon list which contains the edge. Count the number of active polygon whose IN/OUT flag is set to IN. If this number is I, only one polygon is visible. All pixel values from this edge and up to the next edge are set to the color of then polygon. If this number is greater than 1, determine the visible polygon by the smallest z value of each polygon at the pixel under consideration.

These z values are found from the equation of the plane containing the polygon, unless the polygon becomes obscured by another before the next edge is reached, in which case we set the remaining pixels to the color of the obscuring polygon. If this number is 0, pixels from this edge and up to the next one are left unchanged.

b. When the last active edge is processed, we then proceed as follows :

1. Remove those edges for which the value of y max equals the present scan-line value y. If no edges remain, the algorithm has finished.
2. For each remaining active edge, in order, replace x by x + 1/ m. This is the edge intersection with the next scan line y + 1.
3. Increment y to y + 1, the next scan line, and repeat step II.

Subdivision Algorithm

The subdivision algorithm is a recursive procedure based on a two – step strategy that first decides which projected polygons overlap a given area A on the screen and are therefore potentially visible in that area. Second, in each area these polygons are further tested to determine which ones will be visible within this area and should therefore be displayed. If a visibility decision cannot be made, this screen area, usually a rectangular window, is further subdivided either until a visibility decision can be made, or until the screen area is a single pixel.

Starting with the full screen as the initial area, the algorithm divides an area at each stage into four smaller areas, thereby generating a quad tree (see Fig. 7.39).

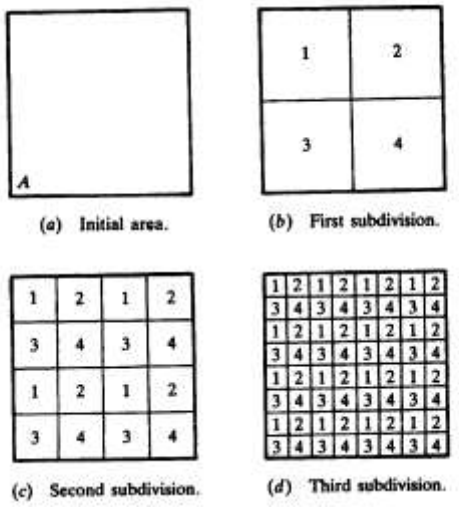


Figure 7.39

The processing exploits area coherence by classifying polygon P with respect to a given screen A into the following categories ; (1) surrounding polygon – polygon that completely contains the area [Fig.7.40(a)], (2) intersecting polygon – polygon that intersects the area [Fig. 7.40(b)], (3) contained polygon – polygon that is completely contained within the area [Fig. 7.40(c)], and (4) disjoint polygon – polygon that is disjoint from the area [Fig. 7.40 (d)].

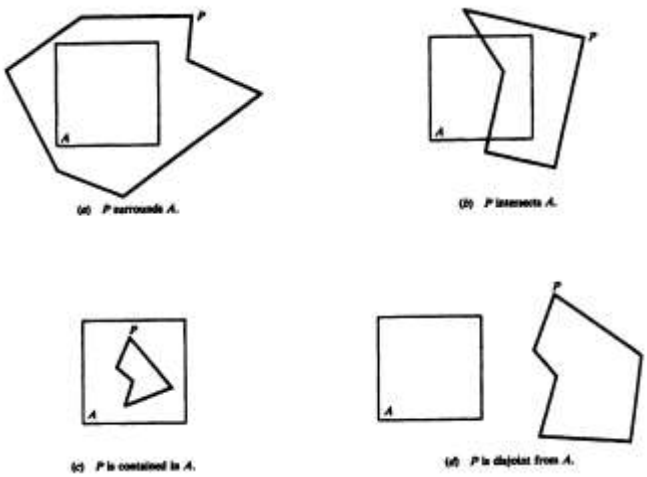


Figure 7.40

The Classification of the polygons within a picture is the main computational expense of the algorithm and is analogous to the clipping algorithms discussed in. With the use of one of these clipping algorithms, a polygon in category 2 (intersecting polygon) can be clipped into a contained polygon and a disjoint polygon (see Fig. 7.41). Therefore, we could proceed as if category 2 were eliminated.

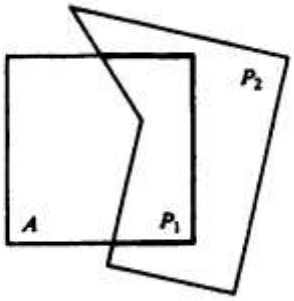


Figure 7.41

For a given screen area, we keep a potentially visible polygons list (PVPL), those in categories 1,2, and 3 (Disjoint polygons are clearly not visible). Also, note that on subdivision of a screen area, surrounding and disjoint polygons remain surrounding and disjoint polygons of the newly formed areas. Therefore, only contained and interesecting polygons need to be reclassified.

Removing Polygons Hidden by a Surrounding Polygon

The key to efficient visibility computation lies in the fact that a polygon is not visible if it is in back of a surrounding polygon. Therefore, it can be removed from the PVPL. To facilitate processing, this list is sorted by z_{min}, the smallest z coordinate of the polygon within this area. In addition for each surrounding polygon S, we also record its largest z coordinate $z_{s_{man}}$.

If, for a polygon P on the list $z_{p_{min}} > z_{s_{man}}$ (for a surrounding polygon S), then P is hidden by S and thus is not visible. In addition, all other polygons after P on the list will also be hidden by S, so we can remove these polygons from the PVPL.

Subdivision Algorithm

1. Initialize the area to be whole screen.
2. Create a PVPL with respect to an area sorted on z min (the smallest z coordinate of the polygon within the area). Place the polygons in their appropriate categories. Remove polygons hidden by a surrounding polygon and remove disjoint polygons.
3. Perform the visibility decision tests:
    a. If the list is empty, set all pixels to the background color.

b. If there is exactly one polygon in the list and it is a classified as intersecting (category 2) or contained (category 3), color (scan – convert) the polygon, and color the remaining area to the background color.

c. If there is exactly one polygon on the list and it is surrounding one, color the area the color of the surrounding polygon.

d. If the area is the pixel (x, y) and neither a, b, nor c applies, compute the z coordinate z(x, y) at pixel (x, y) of all polygons on the PVPL. The pixel is then set to the color of the polygon with the smallest z coordinate.

4. If none of the above cases has occurred, subdivide the screen area into fourths. For each area, go to step 2.

7.8 Summary

- To transform an object from its own model space to a common coordinate space, called world space, in which all objects, light sources, and the viewer coexist. This step is called the modeling transformation stage.
- Manipulation, Viewing , and construction of three-dimensional graphics images requires the use of three – dimensional geometric and coordinate transformation. These transformations are formed by composing the basic transformations translation, scaling, and rotation.
- An Object is displaced a given distance and direction from its original position.
- The process of scaling changes the dimensions of an object.
- Rotation in three dimensions is considerably more complex than rotation in two dimensions. In two dimensions, a rotation is prescribed by an angle of rotations require the prescription of an angle of rotation and an axis of rotation.
- The rotation about an arbitrary axis in space occurs in robotics animation and simulation.
- Once world-coordinate descriptions of the objects in a scene are converted to viewing coordinates, we can project the three-dimensional objects onto the two-dimensional view plane. There are two basic projection methods. In a

parallel projection, coordinate positions are transformed to the view plane along parallel lines.

