COMPUTER GRAPHICS

As per the New NEP Syllabus for BCA 5th Semester Course of
Bengaluru City University and Bangalore University

Authored By

Dr. Aruna Devi. C

MCA, M.Phil, Ph.D

Associate Professor and HOD -BCA

Department of Computer Applications

Dayananda Sagar College of Arts, Science and Commerce

Bengaluru

Skyward Publishers

#157, 7th Cross, 3rd Main Road, Chamarajpet
Bangalore-18. Phone : 080-26603535 / 43706620,
Mob: 9611185999
E-mail: skyward.publishers@gmail.com
Website: www.skywardpublishers.co.in

A Text Book of "Computer Grphics" - As per the New NEP Syllabus for 5th Semester BCA Course of Bengaluru City University & Bangalore University by **Aruna Devi C.**

ISBN : 978-93-95085-95-3

First Edition : 2023

Price : ₹ 150/-

Published by:

Skyward Publishers
#157, 7th Cross, 3rd Main Road, Chamarajpet,
Bengaluru-18. Phone : 080-26603535 / 080-43706620,
Mob: 9611185999
E-mail: skyward.publishers@gmail.com
Website: www.skywardpublishers.co.in

Preface

This text book has been written for Computer Science Course. This text book is organized in such a way that it covers the concept of Computer Graphics.

As Computer Graphics remains one of the most exciting and rapidly growing in computer fields. It is known commonly used in various applications. In this book students can get a basic understanding of graphics and learn how images are drawn using programming language. This book helps the students to learn all the topics from one text book where it covers the all the concepts.

Chapter 1 deals with Introduction to Computer Graphics then the applications of Computer graphics and also about various display devices. Chapter 2 deals with the concept of displaying the 2D objects using algorithms and their attributes. Chapter 3 deals with two dimensional transformations. Chapter 4 deals with viewing transformation and the concept of clipping used to clip an object. Chapter 5 deals with 3D coordinate system and their transformations with different display techniques. Chapter 6 deals with segments concepts. Chapter 7 deals with interactive picture construction techniques. Chapter 8 deals with the different graphical Input devices.

- Author

Acknowledgement

I firstly thank, God for blessing me to complete this work successfully. The preparation of this book would not been possible without the help of many people.

My sincere thanks to the Management, Principal, Vice Principal and all my colleagues of Department of Computer Applications, Dayananda Sagar College of Arts, Science and Commerce, for rendering all help and co-operation necessary for completion of this book.

I wish to express my special thanks to Mr. Srikanth and Skyward Publishers team for providing this opportunity to publishing this book.

My heart felt gratitude to my parents and family members for their support, patience, understanding and motivation during the preparation of this book.

Your valuable suggestions and comments for further improvement of the book can be sent to arunat04@gmail.com

- Dr. Aruna Devi C

Syllabus

Total: 48 Hrs

Unit - I : Graphics Systems and Output Primitives [12 Hours]

Application of computer graphics; Graphic software; Video display devices- Raster scan and random scan displays; CRT functioning - Factors affecting CRT; Raster scan system; Color CRT monitors - Display processor with raster system; Raster co-ordinate system; Color mapping - Instruction set and raster system applications;

Line drawing methods-Direct, DDA and Bresenhams, line attributes, Circle drawing – Direct and midpoint circle drawing – ellipse drawing-Bresenhams ellipse algorithm-Area fillingscanline area filling and character attributes

Unit - II : 2D-Transformation , Windowing and Clipping [12 Hours]

Geometric transformation; Translation; Rotation; Scaling; Reflection and shear matrix representations; Homogeneous co-ordinates - Composite transformation - Raster methods for geometric transformations;

Window and viewport; Clipping process - Point clipping, Line clipping, Text clipping, Line clipping techniques - Cohen Sutherland line clipping algorithm, Midpoint subdivision algorithm; Area clipping - Sutherland and Hodgman polygon clipping algorithm, Window to view port transformation

Unit - III : 3D Graphics [12 Hours]

3D-Coordinate system; 3D-Display techniques; parallel projections, Perspective projections, Orthogonal projections: 3D-Transformations; Translation, Scaling, Rotation, Reflection; polygon surfaces, polygon tables; Octrees; Hidden surface removal; Depth buffer and scan line method Introduction to segments, functions for segmenting, display file, segment attributes, display file compilation;

Unit - IV : Graphical Input Devices and Techniques [12 Hours]

Input Devices: Keyboard, Mouse, Joystick, Touch panels, Track ball, Light pen, Graphic tablets. Positioning techniques, Grid, Constraints, Dynamic manipulation, Gravity field, Rubber band, Dragging, Selection technique, Menu, Pointing and selection by naming. Tablet; Data glove; Digitizers; Voice systems.

CONTENTS

UNIT 1

GRAPHICS SYSTEMS AND OUTPUT PRIMITIVES

CONTENTS

1.1 Introduction

Computer Graphics is a picture that is generated by a computer. Computer Graphics is the discipline of producing picture or images using a computer. This include *modeling* - creation, manipulation, and storage of geometric objects and *rendering* – converting a scene to an image, or the process of transformations, shading, illumination, and animation of the image.

Computer imagery is found on television, in newspapers, for example in weather reports, or in all kinds of medical investigation and surgical procedures. The development of computer graphics has made computers easier to interact with, and better for understanding and interpreting many types of data. Today, we find computer graphics used routinely in the areas as science, engineering, medicine, business, industry, government, art, entertainment, advertising, education, and training.

1.1.1 History of Computer Graphics

- In the 1950's, the first computer driven display was used to generate simple pictures. This display used *Cathode Ray Tube* (CRT). Using dark and light characters, a picture can be reproduced. 1950: Ben Laposky created the first graphic images, an Oscilloscope, generated by an electronic (analog) machine. The image was produced by manipulating electronic beams and recording them onto high-speed film.
- In the 1960's, beginnings of modern *interactive graphics*, output are vector graphics and interactive graphics. One of the worst problems was the cost and inaccessibility of machines. 1960: William Fetter coins the computer graphics to describe new design methods.
- In the early 1970's, output start using *raster displays*, graphics capability was still fairly chunky.
- In the 1980's output are built-in raster graphics, bitmap image and pixel. Personal computers costs decrease drastically; trackball and mouse become the standard interactive devices.
- In the 1990's, since the introduction of *VGA and SVGA*, personal computer could easily display photo-realistic images and movies. 3D image renderings are become the main advances and it stimulated cinematic graphics applications.

1.2 Applications of Computer Graphics

Some of the main applications of computer graphic are given below:

1. Computer-aided design
2. Presentation Graphics
3. Computer Art
4. Entertainment
5. Educational and Training
6. Information Visualization
7. Image Processing
8. Information graphics
9. Virtual Reality

1. Computer Aided Design

Computer Aided Design (CAD) is a type of computer based tool used for *drafting and designing*. CAD is useful in various designing fields such as architecture, mechanical and electrical, automobiles, aircraft, spacecraft, computers, textiles and many more.

This is a type of software, which enables users to create rapid and precise drawings and rough sketch plans of main products.

To design the applications, objects are first displayed in a *wire frame* outline so that the overall shape and internal features of objects can be seen. [Fig. 1.1]

- Wire frame display allows the designers to quickly see the effects of interactive adjustments.

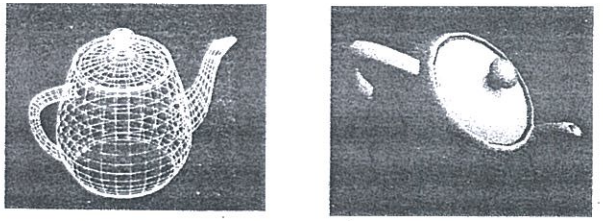


Figure 1.1 Wire-frame Model of a Teapot

- Software packages for CAD applications typically provide the designer with multi window environment, ie., different view of objects.
- In the mechanical field, it is used for designing various machinery and tools that are useful for manufacturing purposes.
- In the field of electronics, it is used in manufacturing process planning, digital circuit design, and other software applications.
- In the field of architecture, it is used for designing all types of buildings. It enables them to design buildings in 2D and 3D models to give almost a real replica of the original work. Architects use interactive graphics methods to layout floor plan, that show the positioning of rooms, doors, windows, stairs, shelves and other building features.
- It is useful in engineering processes in conceptual design, and laying out and analyzing components in manufacturing methods.

2. Presentation Graphics

Presentation graphics, used to produce illustrations for *reports* or to generate 35-mm slides or transparencies for use with projectors. It is used to summarize financial, statistical, mathematical, scientific, data for research reports and other types of reports.

Typical examples of presentation graphics are bar charts, line graphs, surface graphs, pie charts, and other displays showing relationships between multiple parameters.

Figure 1.2 gives examples of two-dimensional graphics combined with geographical information. This illustration shows bar charts combined onto one *graph* and a *pie chart* with three sections. Similar graphs and charts can be displayed in three dimensions to provide additional information.

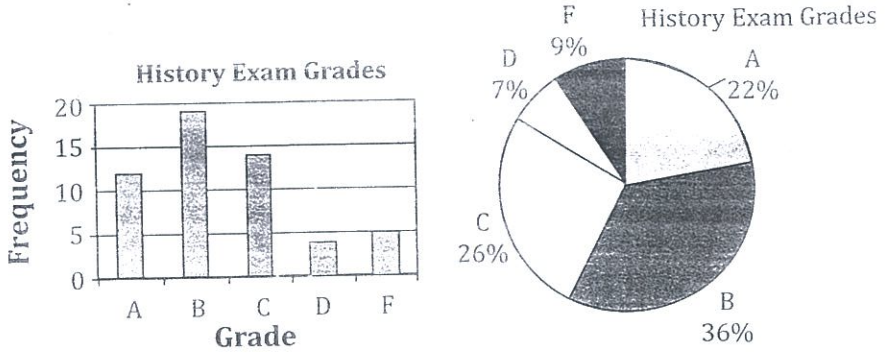


Figure 1.2 Two Dimensional Bar Chart and Pie Chart

3. Computer Art

Computer graphics methods are widely used in both fine art and commercial art applications.

Artists use a variety of computer methods, including special-purpose hardware, artist's paintbrush programs specially developed software, symbolic mathematics packages. CAD, desktop publishing Software, and animation packages that provide facilities for designing object shapes and specifying object motions.

A ***paintbrush*** program that allows artists to "paint" pictures on the screen of a video monitor.

Actually, the picture is usually painted electronically on a graphics tablet (digitizer) using a stylus, which can simulate different brush strokes, brush widths, and colors.

Fine artists use a variety of other computer technologies to produce images.

The artist uses a combination of three-dimensional modeling packages, texture mapping, drawing programs, and CAD software.

Animations are also used frequently in advertising, and television commercials are produced frame by frame, where each frame of the motion is rendered and saved as an image file. The motion of each frame is simulated by moving object positions slightly from their positions in the previous frames. Then the frames are transferred to film. Film animation requires 24 frames for each second in the animation sequence. To playback the animation on a video monitor, 30 frames per second are required.

Morphing is another application where one object is transformed into another. This method has been used in TV commercials for example, To turn an oil can into automobile engine, an automobile into a tiger.

4. Entertainment

Computer graphics methods are now commonly used in making motion pictures, music videos, and television shows.

Computer graphics methods are used regularly in many movies, TV series for generating some graphics scene. Some times the graphics object are combined with the actors and live scenes

Music videos use graphics in several ways. Graphics objects can be combined with the live action, or graphics and image processing techniques can be used to produce a transformation of one person or object into another (morphing). Animation is used frequently in advertising, and television commercials.

5. Educational and Training

Computer graphics techniques is used in education and training. Computer generated models of physical, financial, and economic systems are often used, which can help trainees to understand the operation of the system.