- A parallel projection preserves relative proportions of objects, and this is the method used in drafting to produce scale drawings of three-dimensional objects.
- To obtain a perspective projection of a three-dimensional object, we transform points along projection lines that meet at the projection reference point.
- The process of constructing a perspective view introduces certain anomalies which enhance realism in terms of depth cues but also distort actual sizes and shapes.

7.9 Key Words

Three dimensional graphics, rotation, scaling, translation, parallel, perspective, projection, hidden line, surface elimination, viewing, clipping, wireframe, Bezier curves.

7.10 Self Assessment Questions (SAQ)

1. What happens when two polygons have the same z value and the Z-buffer algorithm is used?
2. Show that the alignment transformation satisfies the relation $A_v^{-1} = A_v^T$.
3. How many view planes (at the origin) produce isometric projections of an object?
4. Find the equations of the planes forming the view volume for the general parallel projection.
5. Hoe is the depth of a polygon determined by the painter's algorithm?
6. How does the subdivision algorithm exploit are coherence?
7. How can hidden-surface algorithms be used to eliminate hidden lines as applied to polygonal mesh models?
8. Find a normalization transformation from the window whose lower left corner is at(0,0) and upper right corner is at (4,3) onto the normalized device screen so that aspect ratios are preserved.

7.11 References/ Suggested Readings

1. High Resolution Computer Graphics using Pascal/C, by Ian O. Angell and Gareth Griffith, John Wiley & Sons
2. Computer Graphics , Second Edition , by Pradeep K. Bhatia , I.K .International Publisher.
3. Computer Graphics (C Version), Donald Hearn and M. Pauline Baker, Prentice Hall,
4. Advanced Animation and Rendering Techniques, Theory and Practice, Alan Watt and Mark Watt , ACM Press/Addison-Wesley

SUBJECT: COMPUTER GRAPHICS

COURSE CODE: MCA-44(iv)	AUTHOR: Abhishek Taneja

LESSON NO. 8

The Concept of Multimedia and GKS

UNIT STRUCTURE

8.0 Objectives

At the end of this chapter the reader will be able to:

- Describe the concept of multimedia
- Describe multimedia hardware and software
- Describe multimedia authoring tools
- Describe Graphic software
- Describe GKS attributes, primitives, window, viewport and display routines

8.1 Introduction

As the name suggests, multimedia is a set of more than one media element used to produce a concrete and more structured way of communication. In other words multimedia is simultaneous use of data from different sources. These sources in multimedia are known as media elements. With growing and very fast changing information technology, Multimedia has become a crucial part of computer world. Its importance has realised in almost all walks of life, may it be education, cinema, advertising, fashion and what not.

Throughout the 1960s, 1970s and 1980s, computers have been restricted to dealing with two main types of data - words and numbers. But the cutting edge of information technology introduced faster system capable of handling graphics, audio, animation and video. And the entire world was taken aback by the power of multimedia.

8.2 What is Multimedia?

Multimedia is nothing but the processing and presentation of information in a more structured and understandable manner using more than one media such as text, graphics, animation, audio and video. Thus multimedia products can be an academic presentation, game or corporate presentation, information kiosk, fashion-designing etc. Multimedia systems are those computer platforms and software tools that support the interactive uses of text, graphics, animation, audio, or motion video. In other words, a computer capable of handling text, graphics, audio, animation and video is called multimedia computer. If the sequence and timing of these media elements can be controlled by the user, then one can name it as *Interactive Multimedia*.

8.3 Hardware and Software for Multimedia

Multimedia Software

Multimedia software comprises a wide variety of software that allows you to combine images with text, music and sound, animation, video, and other special effects. Many kinds of multimedia software are available. *Multimedia software* is sometimes broadly grouped as:

- Multimedia Authoring Tools
- Multimedia Tools for the Web
- Multimedia Presentation Software

Multimedia Authoring can be used to create anything from simple slide shows to full-blown games and interactive applications. A professional multimedia development program is called an authoring tool or authoring software. An authoring tool allows you to give the user interactive control over the sequence and timing of videos, graphics and animation. It also provides a scripting language, also called a macro or authoring language to control the action.

8.3.1 Multimedia Authoring Tool

A *multimedia authoring tool* is a program that helps you write multimedia applications. A multimedia authoring tool enables you to create a final application merely by linking together objects, such as a paragraph of text, an illustration, or a song. Attractive and useful multimedia applications can be produced by defining the objects' relationships to each other, and by sequencing them in an appropriate order. A multimedia authoring tool requires less technical knowledge to master and are used exclusively for applications that present a mixture of textual, graphical, and audio data. Multimedia authoring tools are used to create interactive presentations, screen savers, games, CDs and DVDs. A multimedia authoring tool supports a wide variety of multimedia file formats including images, video, and sound. Clip-art and sound libraries are rarely bundled with a multimedia authoring tool.

8.3.2 Multimedia Presentation

A Multimedia Presentation can be described as an intuitive book. Each page of the book can include pictures, animations, sounds and other controls. The pages can turn themselves or wait for their users to click on Next. Unlike the pages of a traditional book, they can talk to their readers too. No matter what your particular need may be, whether to e-mail an electronic photograph album to your relatives, allow users of your web page to download a multimedia catalog, show prospective clients samples of your work or make a dull subject come alive in class, *multimedia presentation software* can transform your ideas into a complete, professional application. In a live multimedia presentation, the slides are commonly projected onto large screens or printed on overhead transparencies. The slides may be distributed in printed form as handouts to accompany the live presentation.

8.3.3 Multimedia Presentation Software

Multimedia Presentation Software can build self-displaying, interactive slide shows, electronic books and brochures, clickable advertisements, distributable portfolios and all sorts of other multimedia applications. Almost everything it can accomplish requires nothing more than a few mouse clicks. It has a short learning curve. Some multimedia presentation software documents are EXE files - they can be opened on any system. They can be freely distributed on disks and CD-ROMs, and over the Internet. *Multimedia Presentation Software* has some great features like the ability to:

- Build and distribute slide show style presentations.
- Write your own clickable advertisements, to distribute alone or with other products.
- Design training materials that include pictures, sounds and interactive elements.
- Compile a distributable portfolio.
- Completed presentations are frequently published in multiple formats, which may include print, the Web, or electronic files.

Multimedia Presentation Software requires a faster processor. A display adapter and screen driver capable of displaying more than 16 million colors is needed to get the best out of it.

Multimedia Hardware

For producing multimedia you need hardware, software and creativity. In this section we will discuss the multimedia equipment required in a personal computer (PC) so that multimedia can be produced.

(a) Central Processing Unit

As you know, Central Processing Unit (CPU) is an essential part in any computer. It is considered as the brain of computer, where processing and synchronization of all activities takes place. The efficiency of a computer is judged by the speed of the CPU in processing of data. For a multimedia computer a *Pentium* processor is preferred because of higher efficiency. However, the CPU of multimedia computer should be at least 486 with *math coprocessor*. The Pentium processor is one step up the evolutionary chain from the 486 series processor and Pentium Pro is one step above the Pentium. And the speed of the processor is measured in megahertz. It defines the number of commands the computer can perform in a second. The faster the speed, the faster the CPU and the faster the computer will be able to perform. As the multimedia involves more than one medial element, including high-resolution graphics, high quality motion video, and one need a faster processor for better performance.

In today's scenario, a Pentium processor with MMX technology and a speed of 166 to 200 MHz (Megahertz) is an ideal processor for multimedia. In addition to the processor one will need a minimum 16 MB RAM to run WINDOWS to edit large images or video clips. But a 32 or 64 MB RAM enhances the capacity of multimedia computer.

(b) Monitor

As you know that monitor is used to see the computer output. Generally, it displays 25 rows and 80 columns of text. The text or graphics in a monitor is created as a result of an arrangement of tiny dots, called *pixels*. Resolution is the amount of details the monitor can render. Resolution is defined in terms of horizontal and vertical pixel (picture elements) displayed on the screen. The greater the number of pixels, better visualization of the image.

Like any other computer device, monitor requires a source of input. The signals that monitor gets from the processor are routed through a graphics card. But there are

computers available where this card is in-built into the motherboard. This card is also called the graphics adapter or display adapter. This card controls the individual pixels or tiny points on a screen that make up image. There are several types of display adapter available. But the most popular one is Super Virtual Graphics Arrays (SVGA) card and it suits the multimedia requirement. The advantage of having a SVGA card is that the quality of graphics and pictures is better.

Now the PCs, which are coming to the market, are fitted with SVGA graphics card. That allows images of up to 1024 ´ 768 *pixels* to be displayed in up to 16 millions of colours. What determines the maximum resolution and color depth is the amount of memory on the display adapters. Often you can select the amount of memory required such as 512KB, 1MB, 2MB, 4MB, etc. However, standard multimedia requirement is a 2MB of display memory (or Video RAM). But one must keep in mind that this increases the speed of the computer, also it allows displaying more colours and more resolutions. One can easily calculate the minimum amount of memory required for display adapter as (Max. Horizontal Resolution x Max. Vertical Resolution ´ Colour Depths. in Bits)/8192 = The minimum video (or display) memory required in KB.

For example, if SVGA resolution (800´600) with 65,536 colours (with colour depth of 16) you will need

(800 x 600 x 16) / 8192 = 937.5 KB, i.e., approximately 1 MB of display memory.

Another consideration should be the refresh rate, i.e., the number of times the images is painted on the screen per second. More the refresh rate, better the image formation. Often a minimum of 70-72Mhz is used to reduce eye fatigue. As a matter of fact higher resolution requires higher refresh rates to prevent screen flickers.

(c) Video Grabbing Card

As we have already discussed, we need to convert the analog video signal to digital signal for processing in a computer. Normal computer will not be able to do it alone. It requires special equipment called video grabbing card and software to this conversion process. This card translates the analog signal it receives from conventional sources such as a VCR or a video camera, and converts them into digital format. The software available with it will capture this digital signal and store them

into computer file. It also helps to compress the digitized video so that it takes lesser disk space as compared to a non-compressed digitized video.

This card is fitted into a free slot on the motherboard inside the computer and gets connected to an outside source such as TV, VCR or a video camera with the help of a cable. This card receives both video and audio signal from the outside source and conversion from analog to digital signal takes place. This process of conversion is known as sampling. This process converts the analog signal to digital data streams so that this signal can be stored in binary data format of 0's and 1's. This digital data stream is then compressed using the video capturing software and stores them in the hard disk as a file. This file is then used for incorporation into multimedia. This digitized file can also be edited according to the requirements using various editing software such as Adobe Premiere.

A number of digitizer or video grabbing cards are available in the market. However, one from Intel called Intel Smart Video Recorder III does a very good job of capturing and compressing video.

(d) Sound Card

Today's computers are capable of creating the professional multimedia needs. Not only you can use computer to compose your own music, but it can also be used for recognition of speech and synthesis. It can even read back the entire document for you. But before all this happens, we need to convert the conventional sound signal to computer understandable digital signals. This is done using a special component added to the system called sound card. This is installed into a free slot on the computer motherboard. As in the case of video grabber card, sound card will take the sound input from outside source (such as human voice, pre-recorded sounds, natural sounds etc.) and convert them into digital sound signal of 0's and 1's. The recording software used alongwith the sound card will store this digitised sound stream in a file. This file can latter be used with multimedia software. One can even edit the digitised sound file and add special sound effects into it.

Most popular sound card is from Creative Systems such as Sound Blaster-16, AWE32, etc. AWE32 sound card supports 16 channel, 32 voice and 128 instruments and 10 drums sound reproduction. It also has CD-ROM interface.

(e) CD-ROM Drive

CD-ROM is a magnetic disk of 4.7 inches diameter and it can contain data up to 680 Megabytes. It has become a standard by itself basically for its massive storage capacity, faster data transfer rate. To access CD-ROM a very special drive is required and it is known as CD-ROM drive. Let us look into the term ROM that stands for 'Read Only Memory'. It means the material contained in it can be read (as many times, as you like) but the content cannot be changed.

As multimedia involves high resolution of graphics, high quality video and sound, it requires large amount of storage space and at the same time require a media, which can support faster data transfer. CD-ROM solves this problem by satisfying both requirements.Similar to the hard disk drive, the CD-ROM drive has certain specification which will help to decide which drive suit best to your multimedia requirement.

(i) Transfer Rate

Transfer rate is basically the amount of data the drive is capable of transferring at a sustained rate from the CD to the CPU. This is measured in KB per second. For example, 1x drive is capable of transferring 150KB of data from the CD to the CPU. In other terms 1x CD drive will sustain a transfer rate of 150KB/sec, where x stands for 150 KB. This is the base measurement and all higher rates are multiple of this number, x. Latest CD-ROM drive available is of 64x, that means it is capable of sustaining a data transfer rate of 64x150=9600 KB =9.38MB per second from the CD to the CPU.

(ii) Average Seek time

The amount of time lapses between request and its delivery is known as average seeks time. The lower the value better the result and time is measured in milliseconds. A good access time is 150ms.