Some examples of simulators used are:

- Flight simulator.
- Automobile driving simulator
- Space Shuttle simulator
- Military Tanker simulator

For training, special systems are designed. Like the simulators for practice sessions or training of ship captains, aircraft pilots, heavy-equipment operators, and air traffic control personnel. Some simulators have no video screens,

For example:

- A flight simulator with only a control panel for instrument flying. But most simulators provide graphics screens for visual operation.
- Automobile-driving simulator is the simulator used to investigate the behavior of drivers in critical situations. The drivers' reactions are then used as a basis for optimizing vehicle design to maximize traffic safety.

6. Information Visualization

Information visualization is the interdisciplinary study of "the visual representation of large-scale collections of non-numerical information, such as files and lines of code in software systems, library and bibliographic databases, networks of relations on the internet.

It is increasingly applied as a critical component in scientific research, digital libraries, data mining, financial data analysis, market studies, manufacturing production control, and drug discovery.

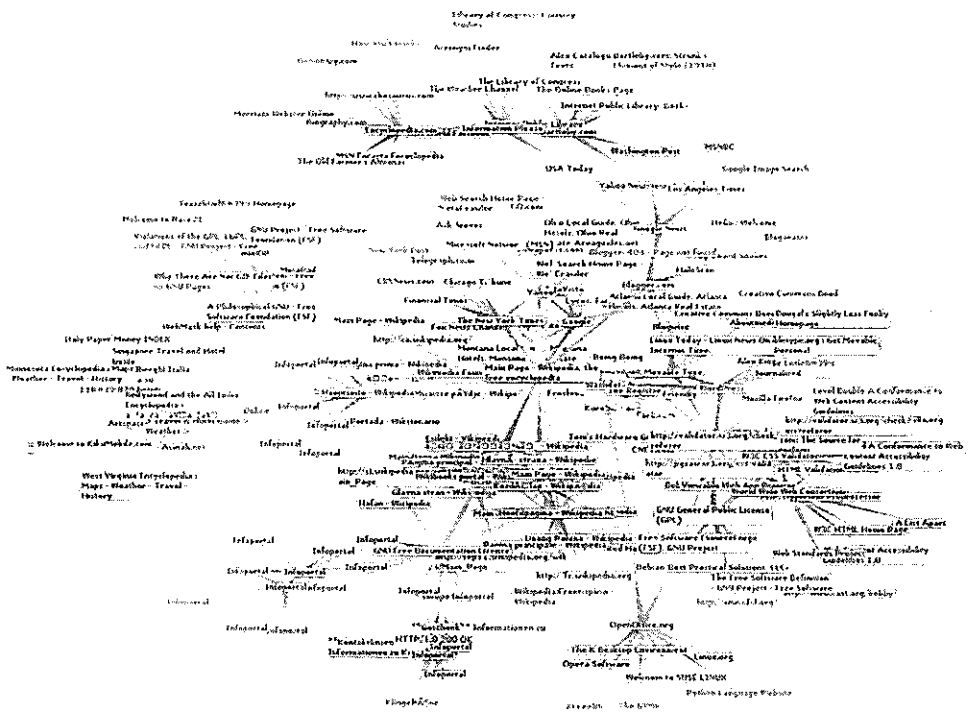


Figure:1.3 Graphic representation of a minute fraction of the WWW, demonstrating hyperlinks

Information visualization presumes that "visual representations and interaction techniques take advantage of the human eye's broad bandwidth pathway into the mind to allow users to see, explore, and understand large amounts of information at once. Information visualization focused on the creation of approaches for conveying abstract information in intuitive ways.

7. Image Processing

Computer graphics and Image processing are fundamentally different operations. Computer graphics is used to create a picture. Image processing, on the other hand applies techniques to modify or interpret existing pictures.

Two principal applications of image processing are:

- Improving picture quality
- Machine perception of visual information, as used in robotics.

In image processing methods, we first digitize a photograph or other picture into an image file. Then digital methods can be applied to rearrange picture parts, to enhance color separations, or to improve the quality of shading.

Medical applications also make extensive use of image processing techniques for picture enhancements. Image processing and computer graphics are typically combined in many applications.

Medicine, for example, uses these techniques to model and study physical functions, to design artificial limbs, and to plan and practice surgery.

8. Information Graphics

Information graphics or infographics are graphic visual representations of information, data or knowledge. These graphics present complex information quickly and clearly, such as in signs, maps, journalism, technical writing, and education. With an information graphic, computer scientists, mathematicians, and statisticians develop and communicate concepts using a single symbol to process information.

In newspapers, infographics are commonly used to show the weather, as well as maps Fig. 1.4 and site plans for newsworthy events, and graphs for statistical data.

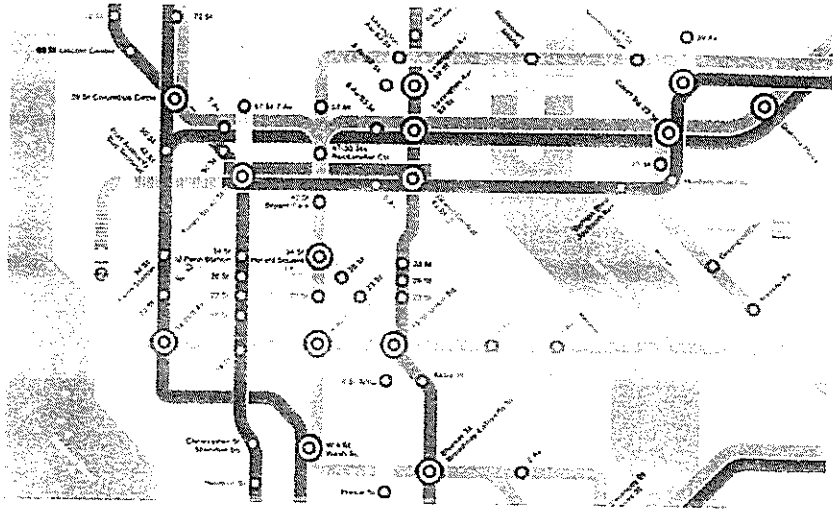


Figure 1.4 A Subway Map

Traffic signs and other public signs rely heavily on information graphics, such as stylized human figures, icons and emblems to represent concepts such as yield, caution, and the direction of traffic. Public places such as transit terminals usually have some sort of integrated "signage system" with standardized icons and stylized maps.

9. Virtual Reality

Virtual reality (VR) is a term that applies to computer-simulated environments that can simulate places in the real world, as well as in imaginary worlds. Most current virtual reality environments are primarily visual experiences, displayed either on a computer screen or through special stereoscopic displays, but some simulations include additional sensory information, such as sound through speakers or headphones. The simulated environment can be similar to the real world—for example, in simulations for pilot.

Figure 1.5 Classic Virtual Reality HMD with Glove

1.3 Computer Graphics Classifications

The computer graphics can be categorized into different variety of ways. These are the following:

- Type of object (dimensionality).
- Type of interaction.
- Role of the picture.
- Relationship between objects and their pictures.

- **Type of Object:** The object can be represented graphically as *abstract or real*. The picture range of possible combination can be 2-D and 3-D type.
  - 2-D type of object can be line, gray scale image.
  - 3-D type of object can be line drawing (or wireframe), line drawing with various effects, shading, color image with various effects.
- **Type of Interaction:** It determines the user's *degree of control* over the object and its image.

  The range here includes:
  - Offline plotting
  - Interactive plotting

  Offline plotting, where a predefined database produced by other application programs.

  Interactive plotting, where the user controls the supply of parameters; real-time animation for flight simulators and interactive designing where the user starts with black screen, defines new objects and gets the desired view.
- **Role of the Picture:** It is the degree to which the picture is an end in itself or is a means to an end. For example drafting, raster painting, animation, and artwork, the drawing is the end product. How ever in CAD applications, the drawing is just a representation of the object being designed or analyzed.

- **Relationship between Objects and their Pictures:** This category arises from the logical and temporal relationship between objects and their pictures. For example, the user may deal with only one picture at a time (plotting), or with sequence of related pictures, or with a collection of objects (as in CAD applications)

1.4 Graphics Software

In computer graphics, graphics software is a program or collection of programs that enable a person to manipulate visual images on a computer.

There are two general classifications for graphics software.

- General programming packages.
- Special purpose application packages.

General graphics programming packages are used in high-level programming language, such as C or FORTRAN. They use a set of graphics functions to generate picture components like Straight lines, polygons, circles and other figures.

Application graphics packages are designed for nonprogrammers, so that users can generate displays without worrying about how graphics operation works. Example, artist's painting programs and CAD.

Some of the graphics software:

- ✓ Photoshop
- ✓ Paint Shop Pro
- ✓ The GIMP
- ✓ Digital Image Suite
- ✓ Illustrator
- ✓ Corel DRAW
- ✓ Microsoft Paint
- ✓ Picasa and many more.

Graphics Software used for?

Graphics software is used in many facets of life and business. Some of the common things people use graphics software for editing and sharing digital photos, creating logos, drawing and modifying clip art, creating digital fine art, creating Web graphics, designing advertisements and product packaging, touching up scanned photos, and drawing maps or other diagrams.

1.5 How Interactive graphics display works

Interactive graphics display consists of three components:

- Frame Buffer or digital memory.
- Display Controller or video controller.
- Television Monitor.

- **Frame Buffer**

  In the frame buffer the displayed image is stored as a *matrix* of intensity values.
- **Display Controller:**

  A special purpose processor, called display controller is used to *control the operation of the display device*. The display controller is used to passes the contents of the frame buffer to the

monitor. The image must be passed repeatedly to the monitor 30 or more times a second, in order to maintain a steady picture on the screen.

In the frame buffer the image is stored as a pattern of binary digital numbers, which represent a rectangular array of pixels. A pixel is the smallest addressable portion of an image. In a black and white image, we can represent black pixel as 1's and white pixel as 0's in the frame buffer.

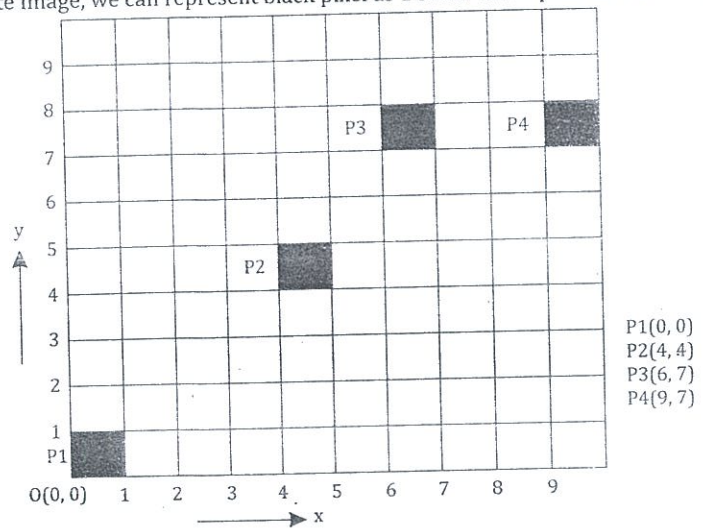


Figure 1.6 Grid of Pixels

Frame buffer is the portion of video card memory that holds the information necessary to display a single screen image. The size of the frame buffer determines the resolution and maximum colors it can display. Frame buffer is a two dimensional table where each row-column stores information about brightness and color values of the corresponding pixel in the screen. In a frame buffer each pixel can be represented by 1 to 24 bits.