Recently computer technology has made tremendous progress. You can now have CDs which can 'write many, read many' times. This means you can write your files in to a blank CD through a laser beam. The written material can be read many times and

they can even be erased and re-written again. Basically this re-writable CD's can be used a simple floppy disk.

(f) Scanner

Multimedia requires high quality of images, graphics to be used. And it takes lot of time creating them. However there are ready-made sources such as real life photographs, books, arts, etc. available from where one easily digitized the required pictures. To convert these photographs to digital format, one need a small piece of equipment called scanner attached to the computer. A scanner is a piece of computer hardware that sends a beam of light across a picture or document and records it. It captures images from various sources such as photograph, poster, magazine, book, and similar sources. These pictures then can be displayed and edited on a computer. The captured or scanned pictures can be stored in various formats like;

File Format Explanation

PICT - A widely used format compatible with most Macintosh

JPEG - Joint Photographic Experts Group - a format that compresses files and lets you choose compression versus quality

TIFF - Tagged Image File Format - a widely used format compatible with both Macintosh and Windows systems

Windows BMP - A format commonly used on MS-DOS and MS-Windows computers

GIF - Graphics Interchange Format - a format used on the Internet, GIF supports only 256 colours or grays

Scanners are available in various shapes and sizes like hand-held, feed-in, and flatbed types. They are also for scanning black-and-white only or color. Some of the reputed vendors of scanner are Epson, Hewlett-Packard, Microtek and Relisys.

(g) Touchscreen

As the name suggests, touchscreen is used where the user is required to touch the surface of the screen or monitor. It is basically a monitor that allows user to interact with computer by touching the display screen. This uses beams of infrared light that are projected across the screen surface. Interrupting the beams generates an electronic signal identifying the location of the screen. And the associated software interprets the signal and performs the required action. For example, touching the screen twice in quick succession works as double clicking of the mouse. Imagine how useful this will be for visually handicapped people who can identify things by touching a surface. Touchscreen is normally not used for development of multimedia, it is rather used for multimedia presentation arena like trade show, information kiosk, etc.

8.4 Applications Area for Multimedia

Placing the media in a perspective within the instructional process is an important role of the teacher and library professional. Following are the possible areas of application of multimedia:

- Can be used as reinforcement
- Can be used to clarify or symbolize a concept
- Creates the positive attitude of individuals toward what they are learning and the learning process itself can be enhanced.
- The content of a topic can be more carefully selected and organized
- The teaching and learning can be more interesting and interactive
- The delivery of instruction can be more standardized.
- The length of time needed for instruction can be reduced.

The instruction can be provided when and where desired or necessary

8.5 Components of Multimedia

(i) Text

Inclusion of textual information in multimedia is the basic step towards development of multimedia software. Text can be of any type, may be a word, a single line, or a paragraph. The textual data for multimedia can be developed using any text editor.

However to give special effects, one needs graphics software which supports this kind of job.

(ii) Graphics

Another interesting element in multimedia is graphics. As a matter of fact, taking into consideration the human nature, a subject is more explained with some sort of pictorial/graphical representation, rather than as a large chunk of text. This also helps to develop a clean multimedia screen, whereas use of large amount of text in a screen make it dull in presentation.

Unlike text, which uses a universal ASCII format, graphics does not have a single agreed format. They have different format to suit different requirement. Most commonly used format for graphics is .BMP or bitmap pictures. The size of a graphics depends on the resolution it is using. A computer image uses *pixel* or *dots* on the screen to form itself. And these dots or pixel, when combined with number of colors and other aspects are called resolution. Resolution of an image or graphics is basically the pixel density and number of colors it uses. And the size of the image depends on its resolution. A standard VGA (Virtual Graphics Arrays) screen can display a screen resolution of 640 ´ 480 = 307200 pixel. And a Super VGA screen can display up-to 1024 ´ 768 = 786432 pixel on the screen. While developing multimedia graphics one should always keep in mind the image resolution and number of colors to be used, as this has a direct relation with the image size. If the image size is bigger, it takes more time to load and also requires higher memory for processing and larger disk-space for storage.

(iii) Animation

Moving images have an overpowering effect on the human peripheral vision. Followings are few points for its popularity.

Showing continuity in transitions:

Animation is a set of static state, related to each other with transition. When something has two or more states, then changes between states will be much easier for users to understand if the transitions are animated instead of being instantaneous. An

animated transition allows the user to track the mapping between different subparts through the perceptual system instead of having to involve the cognitive system to deduce the **mappings.**

Indicating dimensionality in transitions:

Sometimes opposite animated transitions can be used to indicate movement back and forth along some navigational dimension. One example used in several user interfaces is the use of zooming to indicate that a new object is "grown" from a previous one (e.g., a detailed view or property list opened by clicking on an icon) or that an object is closed or minimized to a smaller representation. Zooming out from the small object to the enlargement is a navigational dimension and zooming in again as the enlargement is closed down is the opposite direction along that dimension.

Illustrating change over time

Since animation is a time-varying display, it provides a one-to-one mapping to phenomena that change over time. For example, deforestation of the rain forest can be illustrated by showing a map with an animation of the covered area changing over time.

Multiplexing the display

Animation can be used to show multiple information objects in the same space. A typical example is client-side imagemaps with explanations that pop up as the user moves the cursor over the various hypertext anchors.

Enriching graphical representations

Some types of information are easier to visualize with movement than with still pictures. Consider, for example, how to visualize the tool used to remove pixels in a graphics application.

Visualizing three-dimensional structures

As you know the computer screen is two-dimensional. Hence users can never get a full understanding of a three-dimensional structure by a single illustration, no matter

how well designed. Animation can be used to emphasize the three-dimensional nature of objects and make it easier for users to visualize their spatial structure. The animation need not necessarily spin the object in a full circle - just slowly turning it back and forth a little will often be sufficient. The movement should be slow to allow the user to focus on the structure of the object.

You can also move three-dimensional objects, but often it is better if you determine in advance how best to animate a movement that provides optimal understanding of the object. This pre-determined animation can then be activated by simply placing the cursor over the object. On the other hand, user-controlled movements requires the user to understand how to manipulate the object (which is inherently difficult with a two-dimensional control device like the mouse used with most computers - to be honest, 3D is never going to make it big time in user interfaces until we get a true 3D control device).

Attracting attention

Finally, there are a few cases where the ability of animation to dominate the user's visual awareness can be turned to an advantage in the interface. If the goal is to draw the user's attention to a single element out of several or to alert the user to updated information then an animated headline will do the trick. Animated text should be drawn by a one-time animation (e.g., text sliding in from the right, growing from the first character, or smoothly becoming larger) and never by a continuous animation since moving text is more difficult to read than static text. The user should be drawn to the new text by the initial animation and then left in peace to read the text without further distraction.

One of the excellent software available to create animation is Animator Pro. This provides tools to create impressive animation for multimedia development.