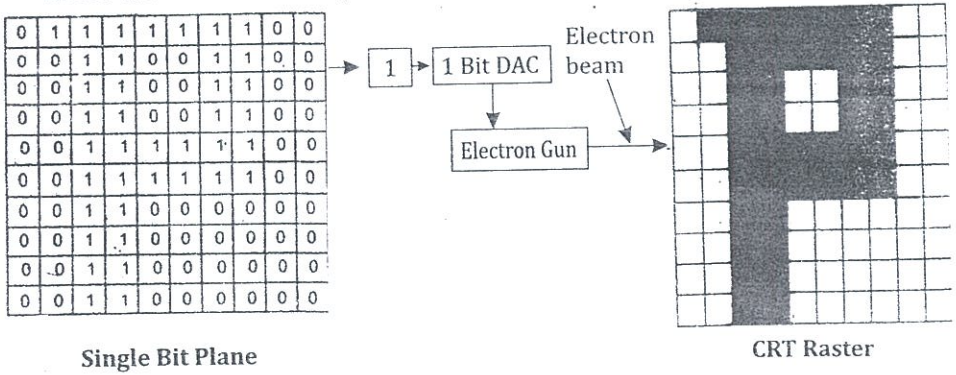


Figure 1.7 Display Device with Single Bit Plane Frame Buffer

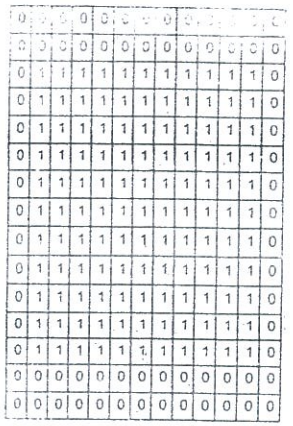


Figure 1.8 Frame Buffer of a Rectangle

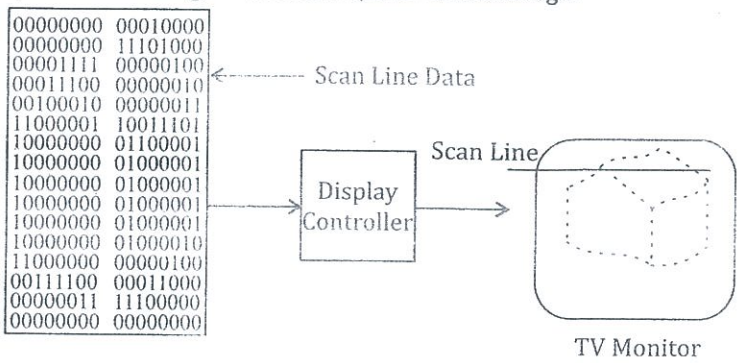


Figure 1.9 The frame Buffer Display.

The display controller reads the byte of data from the frame buffer and converts its 0's and 1's into corresponding video signal. Then the signal is transferred to the TV monitor, producing a black and white pattern on the screen.

Suppose we wish to change the displayed picture. We just need to modify the frame buffer's contents to represent the new pattern of pixels.

Two problems in drawing curved and straight lines on a graphic display are:

- ✓ Choice of which pixel should be black and which pixel is white.
- ✓ Slanting lines and curves in the image will be far from smooth and will instead show unpleasant *"staircase"* effect.

The first problem is solved by using a *procedure, or algorithm*, that computes which pixel should be black from the equation of the line or curve.

The second problem ie., the staircase effect in the picture is solved by using a different sort of display, called a *line drawing display*, which plots continuous lines and curves rather than separate pixels.

Speed is important in displaying pictures. Any display based on the CRT must be refreshed at least 60 times a second.

The image must be transmitted to the display point by point. The longer it takes to transmit each element of the picture the fewer elements can be transmitted and the less information can be displayed.

Until the entire image is transmitted, it will begin flickering. Pictures can be made to Grow, Shrink, Rotate by applying the transformations, based on mathematical techniques: coordinate geometry, trigonometry, and matrix methods.

To create pictures directly on the display screen, a number of different input devices - light pen, tablet, and mouse have been invented to make this kind of interaction more convenient.

1.6 Video Display Devices

1.6.1 Raster Scan Display

Graphics monitor using a CRT is the raster scan display, based on television technology.

Method

- The *electron beam is swept* across the screen, one row at a time and from top to bottom. As the electron beam moves across each row, the beam intensity is turned on and off dependent on information defining the picture to be created. Fig. 1.10.
- 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.
- Each screen point is referred to as a *pixel or pel*. Intensity range for pixel positions depends on the capability of the raster system.
- A bit value of 1 indicates that the electron beam is to be turn on at that position, and a value of 0 indicates that the beam intensity is to be off.
- A system with 24 bits per pixel and a screen resolution of 1024 by 1024 requires 3 megabytes of storage for the frame buffer. On a black-and-white system with one bit per pixel, the frame buffer is called a bitmap. For systems with multiple bits per pixel, the frame buffer is often referred to as a *pixmap*.

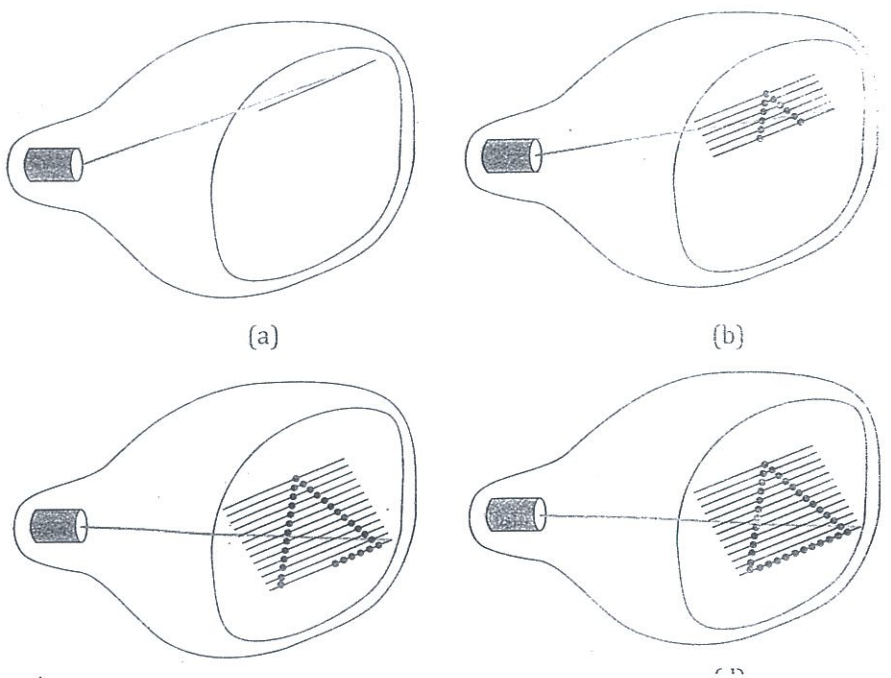


Figure 1.10 Raster Scan Display

Refreshing on raster-scan displays is carried out at the rate of 60 to 80 frames per second, refresh rates are described in units of cycles per second, or Hertz (Hz), where a cycle corresponds to one frame.

At the end of each scan line, the electron beam returns to the left side of the screen to begin displaying the next scan line. The return to the left of the screen, after refreshing each scan line, is called the horizontal retrace of the electron beam. Similarly when the electron beam returns to the top left corner of the screen to begin the next frame is called vertical retrace.

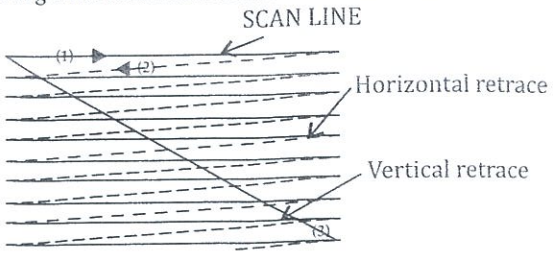


Figure 1.11 Interlacing Scan Lines on a Raster-scan Display

First all points on the even number scan line are displayed, then all points along the odd-numbered lines are displayed.

In raster-scan systems each frame is displayed in two passes using an *interlaced refresh procedure.* The first pass, the beam sweeps across every other scan line from top to bottom. The second pass the vertical retrace, the beam sweeps out the remaining scan lines. Interlacing technique the entire screen is scanned in half the time, it would take to sweep across all the lines, at once, from top to bottom; it is effective method for reducing flickering.

Advantages of Raster-scan Display

- Decreasing memory costs have made raster systems popular.
- High degree realism is achieved in picture with the aid of advanced shading and hidden surface technique.
- Computer monitors and TVs use this method.

Disadvantages of Raster-scan Display

- The lines produced are zigzag as the plotted values are discrete.
- Resolution is low.

1.6.2 Random Scan Display

In a random scan display, 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 are referred to as vector display *(or* stroke-writing or calligraphic displays). Fig. 1.12

Picture definition is now stored as a set of line drawing commands in an area of memory referred to as the refresh display file or refresh buffer.

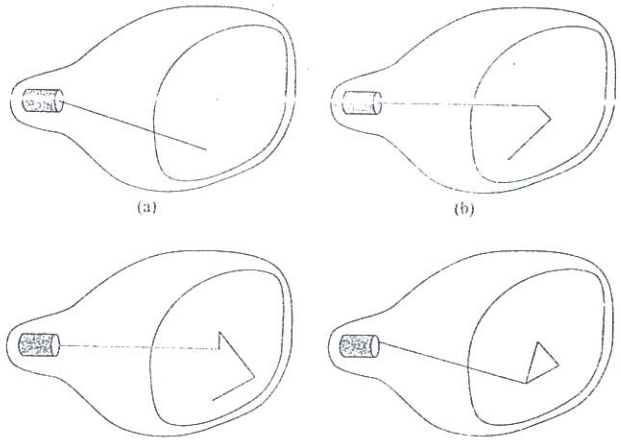


Figure 1.12 Random Scan System

To display a specified picture, the system cycles through the set of commands in the display file drawing each component line in turn. After all line drawing commands have been processed, the system cycles back to the first line command in the list. Random-scan displays are designed to draw all the component lines of a picture 30 to 60 times each second. When a small set of lines is to be displayed, each refresh cycle is delayed to avoid refresh rates greater than 60 frames per second. Faster refreshing of the set of lines could burn out the phosphor.

Random-scan systems are designed for line drawing applications and cannot display realistic shaded scenes.

Advantages of Random Scan Display

- High resolution.
- Produce smooth line drawings.

Disadvantages of Random Scan Display

- Costlier.
- Difficult to display realistic scenes.
- Complex scene cause visible flicker.

Difference between Raster Scan and Random Scan Display

Sl. No	Raster Scan	Random Scan
1.	In Raster scan display the electron beam is swept across the screen, one row at a time from top to bottom.	In Random scan display the electron beam directed only to the parts of the screen where picture is to be drawn.
2.	Picture definition is stored in a memory area called refresh buffer (or) frame buffer. It holds the set of intensity values for all screen points.	Picture definition is stored in a memory area called refresh buffer. It stores as a set of line drawing commands.
3.	Raster displays have less resolution.	Random displays have high resolutions since the picture definition is stored as a set of line drawing commands and not as a set of intensity values.
4.	The lines produced are zig zag as the plotted values are discrete.	Smooth lines are produced as the electron beam directly follows the line path.
5.	High degree realism is achieved in picture with the aid of advanced shading and hidden surface technique.	Realism is difficult to achieve.

6.	Decreasing memory costs have made raster systems popular.	Random scan systems are generally costlier.

1.7 Cathode Ray Tube (CRT)

A cathode ray tube (CRT) is a vacuum tube in which images are produced when an electron beam strikes a phosphor-coated surface.

The electron gun generates a beam of electrons, which passes through focusing and deflection system that direct the beam toward specified positions on the phosphor-coated screen.