(iv) Video

Beside animation there is one more media element, which is known as video. With latest technology it is possible to include video impact on clips of any type into any multimedia creation, be it corporate presentation, fashion design, entertainment games, etc.

The video clips may contain some dialogues or sound effects and moving pictures. These video clips can be combined with the audio, text and graphics for multimedia presentation. Incorporation of video in a multimedia package is more important and complicated than other media elements. One can procure video clips from various sources such as existing video films or even can go for an outdoor video shooting.

All the video available are in analog format. To make it usable by computer, the video clips are needed to be converted into computer understandable format, i.e., digital format. Both combinations of software and hardware make it possible to convert the analog video clips into digital format. This alone does not help, as the digitised video clips take lots of hard disk space to store, depending on the frame rate used for digitisation. The computer reads a particular video clip as a series of still pictures called *frames*. Thus video clip is made of a series of separate frames where each frame is slightly different from the previous one. The computer reads each frame as a bitmap image. Generally there are 15 to 25 frames per second so that the movement is smooth. If we take less frames than this, the movement of the images will not be smooth.

To cut down the space there are several modern technologies in windows environment. Essentially these technologies compress the video image so that lesser space is required.

However, latest video compression software makes it possible to compress the digitised video clips to its maximum. In the process, it takes lesser storage space. One more advantage of using digital video is, the quality of video will not deteriorate from copy to copy as the digital video signal is made up of digital code and not electrical signal. Caution should be taken while digitizing the video from analog source to avoid frame droppings and distortion. A good quality video source should be used for digitization.

(v) Audio

Audio has a greater role to play in multimedia development. It gives life to the static state of multimedia. Incorporation of audio is one of the most important features of multimedia, which enhance the multimedia usability to its full potential. There are several types of sound, which can be used in multimedia. They are human voices,

instrumental notes, natural sound and many more. All these can be used in any combination as long as they give some meaning to their inclusion in multimedia.

- There are many ways in which these sounds can be incorporated into the computer. For example;
- Using microphone, human voice can directly be recorded in a computer.
- Pre-recorded cassettes can be used to record the sound into computer.
- Instrumental sound can also be played directly from a musical instrument for recording into the computer.

8.6 Multimedia Authoring Tools

See section 8.3.1

8.7 Computer Graphic Software

To Graphics Kernel System (GKS) was development in to the need for a standardized method of developing graphic program. It represent a standard graphic interface with consistent syntax. Furthermore, GKS was designed so that it may be bound by means of subroutines to most common programming language such as C, FORTRAN 77, PASCAL, and BASIC.

GKS represents a programming language in the sense that it presents the programmer with a consistent set of reserved words with specific language bindings used within a specific syntactical structure. For example, the GKS reserved word for plotting points is “POLYMARKER.” (the reserved word POLYMAKER is followed by its corresponding parameters, n, X, Y, where n is the number of points, X is the array of the x co-ordinates and Y is the array of the y co-ordinates.) Thus the GKS statements

X(1) = 3 Y (1) =2

POLYMARKER (1,X,Y)

Represents a language-independent command-data list that would plot the point (3,2). In actual implementation with a programming language, in this case FORTRAN, the command-data structure would appear in this form:

DIMENSION X(1), Y(1)

X(1) =3

Y(1) =2

CALL GPM (n, X, Y)

(Note: in the BASIC implementation, an ampersand interpreter is used.)

It is important to remember that GKS presents a standardized body of consistence commands, producers, and language binding and syntax. This allows a GKS-based system to be portable and take advantage of improved algorithm. For example, a system using the slope method of line plotting could easily be upgraded to make use of Bresenham's line algorithm.

To avoid confusion. From this point on only the GKS names of commands will be shown. For example, the FORTRAN implementation CALL GPM (n, X, Y,) will be shown only as POLY MARKER (n, X, Y,)

8.8 GKS Primitive

The graphic kernel system is based on four basic primitive: POLYLINE, POLYMAKER,FILL AREA, and TEXT. The syntax of each of the primitive is as follows:

POLYLINE (n, X, Y)

POLIMARKER (n, X, Y)

FILL AREA (n, X, Y)

Where n= number of data points

X = X array

Y = Y array

TEXT (x, y, "string")

Where (x, y)= the starting coordinates for the starting and "string" = the text that is to be out put.

8.9 GKS Attributes

Each GKS primitive can have many attributes, which are set with an index number and the SET command. For example, the default value for POLYLINE INDEX is 1. a POLYLINE INDEX of generates a solid line. Therefore, if the X array = (1, 4) and the Y array= (2, 8), the following would generate a solid line from point (1, 2) to point (4, 8) (see fig. 8.1).

Set Polyline Index(1)

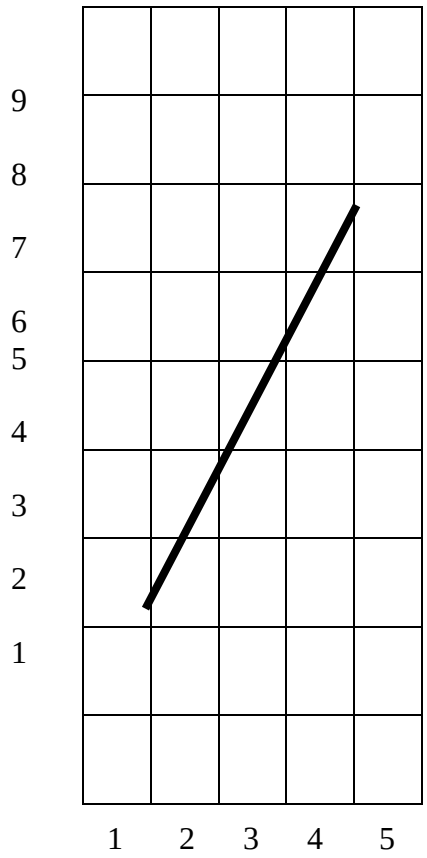


Figure 8.1

The GKS POLYLINE INDEX of 2 creates a dashed line. Thus, using the same data, X array =(1, 4) and Y array = (2, 8), the following code would generate a dashed line from (1, 2) to (4, 8) (see fig. 8.2).