The phosphor then emits a small spot of light at each position contacted by the electron beam, 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*

..7.1 CRT Functioning

A cathode ray tube consists of several basic components:

- Heating Metal (Filament)
- Cathode
- Control Grid

When electricity is supplied, the filament heats up and a stream or "ray" of electrons pours off the element into the vacuum.

The negatively charged electrons are attracted to positively charged anodes which focus the particles into beams, accelerating them to strike the phosphor-coated screen. Phosphor will glow when the electron beam touches the screen.

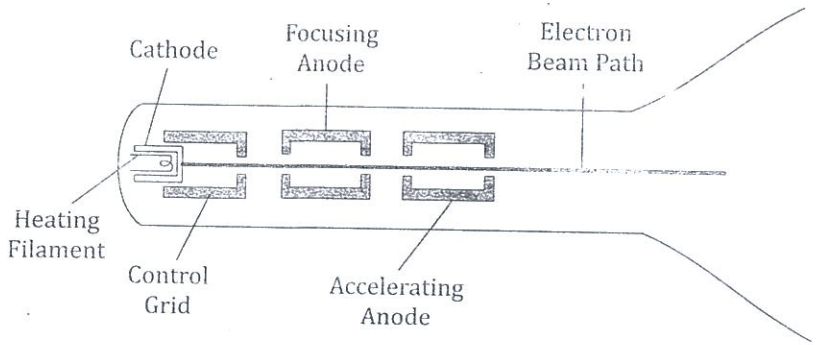


Figure 1.13 Operation of an Electron Gun

Intensity of the electron beam is controlled by setting voltage levels on the *control grid*.

- A high negative voltage applied to the control grid will shut off the beam.

- A smaller negative voltage on the control grid decreases the number of electrons passing through.
- The brightness of the picture is controlled by the control grid.

Focusing System

The electron beam is forced to converge into a small spot on the phosphor coated screen. Otherwise the electron will repel each other. Focusing is done with the help of a focusing structure containing metal plates at different potentials.

Acceleration Anode

It consists of two metal plates mounted perpendicular to the beam axis with holes at their centers through which the beam pass. It is maintained at sufficient high relative potential to accelerate the beam to necessary velocity.

Deflection System

Deflection system is to change direction. It deflects information rapidly so that it can display clearly on screen without flickering.

The deflection system contains a set of coils mounted at the neck of the tube. There are two sets of deflecting plates: *horizontal* and *vertical*.

The horizontal deflection plates controls the vertical deflection. The vertical deflection plates controls horizontal deflection.

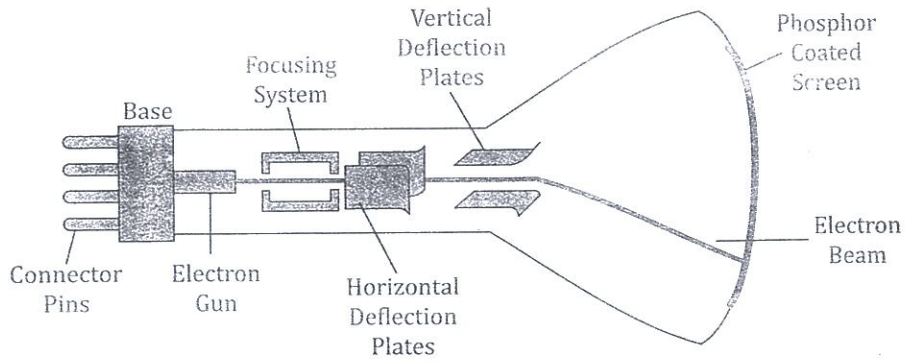


Figure 1.14 Electrostatic Deflection of the Electron Beam in a CRT

Phosphors

The phosphors used in graphic display are chosen for color and persistence.

The properties are:

- ✓ The persistence which is the time it takes to emit light from the screen to decay to 1/10th of its original intensity.
- ✓ Color should be white, especially for application where dark colored information appears on light background.
- ✓ High efficiency in terms of electric energy converted to light.
- ✓ Resistance to burning under prolonged execution.

To improve performance many different phosphors have been produced, using various compounds of calcium, cadmium, and zinc together with traces of rare - earth elements.

1.7.2 Factors Affecting CRT

The important factor is the quality of the image in computer graphics. The Quality of image depends on the following:

- Resolution
- Addressability.
- Persistence.
- Aspect Ratio.

Resolution

- Resolution is the number of pointes per inch or centimeter that can be plotted horizontally & vertically.
- It is nothing but the clarity and sharpness of the picture.
- The smaller the spot size, the higher the resolution.
- The higher the resolution, the better is the graphics system.
- High quality resolution is 1280 × 1024

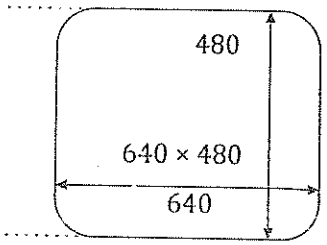


Figure 1.15 Screen with 640 × 480 Resolution

Size	Resolution
13-inch	640 × 480
15-inch	800 × 600
17-inch	1024 × 768
21-inch	1024 × 768

Table

Persistence

Persistence is defined as the time it takes to emit light from the screen to decay to 1/10th of its original intensity. How long small spots continue to emit light after the beam is moved.

- Lower persistence requires high refresh rate & it is good for animation.
- High persistence is useful for displaying highly complex static picture.
- Graphics monitors are usually constructed with 10 to 60 microseconds.

Addressability

- Addressability is a measure of the spacing between the centers of vertical and horizontal lines.
- The picture on a screen consists of intensified points.
- The smallest addressable point on the screen is called pixel or picture element.
- In graphics mode there are 800×600 pixels.

Aspect Ratio

Aspect ratio gives the ratio between vertical points and horizontal points necessary to produce equal length lines in both directions on the screen.

Aspect ratio = ¾ means: vertical line with 3 points is equal in length to horizontal line of 4 points.

1.8 Color CRT Monitors

CRT monitors display colour picture by using combination of phosphors that emit different colored light. Colour CRT monitors use two techniques:

- Beam-Penetration method.
- Shadow–Mask method.

1.8.1 Beam-Penetration Method

The beam penetration method for displaying color pictures has been used in random-scan systems. There are *two layers of phosphor*; they are red phosphor and green phosphor which are coated on to the inside of the CRT screen. The multilayered phosphor in which a layer of red phosphor is deposited behind the initial layer of green phosphor.

Penetration of Electron beam

The color depends on how far the electron beam penetrates into the phosphor layers.

- A beam of slow electrons excites only the outer red layer to produce a *red colour*.
- A beam of very fast electron penetrates through the red layer and excites the inner green layer to produce *green colour*.
- At intermediate beam speed a combination of red and green light is emitted to show two different colours, *orange and yellow*.

Disadvantages of Beam-Penetration Method

- Problem with this method is the need to change the beam accelerating potential by significant amount in order to switch colours can be displayed.
- Only 4 colours are possible.
- Quality of picture is not good when it is compared to other techniques.

1.8.2 Shadow-Mask Method

Shadow–Mask method are commonly used in raster scan systems, example Colour TV.

They produce a much wider range of colours 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.

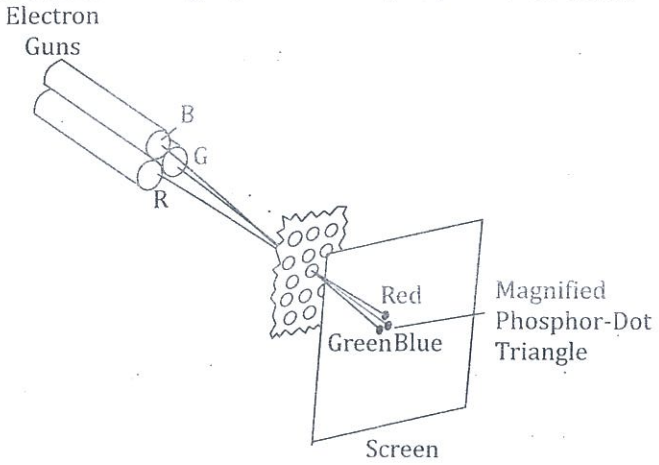


Figure 1.16 Shadow mask CRT

A metal *shadow mask* pierced with small holes, placed behind the phosphor coated screen. 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.

Different colours are displayed by varying the intensity levels of the three electron beams. By turning off the red and green guns, we get only the colour blue. Other combinations of beam intensities produce a small light spot for each pixel position.

The color we see depends on the amount of excitation of the red, green, and blue phosphors.

- A white (or gray) is produced by 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.
- Cyan is produced when blue and green are activated equally.

In some low-cost systems, the electron beam can only be set to on or off, and it can produce eight colors.

More sophisticated systems can set intermediate intensity levels for the electron beams, and it can produce several million different colors.

1.9 Display Processors

It is an interpreter or piece of hardware that convert display processor code into pictures. There are two types of display processor.

- Raster-Scan Display Processor.
- Random-Scan Display Processor.

1.9.1 Raster-Scan Systems

Interactive raster graphics systems typically employ several processing units. In addition to the central processing unit or CPU, a special-purpose processor, called the **video controller or display controller,** is used to control the operation of the display device.

Organization of a Simple Raster System

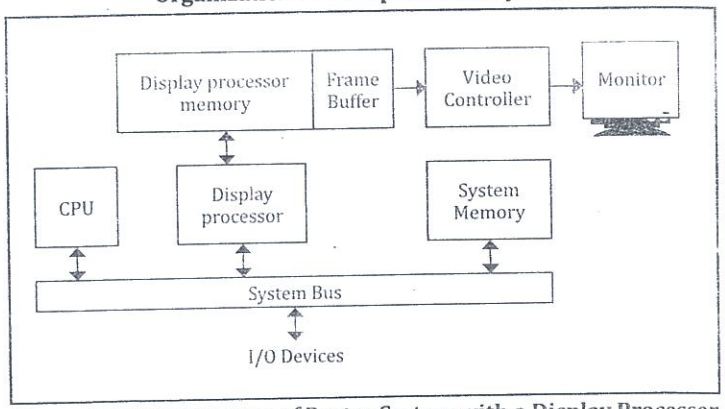


Figure 1.17 Architecture of Raster System with a Display Processor

A Major Task of the Display Processor:

- Digitizing a picture definition in an application program into a set of pixel- intensity values for storage in the frame buffer.
- Digitizing means simply capturing an analog signal in digital form.
- This digitization process is called **scan conversion.**

- The functions generate various line styles, displaying color areas, and performing certain transformations and manipulations on displayed objects.
- A fixed area of the system memory is reserved for the frame buffer, and the video controller is given direct access to the frame-buffer memory.
- Frame-buffer locations and the corresponding screen positions are referenced in Cartesian coordinates.
- Display processors are typically designed to interface with interactive input devices, such as a mouse.

1.9.2 Random-Scan Systems

The organization of a simple random-scan system (sometimes called vector scan system).

- An application program is input and stored in the system memory along with a graphics package.
- Graphics commands in the application program are translated by the graphics package into a display file stored in the system memory.
- This display file is then accessed by the display processor to refresh the screen.
- The display processor is referred as Display processor unit or graphics controller which goes through each command in the display file during every refresh cycle.

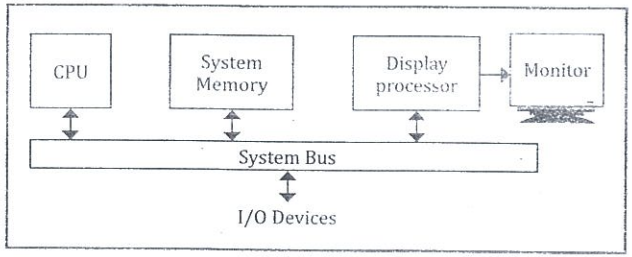


Figure 1.18 Architecture of a Simple Random Scan System

1.10 Color Mapping

A **color model** is an abstract mathematical model describing the way colors can be represented as tuples of numbers, typically as three or four values or color components. A color model is a specification of a 3-D coordinate system.