Set Polyline index(2)

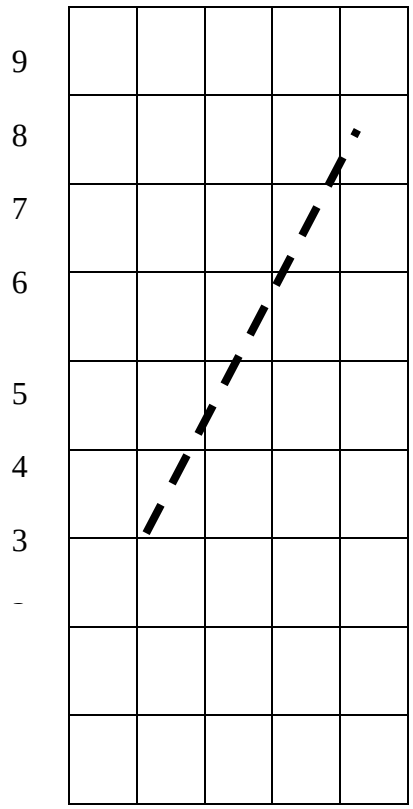


Figure 8.2

In addition to the POLYLINE INDEX, there are also index settings for POLYMARKER, FILL AREA, and TEXT. In each case, the SET function INDEX is used to set the attributes of the command prior to its use. For example, using the data X array = (2, 6, 6, 2) and Y array = (2, 2, 6, 6) with the commands

SET POLYLINE INDEX (1)

POLYMARKER (4, X, Y)

Would appear as shown in fig. 8.3 (a),

SET POLYMARKER INDEX (3)

POLYMARKER (4, X, Y)

Would appear as shown in fig. 8.3 (b), and

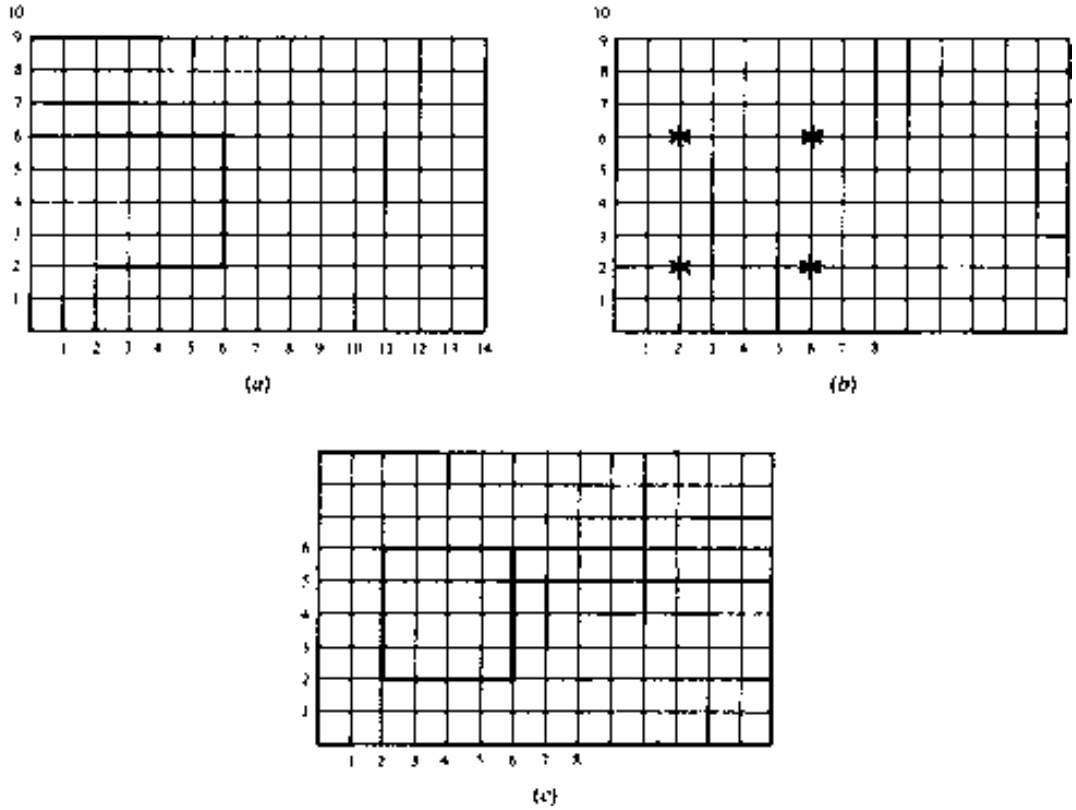


Figure 8.3

SET FILL AREA INDEX(0)

FILL AREA (4,X,Y)

Note that FILL AREA always connects the first and last points in the array 8.3 (c)

Other attributes supported by GKS allow the programmer to set the width of lines, change, color, select the style of marker used by POLYMAKER, and change the style and pattern used by FILL AREA. For example, the attribute value 3 for POLYMAKER gives an asterisk [see Fig. 8.3 (c)].

Text font is selected with the SET TEXT INDEX (n) command. However, text has several other attributes which can be changed with other set commands. These set commands allow the programmer to select character height, slant, color, spacing, and angle. Unique to text are the SET CHARACTER UO VECTOR (X, Y) and the SET

TEXT PATH (path) commands. CHARACTER UP VECTOR (X, Y) sets the slope at which text will be printed. Here , X represents the change in x, and Y, the change in y. For example

SET CHARACTER UO VECTOR (1, 1)

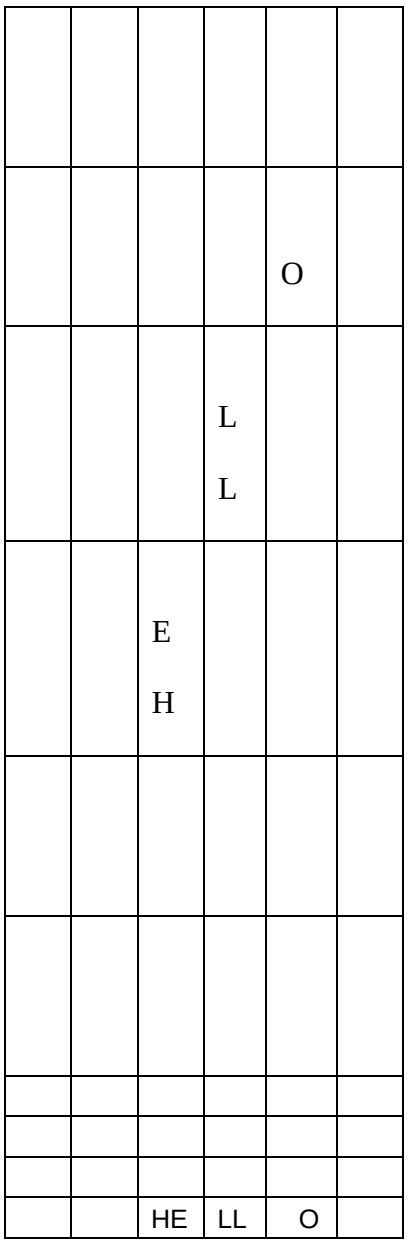
TEXT (2, 2, “HELLO”)

Would appear as in Fig. 8.4, while

SET CHARACTER UP VECTOR (1, 0)

TEXT (2, 2, “HELLO”)

Would appear as in Fig. 8.4.