The different type of color models:

- RGB color model.
- CMY color model.
- YIQ color model.

RGB Color Model

The RGB color model is composed of the primary colors Red, Green, and Blue. This system defines the color model that is used in most color CRT monitors and color raster graphics. The RGB model uses the Cartesian coordinate system.

This model is called ***additive***, and the colors are called primary colors. The color subspace of interest is a cube shown in Fig. 1.19 "RGB Color Model. (RGB values are normalized to 0..1), in which RGB values are at three corners; cyan, magenta, and yellow are the three other corners, black is at their origin; and white is at the corner farthest from the origin. The primary colors can be added to produce the secondary colors

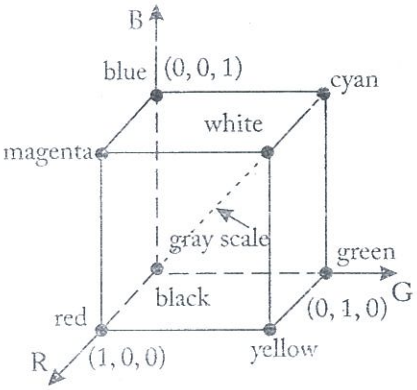
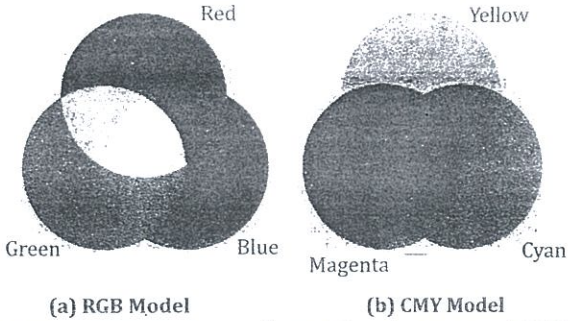


Figure 1.19 RGB Color Model

For example - magenta is red plus blue, cyan is green plus blue, and yellow is red plus green. The combination of red, green, and blue at full intensities makes white.



(a) RGB Model **(b) CMY Model**

Figure 1.20 Primary and Secondary Colors for RGB and CMY Models

The importance of the RGB color model is that it relates very closely to the way that the human eye perceives color. RGB is a basic color model for computer graphics because color displays use red, green, and blue to create the desired color.

CMY Color Model

The CMY color model Fig. 1.21 is a subset of the RGB color model and is primarily used in color print production. CMY color model stands for cyan, magenta, and yellow. The CMY color space is ***subtractive***, meaning that cyan, magenta yellow, and black pigments or inks are applied to a white surface to subtract some color from white surface to create the final color.

For example - Cyan is white minus red, magenta is white minus green, and yellow is white minus blue.

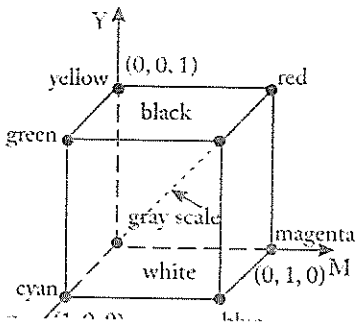


Figure 1.21 CMY Color Model

YIQ Model

This is the color model used by the U.S. Commercial Color Television Broadcasting. It is a recoding of RGB for transmission efficiency and for downward compatibility for black & white television.

The YUV color space is "derived" from the RGB space. It comprises the *luminance* (Y) and two color difference (U, V) components. The luminance can be computed as a weighted sum of red, green and blue components; the color difference, or *chrominance*, components are formed by subtracting luminance from blue and from red.

1.11 Instruction Set

In order to provide a flicker-free display, the display processor must execute its program 30 to 60 times per second because there is no pixmap (pixmap is for multiple-bit-per-pixel system). Since pixel map refers to the contents of the refresh buffer and to the buffer memory itself.

The display processing unit (DPU), or graphics controller has an ***instruction set*** and ***instruction address register***, which goes through the classic instruction fetch, decode, and execute cycle found in any computer.

The program executed by the DPU is in main memory, which is shared by the general CPU and DPU.

The application program and graphics subroutine package also reside in main memory and executed on the general CPU.

The graphics package creates a display program of DPU instructions and tells the DPU where to start the program.

The DPU then asynchronously executes the display program until it is told to stop by the graphics package.

A JUMP instruction at the end of the display program transfers control back to its start, so that the display continues to be refreshed without CPU intervention.

1.12 Other Display Technologies

In refresh line-drawing display based on CRT has some disadvantages

- High cost
- The tendency to flicker when the display picture is complex.

These two problems led to the development of *inherent image storage capability.*

The most widely used display devices are

- Direct View storage Tube (DVST)
- Plasma Panel
- Liquid Crystal Displays (LCDs)

1.12.1 Direct View Storage Tube (DVST)

CRT uses the method of refresh the screen for the image to remain visible. In case of DVST it uses the method to maintaining a screen image, is to store the picture information inside the CRT.

Two electron guns are used in a DVST:

- The Primary Gun - is used to store the picture pattern
- Flood Gun - maintains the picture display

Advantages of Direct View Storage Tube (DVST)

No refreshing is needed; very complex pictures can be displayed at very high resolutions without flicker.

Disadvantages of Direct View Storage Tube (DVST)

- 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.

1.12.2 Flat Panel Display

Flat Panel displays have reduced volume, weight, and power requirements compared to a CRT.

The feature of flat-panel displays is that they are thinner than CRTs as they can be wall mounted. Examples are small TV monitors, calculators, pocket video games, laptop computers, and armrest viewing of movies on airlines, as advertisement boards in elevators.

Flat-panel displays are categories as:

- Emissive displays
- Non-emissive displays

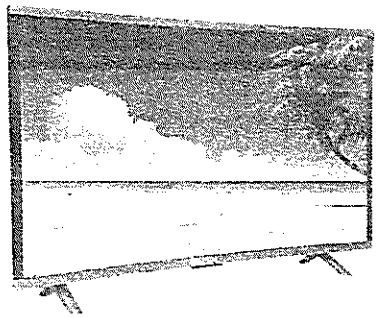


Figure 1.22 Flat screen TV's

The *emissive displays* are devices that convert electrical energy into light. *Plasma Panel* is one example.

The *Non-emissive displays* use optical effects to convert sunlight or light from some other source into graphics patterns. Example is *liquid-crystal device.*

1.12.3 Plasma Panel

A plasma display panel (PDP) is a type of flat panel display common to large TV displays. They are called "plasma" displays because the pixels rely on plasma cells.

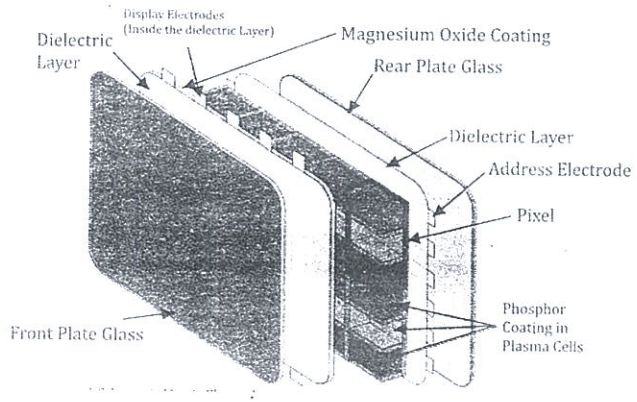


Figure 1.23 Plasma Panel Display

It has millions of tiny cells in between two panels of glass. The cells hold a mixture of noble gases such as neon. The gas in the cells is electrically turned into plasma which then excites phosphors to emit light.

How Plasma Displays Work?

The xenon, neon, and helium gas in a plasma television is contained in hundreds of thousands of tiny cells positioned between two plates of glass. Long electrodes are also put together between the glass plates, in front and behind the cells. The address electrodes sit behind the cells, along the rear glass plate. The transparent display electrodes, which are surrounded by an insulating dielectric material and covered by a magnesium oxide protective layer, are mounted in front of the cell, along the front glass plate. As the gas ions rush to the electrodes and collide, photons are emitted.

In a *monochrome plasma panel*, the ionizing state can be maintained by applying a low-level voltage between all the horizontal and vertical electrodes, even after the ionizing voltage is removed. To erase a cell all voltage is removed from a pair of electrodes. This type of panel has inherent memory and does not use phosphors.

In *color panels*, the back of each cell is coated with a phosphor. The ultraviolet photons emitted by the plasma excite these phosphors to give off colored light.

Every pixel is made up of three separate subpixel cells, each with different colored phosphors. One subpixel has a red light phosphor, one subpixel has a green light phosphor and one subpixel has a blue light phosphor. These colors blend together to create the overall color of the pixel, the same as shadow mask CRT or color LCD. Brightness is controlled by pulse-width modulation. Plasma displays use the same phosphors as CRTs, which accounts for the extremely accurate color reproduction when viewing television or computer video images.

Characteristics of Plasma Panel

- Plasma displays are bright having a wide range of colours.
- They have a very low-luminance "dark-room" black level compared to the lighter grey of the unilluminated parts of an LCD screen.
- Plasma displays use as much power per square meter as a CRT or television.

Advantages of Plasma Panel

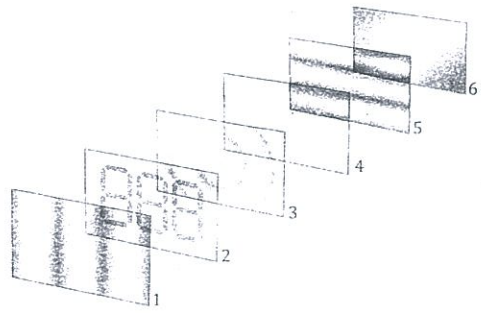
- Slim profile.
- Can be wall mounted.
- Less bulky than rear-projection televisions.
- Produces deep blacks allowing for superior contrast ratio.
- Wider viewing angles than those of LCD; images do not suffer from degradation at high angles unlike LCD's.

Disadvantages of Plasma Panel

- Susceptible to "large area flicker".
- Generally do not come in smaller sizes than 37 inches.
- Susceptible to reflection glare in bright rooms.
- Heavier than LCD due to the requirement of a glass screen to hold the gases.
- Use more electricity, on average, than an LCD TV. redrawn.

1.12.4 Liquid Crystal Display

A liquid crystal display (LCD) is a thin, flat electronic visual display that uses the light modulating properties of liquid crystals. Liquid Crystal do not emit light directly. They are used in a wide range of applications including: computer monitors, television, instrument panels, aircraft cockpit displays, signage, etc. They are more compact, lightweight, portable, less expensive, more reliable, and easier on the eyes.



1. Polarize.
2. Glass substrate with ITO electrodes.
3. Liquid crystals Layer.
4. Glass substrate with common electrode film (ITO) with horizontal ridges to line up with the horizontal filter.
5. Horizontal filter film to block/allow through light.
6. Reflective surface to send light back to viewer.

Figure 1.24 Schematic diagram of Liquid Crystal Display

Each pixel of an LCD typically consists of a layer of molecules aligned between two transparent electrodes, and two polarizing filters, the axes of transmission of which are perpendicular to each other. With no actual liquid crystal between the polarizing filters, light passing through the first filter would be blocked by the second (crossed) polarizer. The surfaces of the electrodes that are in contact with the liquid crystal material are treated so as to align the liquid crystal molecules in a particular direction. Before applying an electric field, the orientation of the liquid crystal molecules is determined by the alignment at the surfaces of electrodes.