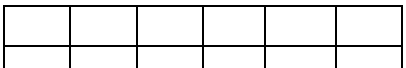


Figure 8.4

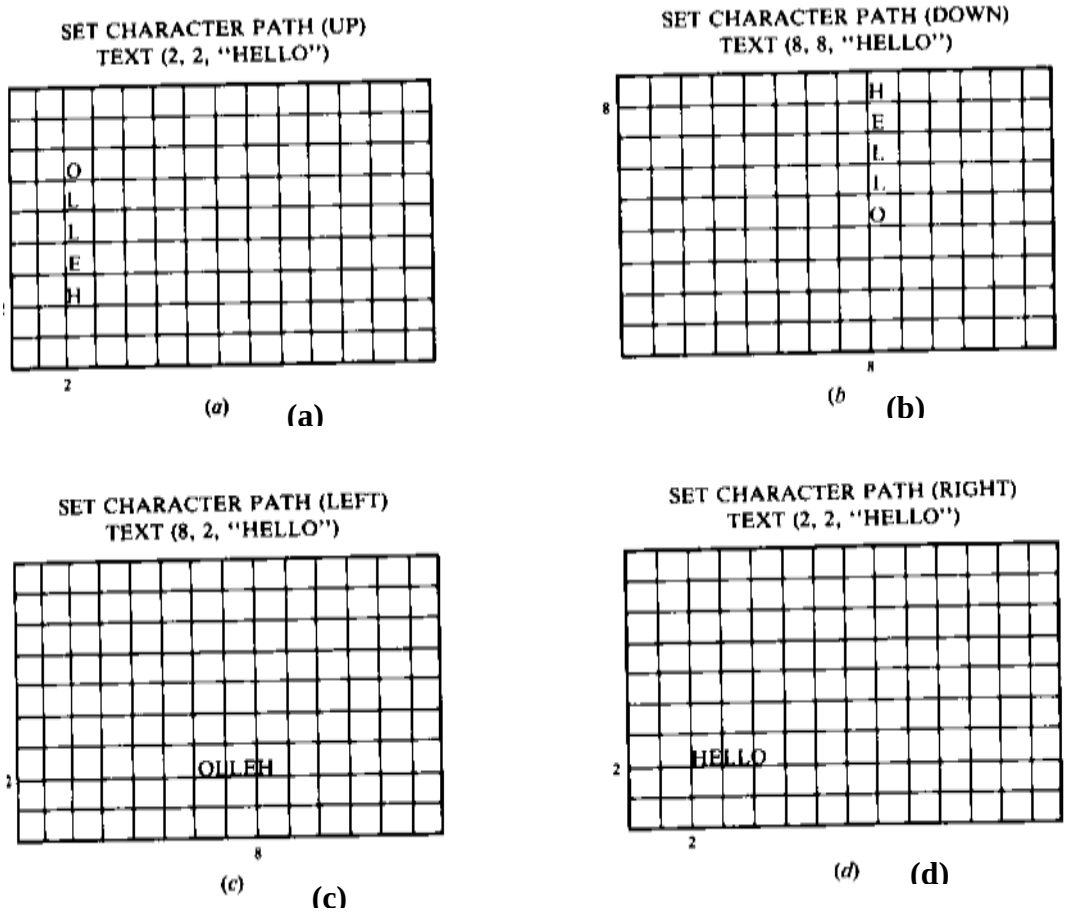


Figure 8.5

The character path attribute sets the direction in which the characters will be printed. The programmer can set character path UP, DOWN, LEFT, or RIGHT [see Figs. 8.5 (a) through 8.5 (d)].

8.10 GKS Window and Viewport

The format of the GKS VIEWPORT command is as follows

SET VIEWPORT(n, x_1, x_2, y_1, y_2)

Where n = nth view port

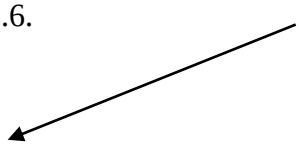
(x_1. y_1) = lower left corner of view port in word coordinates.

(x_2 . y_2) = upper right corner of view port in world coordinates

for example.

SET VIEWPORT (n, 10, 100,10, 100)

Would define the area shown in fig. 8.6.



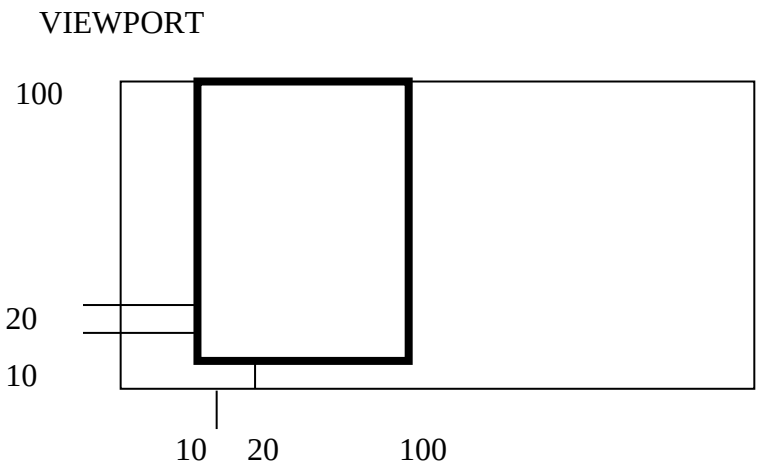


Figure 8.6

The GKS WINDOW command is as follows:

SET WINDOW (n, x_1, x_2, , y_1, y_2,)

Where n = nth window

(x_1 , y_1) = value of lower left corner of window in user coordinates

(x_2 , y_2) = value of upper right of window in user coordinates

the WINDOW command thus specifies how data will be mapped onto the display. For example, the following would result in fig. 8.7:

X (1) = 10, Y(1) = 10

SET VIEWPORT (1, 10, 100, 10, 100)

SET WINDOW (1, 0, 20, 0, 10)

POLYMARKER (1, X, Y)

As GKS supports multiple view ports and windows, a provision to indicate which view port is to be written to is required. View port selection is done with the SELECT NORMALIZATION TRANFORMATION (n) command, where n represents the numb3er of the view port to be selected.