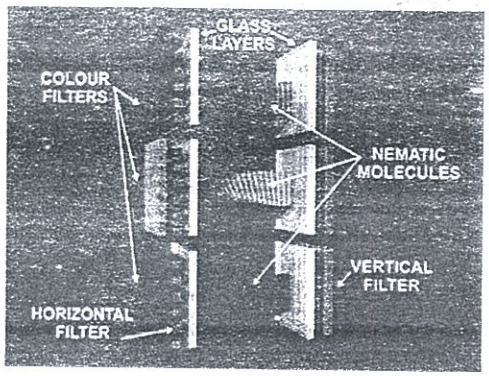


Figure 1.25 Sub Pixels of Color LCD

In colour LCD's each individual pixel is divided into three cells, or subpixels, which are coloured red, green, and blue, respectively, by additional filters. Each subpixel can be controlled independently to yield thousands or millions of possible colours for each pixel. CRT monitors employ a similar 'subpixel' structures *via* phosphors, although the electron beam employed in CRTs do not hit exact *subpixels*.

There are two type of LCD displays:

- Passive-matrix LCD.
- Active-matrix LCD.

Passive Matrix Display

LCD's with a small number of segments, such as those used in digital watches and pocket calculators, have individual electrical contacts for each segment. An external dedicated circuit supplies an electric charge to control each segment. This display structure is unwieldy for more than a few display elements. Each row or column of the display has a single electrical circuit. The pixels are addressed one at a time by row and column addresses. This type of display is called *passive-matrix addressed* because the pixel must retain its state between refreshes without the benefit of a steady electrical charge.

Active Matrix Display

Another method for constructing LCD's is to place a transistor at each pixel location, using thin-film transistor technology. The transistors are used to control the voltage at the pixel locations and to prevent charge from gradually leaking out of the liquid-crystal cells. These devices are called active-matrix displays.

Difference between CRT and LCD Monitors

- CRTs are big and bulky while LCD's are thin and light
- CRTs consume more power compared to LCD's
- A byproduct of the power consumption, CRTs also get much hotter compared to LCD's
- LCD's have a greater response time than CRTs
- LCD's are considerably more expensive than CRTs
- LCD's have more applications than CRTs
- LCD's can suffer from dead pixels while CRTs do not

1.13 Output Primitives

A picture is completely specified by the set of *intensities* for the pixel positions in the display. To construct a useful picture on a point plotting display we must build the picture out of many hundred of pixels. Lines and curves must be drawn with closely spaced pixels. Point plotting did not use frame buffers instead where fed with a stream of point coordinates by the computer. But the flickering was not avoided.

Graphics programming packages provide functions to describe a scene in terms of basic geometric structures, referred to as output primitives. Points and straight line segments are the simplest geometric components of pictures. Additional output primitives that can be used to construct a picture include circles and other conic sections, quadric surfaces, spline curves and surfaces, polygon color areas, and character strings.

1.14 Points and Lines

Point plotting is a single coordinate position. In a CRT monitor, for example, the electron beam is turned on to illuminate the screen phosphor at the selected location. Line drawing is accomplished by calculating intermediate positions along the line path between two specified endpoint positions. An output device is then directed to fill in these positions between the endpoints.

For *analog devices*, such as a vector pen plotter or a random-scan display, a straight line can be drawn smoothly from one endpoint to the other. *Digital devices* display a straight line segment by plotting discrete points between the two endpoints. Discrete coordinate positions along the line path are calculated from the equation of the line.

Example

For a raster video display, the line color (intensity) is loaded into the frame buffer at the corresponding pixel coordinates. Reading from the frame buffer, the video controller then "plots" the screen pixels. A computed line position of (10.48, 20.51), for example, would be converted to pixel position (10, 21). Thus rounding of coordinate values to integers causes lines to be displayed with a *stair step appearance ("the jaggies")*.

Figure 1.26 Stair Step effect produced when a Line is Generated.

Pixel positions are referred to scan line no. and column number. To load an intensity value into the frame buffer at a position corresponding to column *x* along scan line y is known as scan conversion line.

The procedure is

setPixel(x, y , intensity)

getPixel(x, y)

1.15 Line Drawing Algorithms

Straight line segments are uses in computer generated pictures. General requirement to draw a line:

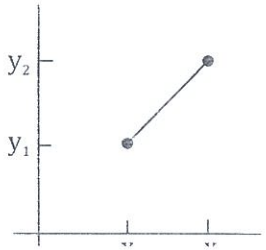
- Lines should appear straight.
- Lines should terminate accurately.
- Lines should have constant density.
- Line density should be independent of line length and angle.
- Lines should be drawn rapidly.

The Cartesian slope intercept equation for a straight line is

$$y = m \cdot x + b \quad (1)$$

Where m is the slope of the line
b as the y intercept.

The two endpoints are represented as (x_1, y_1) and (x_2, y_2)



Slope of the line m is

$$m = \frac{y_2 - y_1}{x_2 - x_1} \quad (2)$$

y intercept b

$$b = y_1 - m \cdot x_1 \quad (3)$$

Figure 1.27 Line path between two endpoints

For any given x interval Δx along a line, we compute the corresponding y interval Δy

$$\Delta y = m\,\Delta x \quad (4)$$

$$\text{Therefore} \Delta x = \frac{\Delta y}{m} \quad (5)$$

These equations form the basis for determining deflection voltages in analog devices.

- For lines with slope $|m| < 1$, Δx can be set proportional to a small horizontal deflection voltage and the corresponding vertical deflection is then set proportional to Δy as calculated from Eq.4.

- For lines with slope $|m| > 1$, Δy can be set proportional to a small vertical deflection voltage with the corresponding horizontal deflection voltage set proportional to Δx, calculated from Eq.5
- For lines with $m = 1$, $\Delta x = \Delta y$ and the horizontal and vertical deflections voltages are equal.
- In each case, a smooth line with slope m is generated between the specified endpoints.

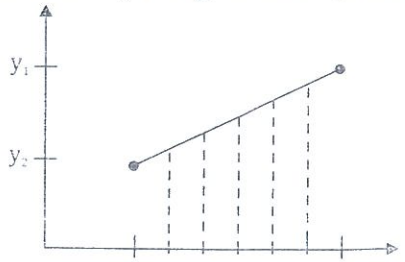


Figure 1.28 Straight line segments with five sampling positions

1.15.1 DDA Line Drawing Algorithm

The digital differential analyzer (DDA) is a scan-conversion line algorithm based on calculating either Δy or Δx, using equation 4 and 5.

Steps:

We sample the line at unit intervals in one coordinate and determine corresponding integer values nearest to the line path for the other coordinate.

Consider first a line with positive slope as shown in Fig. 1.27.

- If the slope m <= 1, we sample at unit *x* intervals ($\Delta x = 1$) and compute each successive y value as

$$y_{k+1} = y_k + m \quad (6)$$

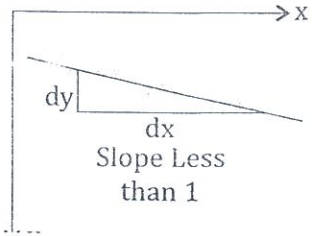


Figure 1.29 Line with Negative Slope

where **'k'** takes integer values starting from 1, for the first point, and increases by 1 until the final endpoint is reached. And **'m'** can be any real number between 0 and 1.

- If the positive slope m >= 1 then Δy = 1 and compute each successive x value as

$$x_{k+1} = x_k + \left(\frac{1}{m}\right) \quad (7)$$

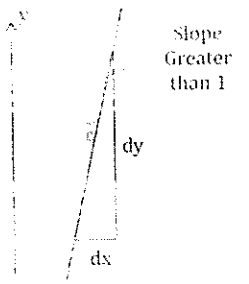


Figure 1.30 Line with Positive Slope

These Equations 6 and 7 are based on the assumption that lines are to be processed from the left endpoint to the right endpoint. If this processing is reversed, so that the starting endpoint is at the right, then either we have

$$\Delta x = -1 \quad \text{with } y_{k+1} = y_k - m \text{ and} \tag{8}$$

$$\Delta y = -1 \quad \text{with } x_{k+1} = x_k - \frac{1}{m} \tag{9}$$

DDA Line Drawing

This algorithm is summarized in the following

Algorithm

```
Step 1:   Input 2 end points (x1, y1) and (x2, y2).
Step 2:   Calculate dx= x2 - x1 and dy= y2 - y1.
Step 3:   Obtain the number of steps by using the condition
             If abs(dx) > abs(dy) then steps = abs(dx) else steps = abs(dy)
Step 4:   Determine the increment values of x and y as
             xinc = dx/steps and yinc = dy/steps
Step 5:      Plot the starting point (x, y)
Step 6:   For k = 1 to steps in increments of 1
Step 7:   Plot the next point by generating it as
             x = x + xinc
             y = y + y inc
Step 8:   End for loop
Step 9:   Stop.
```

(OR)

Procedure

```
void linedda(int x1,int y1,int x2,int y2)
{
      int dx=x2 - x1, dy=y2 - y1, steps, k;
      float xinc, yin, x= x1, y= y1;

      if(abs(dx)>abs(dy))
         steps=abs(dx);
      else steps=abs(dy);
      xinc=dx/(float)steps;
      yinc=dy/(float)steps;
      putpixel(round(x),round(y), 1)
      for(k=0; k<steps; k++)
```



```
      {
         x = x + xinc;
         y = y + yinc;
         putpixel(round(x),round(y), 1);
      }
}
```

Advantages of DDA Line Drawing

Faster method for calculating pixel position then the equation of a pixel position.

$y = mx + b$.

Disadvantages of DDA Line Drawing

The accumulation of round of error is successive addition of the floating point increments is used to find the pixel position but it take lot of time to compute the pixel position.

Example

DDA Line drawing for a line with endpoints (20, 10) and (28, 20)

$x_1 = 20, y_1 = 10$ and	$x_2 = 28, y_2 = 20$	abs(dx) > abs(dy)	
$dx = x_2 - x_1$,	$dy = y_2 - y_1$	abs (8) > abs(10) false	
$dx = 28 - 20$	$dy = 20 - 10$	then steps= abs(dy)	◊ steps = 10
$dx = 8$	$dy = 10$	xinc=dx/(float)steps;	◊ xinc = 8 / 10 = 0.8
		yinc=dy/(float)steps;	◊ yinc = 10 / 10 = 1

k	x	y
0	20	10
1	20.8	11
2	21.6	12
3	22.4	13
4	23.2	14
5	24	15
6	24.8	16
7	25.6	17
8	26.4	18
9	27.2	19
10	28	20

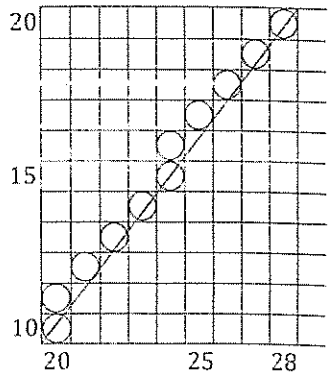


Figure 1.31 DDA Line Drawing Algorithms

1.15.2 Bresenham's Line Algorithm

An accurate and efficient raster line-generating algorithm, developed by Bresenham, scan converts lines using only incremental integer calculations. It entirely implements with integer arithmetic. Integer arithmetic is much faster than floating point. This algorithm does not require any multiplication or division.

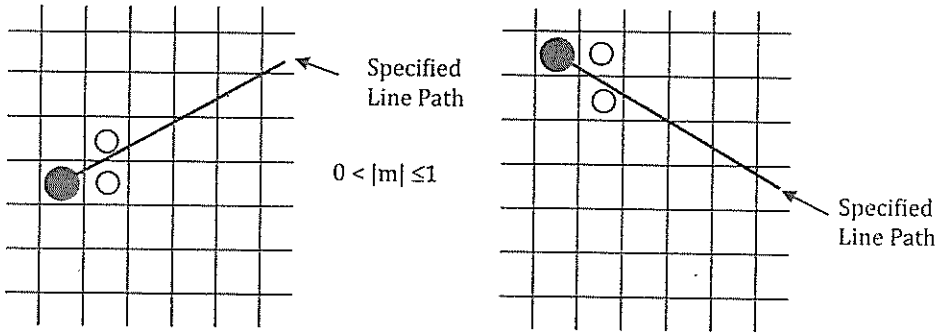


Figure 1.32 Straight Line Segment with (a) Positive Slope
(b) Negative Slope – where the next pixel to be chosen.

Difficulties when we plot the interval values, we have to decide which of the two possible pixel position is closer to the line path. This is solved by Bresenham algorithm.

To illustrate Bresenham approach considers the scan conversion process for lines with positive slope less than 1.

Pixel positions along a line path are determined by sampling at unit – x intervals. Starting from (x0, y0) of a line we step to each column and plot the pixel whose scan – line y closest to line path.

Assume, we have to determine that the pixel at (x_k, y_k) is to be displayed. We need to decide which pixel to plot in column x_{k+1}. Our choices are the pixel at position (x_{k+1}, y_k) and (x_{k+1}, y_{k+1}).

At sampling position $x_k + 1$, we label vertical pixel separations from the mathematical line path as d1 and d2.

According to Bresenhams Principal the difference between the vertical distance d1 = $y - y_k$ and d2 = $y_{k+1} - y$ is computed and the difference is used to select the upper or lower pixel.

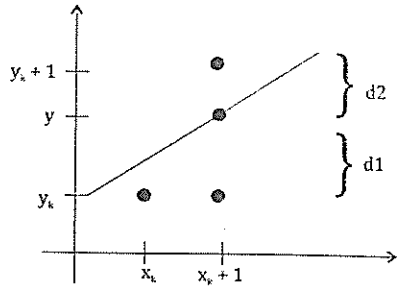


Figure 1.33 Distance between pixel positions and the line y coordinate at sampling position x_k+1

Defining the decision parameter as $p_0 = 2\Delta y - \Delta x$. The next value is evaluated depending on whether p_0 is positive or negative. To calculate the value of the decision parameter p_0, the first value p_0 is evaluated at the starting pixel position (x0, y0) as

$$\boxed{P_0 = 2\Delta y - \Delta x} \quad \text{.....(10)}$$

At each x_k along the line, starting at k =0, perform the following test.

If $p_k < 0$, the next point to plot is $(x_k + 1, y_k)$ and

$$\boxed{P_{k+1} = P_k + 2\Delta y} \quad \text{.....(11)}$$

Otherwise, the next point to plot is $(x_k + 1, y_k + 1)$ and

$$\boxed{P_{k+1} = P_k + 2\Delta y - 2\Delta x} \quad \text{.....(12)}$$

Bresenham's Line Drawing

Algorithm

Step 1: Input 2 end points and store the left endpoint in (x_0, y_0).

Step 2: Load (x_0, y_0) into the frame buffer and plot the first point.

Step 3: Calculate dx, dy, 2dy and 2dy - 2dx and obtain the starting value for the decision parameter as

$P_0 = 2dy - dx$

Step 4: At each x_k along the line, starting at k = 0, check the following test:

If $P_k < 0$, the next point to plot is $(x_k + 1, y_k)$ and

$P_k + 1 = P_k + 2dy$

Otherwise, the next point to plot is $(x_k + 1, y_k + 1)$ and

$P_k+1 = P_k + 2dy - 2dx$

Step 5: Repeat Step 4 dx times.

(OR)

Procedure

```
procedure linebres (int x1, int y1, int x2, int y2);
{
      int dx=abs(x2 - x1), dy=abs(y2 - y1), x, y, xend;
      int p= 2*dy - dx;
      int twoDy= 2*dy;
      int twoDyDx= 2*(dy - dx);
if (x1 > x2)
{
      x = x2;
      y = y2;
      xend= x1;
}
```

```
else
{
      x = x1;
      y = y1;
      xend= x2;
}
      setPixel(x, y);
      while (x < xend)
      {
         x++;
         if (p < 0)
            p = p + twoDy;
         else
            {
               y++;
               p = p + twoDyDx;
            }
               setPixel(x, y);
      }
}
```

Example

Bresenham's Line drawing for a line with endpoints (20,10) and (30,18)

$x_1 = 20, y_1 = 10$ and $x_2 = 30, y_2 = 18$

dx = abs(30 - 20)	dy = abs(18 - 10)
dx = 10	dy = 8
p_0 = 2*dy - dx;	
= 6	

We plot the initial point $(x_0, y_0) = (20, 10)$, and determine successive pixel position along the line path from the decision parameter as

k	P_k	(X_{k+1}, Y_{k+1})
0	6	(21,11)
1	2	(22,12)
2	-2	(23,12)
3	14	(24,13)
4	10	(25,14)
5	6	(26,15)
6	2	(27,16)
7	-2	(28,16)
8	14	(29,17)
9	10	(30,18)

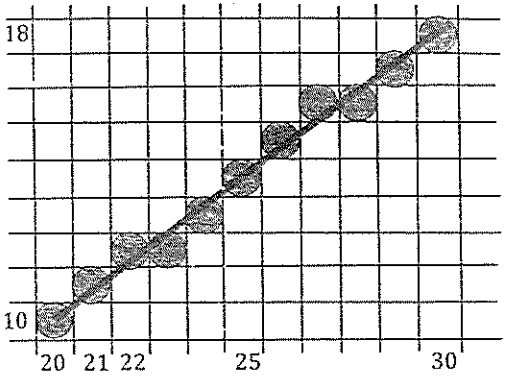


Figure 1.34 Bresenham's Line Algorithm

1.16 Circle Generating Algorithm

A circle is defined as the set of points that are all at a given distance 'r' from a center position (x_c, y_c). Fig 1.35(a).

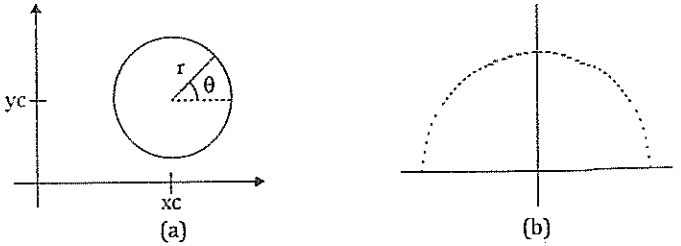


Figure 1.35 (a) Circle with center coordinates (x_c, y_c) as radius r.
(b) Circle plotted with (x_c, y_c)=(0,0).

The distance relationship is expressed by the Pythagorean theorem in Cartesian coordinates as

$$(x - x_c)^2 + (y - y_c)^2 = r^2 \quad (13)$$

This equation can be used to calculate the position of points on a circle circumference by stepping along the x- axis in unit steps from $x_c - r$ to $y_c + r$ and calculation corresponding 'y' value

$$y = y_c \pm \sqrt{r^2 - (x - x_c)^2} \quad (14)$$

Disadvantages of Circle Generating Algorithm

- It involves considerable computation at each step.
- The spacing between plotting pixel position is not uniform. Fig 1.35 (b).

Another Method to Eliminate Spacing

To calculate points along the circular boundary using polar co-ordinates, r and θ. Circle equation in parametric polar form is

$$\boxed{\begin{aligned} x &= x_c + r\cos\theta \\ y &= y_c + r\sin\theta \end{aligned}} \quad (15)$$

1.16.1 DDA Circle Drawing Algorithm

Algorithm

```
Step 1:   Input xcenter, ycenter and r.
Step 2:   Plot the pixel at(xcenter, ycenter)
Step 3:   for theta = 0 to 2 * 3.14159 in steps of 0.01
Step 4:   x= xcenter + r cos(theta)
Step 5:   y= ycenter + r sin(theta)
Step 6:   End for
Step 7:   Stop.
```

1.16.2 Bresenham's Circle Algorithm

If a circle is to be plotted efficiently, the use of trigonometric and power functions must be avoided. It also involves considerable amount of computation at each step and the time taken is more. This disadvantage is solved by Bresenham's circle algorithm.

A circle using Bresenham's algorithm works as follows. Computation can be reduced by considering the symmetry of circles. The shape of the circle is similar in each quadrant. We can generate the circle section in the second quadrant of the xy plane by noting that the two circle sections are symmetric with respect to the y-axis. And circle sections in the third and fourth quadrants can be obtained from sections in the first and second quadrants by considering symmetry about the *x*-axis.

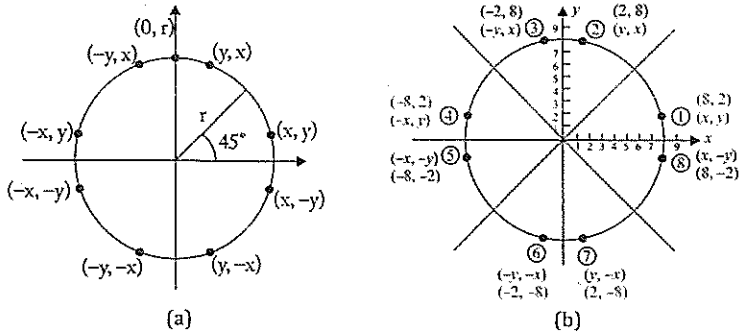


Figure 1.36 (a) Symmetry of a Circle. (b) Example of Symmetry of a Circle

The symmetry conditions are illustrated in Fig.1.36, where a point at position (x, y) on a one-eighth circle sector is mapped into the seven circle points in the other octants of the xy plane. We can generate

all pixel positions around a circle by calculating only the points within the sector from $x = 0$ to $x = y$.

Circle generating algorithms are based on incremental calculation of ***decision parameters,*** as in the Bresenham's line algorithm, which involves only simple integer operations. The decision parameter is used to find the closest pixel to the circumference at each sampling step.

A method for direct distance comparison is to test the halfway position between two pixels to determine if this midpoint is *inside or outside the circle.* This method is more easily applied to the circle with integer radius.

1.16.3 Midpoint Circle Algorithm

This is an incremental circle algorithm that is very similar to Bresenham's approach. We sample at unit intervals and determine the closest pixel position to the specified circle path at each step. For a given radius *r* and screen center position (x_c, y_c), we can first set up our algorithm to calculate pixel positions around a circle path centered at the coordinate origin (0, 0).

To apply the midpoint method, we define a circle function:

$$\boxed{f_{circle}(x,y) = x^2 + y^2 - r^2} \quad (16)$$

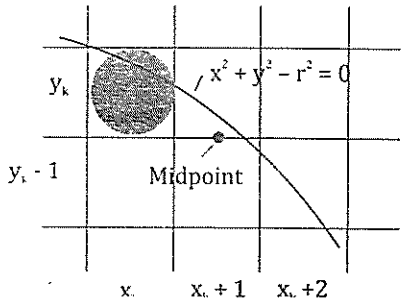


Figure 1.37 Midpoint between Pixels at Position x_k+1.