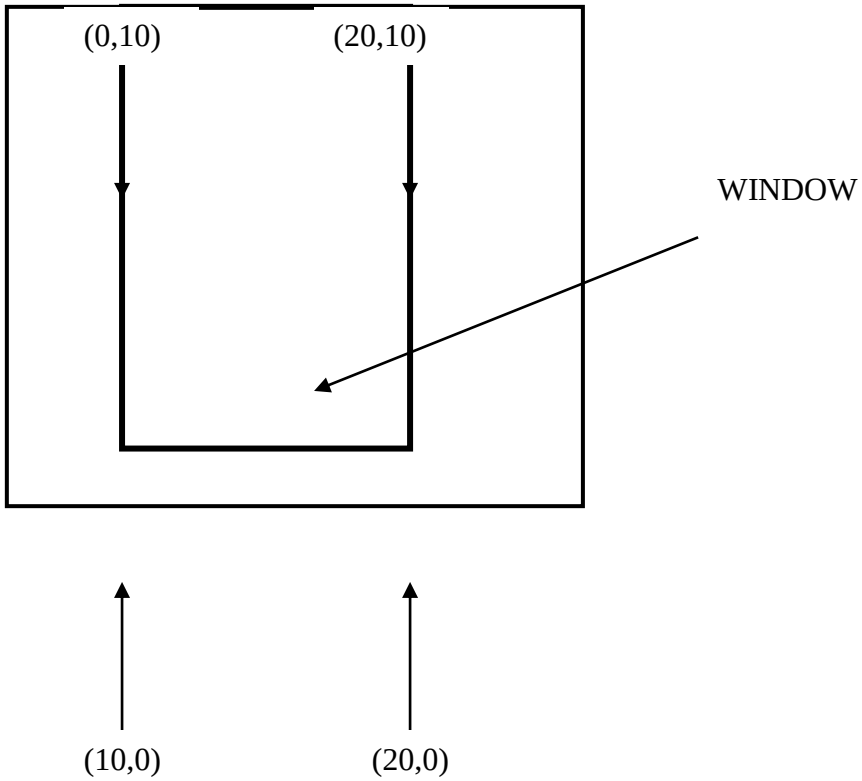


Figure 8.7

8.11 GKS Display Subroutines

While the four basic primitives supported by GKS-POLYLINE, POLY MARKER, FILLAREA, and TEXT-along with their attributes allow the programmer to construct a wide range of images, most developers would like to have access to some other commonly used primitives. For example, in a charting application, use of an axis command would be valuable.

When constructing commands, it is important to allow for future flexibility.

For example, an axis command should allow control over tick mark intervals, length of tick marks, style of tick marks, length of axes, and axes intersection point. For the

s9implest case, no tick marks are used; the axes run the length of the current window and intersect at the origin. The resulting code for simple axes might appear as follows:

```
SUBROUTINE AXIS (X, Y)
DIMINSION XA (2)YA (2)
XA (1) =X(1)
YA (1) = 0
XA(2)=X(2)
YA(2)=0
CALL GPL (2, XA, YA)
XA(1=0
YA(1)=Y(1)
XA(2)=0
YA(2)=Y(2)
CALL GPL (2, XA, YA)
```

Where X(1)= X MIN

X(2)=X MAX

Y(1)=Y MIN

Y(2)=Y MIN

GPL is the subroutine to draw a POLYLINE

8.12 Summary

1. **Multimedia** is a set of more than one media element used to produce a concrete and more structured way of communication.
2. **Multimedia** is nothing but the processing and presentation of information in a more structured and understandable manner using more than one media such as text, graphics, animation, audio and video.

3. If the sequence and timing of these media elements can be controlled by the user, then one can name it as ***Interactive Multimedia***.
4. A **hypertext** system consists of *nodes* - which contain the text - and *links* between the nodes, which define the paths the user can follow to access the text in non-sequential ways. The links represent associations of meaning and can be thought of as cross-references.
5. **Hypermedia** represents an evolution of the hypertext concept. In hypermedia systems the nodes within the web may each contain one or more types of data, from simple text to audio and video clips.
6. Text, graphics, animation, audio and video are the **elements of multimedia**.
7. Musical Instrument Digitisation Interface or **MIDI** provides a protocol or a set of rules, using which the details of a musical note from an instrument is communicated to the computer.
8. **Multimedia Authoring Tools** are the tools for the development of multimedia applications represent the essential part of the system for organization and arrangement of multimedia project elements, such as graphics, sound, animation and video clips. They are used for designing interactively and user interface, for presenting the project on the screen and for synthesizing of multimedia elements into cohesive project.
9. GKS represents a standard graphic interface with consistent syntax
10. The graphic kernel system is based on four basic primitive: POLYLINE, POLYMAKER,FILL AREA, and TEXT
11. GKS WINDOW command is as follows:
12. SET WINDOW (n, x_1, x_2, , y_1, y_2,)

Where n = nth window

(x_1 , y_1) = value of lower left corner of window in user coordinates

(x_2 , y_2) = value of upper right of window in user coordinates

13. The Programmers Hierarchical Interactive Graphics System, an internationally agreed standard for three-dimensional graphics
14. GKS clipping function can be set to either of two states, CLIP or NOCLIP

15. The graphics kernel system allows multiple users to work on a single application.

8.13 Key Words

Multimedia: Multimedia refers to the integration of different types of media, such as text, audio, video, images, and animations, into a single digital presentation or application. It encompasses various forms of media content that can be experienced simultaneously or sequentially.

Multimedia Authoring Tools: Multimedia authoring tools are software applications that enable the creation of interactive multimedia projects. These tools provide a range of features and functionalities to design, develop, and publish multimedia content, including graphics, audio, video, animations, and interactivity. Some popular multimedia authoring tools are Adobe Animate, Adobe After Effects etc.

Multimedia Components: Multimedia components refer to the different types of media elements used in multimedia projects to convey information or enhance the user experience. Text, graphics, audio, video are some example of multimedia components.

8.14 Self Assessment Questions

1. Define the concept of multimedia?
2. Define Hypermedia and how it is different from Hypertext?
3. What are the various elements of multimedia? Define.
4. What is the mechanism of digitized sound? How does the computer reconstruct sound wave from a sample data?
5. Explain the concept of video on multimedia.
6. What common programming language is GKS based on?
7. What do the elements in the POLYLINE X and Y arrays represent?
8. What do the elements in the X and Y arrays represent in the POLYMARKER command?
9. Assuming that X= (10, 20, 20, 10) and Y=(10, 10, 20, 20), the commands

   SET FILL AREA INDEX (0)

FILL AREA (4, X, Y)

10. Will generate a square with only four instead of five elements in the X and y ARRAYS. WHY?
11. What are some of the types of attributes that can be changed for (a) FILL AREA, (b) POLYMAKER, and (c) POLYLINE?
12. What are some of the types of attributes that can be changed for TEXT?
13. What would a display look like after the following commands?

    SET CHARACTER UP VECTOR (1, -1)

    TEXT (10, 30, “HELLO”)
14. How would the display appear after the following commands were executed?

    SET TEXT PATH (DOWN)

    TEXT (8, 8, “HELLO”)

8.15 References/Suggested Readings

1. Computer Graphics (C Version), Donald Hearn and M. Pauline Baker, Prentice Hall,
2. Computer Graphics , Second Edition , by Pradeep K. Bhatia , I.K .International Publisher.
3. Advanced Animation and Rendering Techniques, Theory and Practice, Alan Watt and Mark Watt , ACM Press/Addison-Wesley
4. Graphics Gems I-V, various authors, Academic Press
5. Computer Graphics, Plastok, TMH
6. Principles of Interactive Computer Graphics, Newman, TMH