Any point (x, y) on the boundary of the circle with radius 'r' satisfies the equation

$$\boxed{f_{circle}(x,y) = 0} \quad (17)$$

To check the relative position of a point (x, y) is

$$f_{circle}(x,y)\begin{cases} <0 & \text{then } (x,y) \text{ is inside the circle} \\ =0 & \text{then } (x,y) \text{ is on the circle} \\ >0 & \text{then } (x,y) \text{ is outside the circle} \end{cases}$$

Assuming we have just plotted the pixel at (x_k, y_k), we next need to determine whether the pixel at position at $(x_k + 1, y_k)$ or the one at position $(x_k + 1, y_k - 1)$ is closer to the circle. Our decision parameter is the midpoint between these two pixels.

$$\boxed{P_k = f_{circle}\left(x_k + 1, y_k - \frac{1}{2}\right)} \quad ...(18)$$

$$p_k = (x_k + 1)^2 + \left(y_k - \frac{1}{2}\right)^2 - r^2 \quad ...(19)$$

For successive decision parameter are obtained by incremental calculations.

$$\boxed{p_{k+1} = f_{circle}\left(x_{k+1} + 1, y_{k+1} - \frac{1}{2}\right)} \quad ...(20)$$

$$= (x_k + 1 + 1)^2 + \left(y_{k-1} - \frac{1}{2}\right)^2 - r^2 \quad ...(21)$$

If $p_k < 0$, this midpoint is inside the circle and the pixel at y_k is closer to the circle boundary. Otherwise, the midpoint is outside or on the circle boundary, and we select the pixel $y_k - 1$.

Successive decision parameters are obtained using incremental calculations. Therefore the decision parameters can be written as

$$p_{k+1} = p_k + 2(x_k + 1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1 \quad (22)$$

where y_{k+1} is either y_k or y_{k-1} depending on the sign of p_k.

if $p_k < 0$ $\quad p_{k+1} = p_k + 2x_k + 1$

if $p_k \geq 0$ $\quad p_{k+1} = p_k + 2x_{k+1} - 2y_{k+1} + 1$

Computing p_0 at $(x_0, y_0) = (0, r)$

$$p_0 = f\left(1, r - \frac{1}{2}\right) \qquad \boxed{p_0 = \frac{5}{4} - r} \quad (23)$$

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

Where r, the radius is an integer, p_0 can be rounded off to set

$$\boxed{p_0 = 1 - r} \quad (24)$$

Midpoint Circle Drawing

Algorithm

Step 1: Input radius r and circle center (x_c, y_c). Obtain the first point on the circumference of a circle centered on the origin as $(x_0, y_0) = (0, r)$.

Step 2: Calculate the initial value of the decision parameter as $p_0 = \frac{5}{4} - r$. If r is an integer use $p_0 = 1 - r$.

Step 3: At eack x_k position, starting at k = 0, perform the following test

if $p_k < 0$ then $p_{k+1} = p_k + 2x_k + 1$

otherwise the next point along the circle is $(x_k + 1, y_k - 1)$ and

$p_{k+1} = p_k + 2x_{k+1} - 2y_{k+1} + 1$

Step 4: Determine symmetry points in the seven octants using the function plotpoints().

Step 5: Repeat steps 3 through 4 until x>=y.

Step 6: Stop.

(OR)

Procedure

```
void circleMidpoint (int xCenter, int yCenter, int radius)
      {
         int x = 0;
         int y = radius;
         int p = 1 - radius;

         void circlePlotPoints (int, int, int, int);
               /* Plot first set of points */
         circlePlotPoints(xCenter, yCenter, x, y);
         while (x < y)
         {
            x++ ;
            if (P < 0)
               p = p + 2 * x +1;
            else {
                  Y--;
                  p =p + 2 * (x - y) + 1;}
            circlePlotPoints(xCenter, yCenter, x, y);
      }
}
/*function circleplotpoints, used to plot the corresponding points (x,y) in all 4 quadrents*/
void circlePlotPoints (int xCenter, int yCenter, int x, int y)
{
      setpixel (xCenter + x, yCenter + y) ;
      setpixel (xCenter - x, yCenter + y);
      setpixel (xCenter + x, yCenter - y);
      setpixel (xCenter - x, yCenter - y ) ;
      setpixel (xCenter + y, yCenter + x);
      setpixel (xCenter - y , yCenter + x);
      setpixel (xCenter + y , yCenter - x);
      setpixel (xCenter - y , yCenter - x);
}
```

Example

Bresenham's Circle drawing for a circle with radius r = 10, we demonstrate the midpoint circle algorithm by plotting points in the first quadrant from x=0 to x=y.

$p_0 = 1 - r = 1 - 10$ $\quad \boxed{P_0 = -9}$

For the circle centered on the coordinate origin, initial point is $(x_0, y_0) = (0, 10)$

Calculating the Decision Parameter

k	P_k	(x_{k+1}, y_{k+1})	$2x_{k+1}$	$2y_{k+1}$
0	-9	(1, 10)	2	20
1	-6	(2, 10)	4	20
2	-1	(3, 10)	6	20
3	6	(4, 9)	8	18
4	-3	(5, 9)	10	18
5	8	(6, 8)	12	16
6	5	(7, 7)	14	14

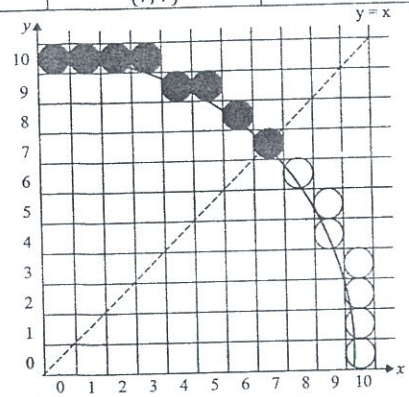


Figure 1.38 Circle using Midpoint Circle Algorithm with Radius as 10.

1.17 Ellipse Generating Algorithm

An ellipse is defined as the set of points such that the sum of the distances from two fixed positions (foli) is the same for all points.

The ellipse has 2 axes:

1. Major axes (rx)
2. Minor axes (ry)

1. **Major Axes :** It is the straight line segment extending from one side of the ellipse to the other through the foci.

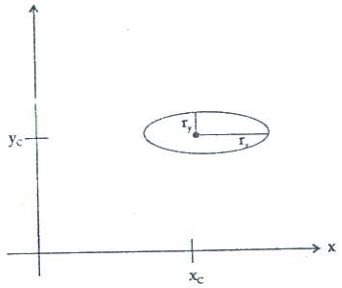


Figure 1.39 (a) Ellipse Centered at (x_c, y_c)

2. **Minor Axes :** It spans the shorter dimension of the ellipse, bisecting the major axis at the half way position (ellipse center between the two foci)

Equation of Ellipse

The equation of the ellipse can be written in terms of the ellipse center coordinates and parameters r_x and r_y as

$$\left(\frac{x-x_c}{r_x}\right)^2+\left(\frac{y-y_c}{r_y}\right)^2=1 \qquad \ldots(1)$$

with $(x_c, y_c) = (0, 0)$ sub in (1)

$$f_{ellipse}(x, y) = r_y^2x^2 + r_x^2y^2 - r_x^2r_y^2$$

where
$$f_{ellipse}^{(x,y)} = \begin{cases} <0 \text{ then the point } (x,y) \text{ is inside the ellipse} \\ =0 \text{ then the point } (x,y) \text{ is on the ellipse} \\ >0 \text{ then the point } (x,y) \text{ is outside the ellipse} \end{cases}$$

- Like the circle in the ellipse also we can calculate the symmetry.
- In ellipse symmetry four point are calculated rather than eight way.

Midpoint ellipse Alg

- We must calculate Pixel positions along the elliptical throughout one quadrant, then we obtain the remaining positions by symmetry.
- To calculate pixel positions in one quadrant, the quadrant is divided into 2 regions at point Q.

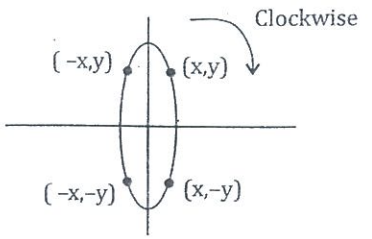


Figure 1.39 (b) Symmetry of a Ellipse

Region 1 : We can start at position (0, ry) and step clockwise along the elliptical path in the first quadrant, shifting from unit steps in x to unit steps in y, when slope becomes less than -1.

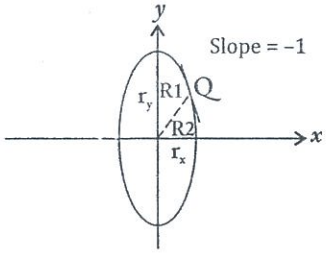


Figure 1.40 Region 1 and Region 2 in an Ellipse

Region 2:

* Alternatively we could start at $(r_x, 0)$ and select points in a counter clockwise order, shifting unit steps in y to unit steps in x, when the slope becomes greater than –1.

* Slope of the curve is –1
* over region 1 the slope is < 1
* over region 2 the slope is > 1
* As in the circle Alg, the next positions along the elliptical path is selected by evaluating the decision parameter at the midpoint. With the decision parameter P_k and P_{k+1} we get the decision parameter at Region 1 and Region 2
* Decision parameter at Region 1

$$P1_0 = r_y^2 - r_x^2 r_y + \frac{1}{4} r_x^2$$

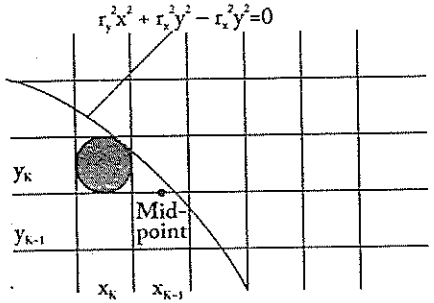


Figure 1.41 Midpoint between Pixel Sampling Position xk + 1

* Decision parameter at Region 2 $P2_0 = r_y^2\left(x_0 + \frac{1}{2}\right)^2 + r_x^2(y_0 - 1)^2 - r_x^2 r_y^2$

Midpoint Ellipse Drawing

Algorithm

Step 1: Input r_x, r_y and center of ellipse (x_c, y_c)
Calculate first point as $(x_0, y_0) = (0, r_y)$

Step 2: For region 1, compute initial decision parameter value as

$$P1_0 = r_y^2 - r_x^2 r_y + \frac{1}{4} r_x^2$$

Step 3: At each x_k position in region 1, with k = 0, test the value of $p1_k$.
If $p1_k < 0$ the next point on the ellipse is (x_{k+1}, y_k) and
$p1_{k+1} = p1_k + r_y^2 + 2r_y^2 x_{k+1}$
Similarly if $p1_k > 0$ the next point along the ellipse is $(x_k + 1, y_k - 1)$ and
$p1_{k+1} = p1_k + r_y^2 + 2r_y^2 x_{k+1} - 2r_x^2 y_{k+1}$
and increment with $2r_y^2 x_{k+1} = 2r_y^2 x_k + 2r_y^2$, $2r_x^2 y_{k+1} = 2r_x^2 y_k - 2r_x^2$

Step 4: Determine symmetry points in the other 3 quadrants and plot them.

Step 5: Repeat step 3 and 4 while $2r_y^2 x >= 2r_x^2 y$

Step 6: For Region 2, calculate the initial value of the decision parameter using the last point (x_0, y_0) of region 1 as $P2_0 = r_y^2\left(x_0 + \frac{1}{2}\right)^2 + r_x^2(y-1)^2 - r_x^2 r_y^2$

Step 7: At each y_k position in region 2, test the value of p2.
If $p2_k <= 0$ the next point along the ellipse is $(x_k + 1, y_k - 1)$ and
$p2_{k+1} = p2_k + r_x^2 + 2r_y^2 x_k + 1 - 2r_x^2 y_{k+1}$
Or $p2_{k+1} = p2_k + r_x^2 + 2r_y^2 x_k - 2r_x^2 y_{k+1}$
Otherwise if $p2_k > 0$, the next point along the ellipse is $(x_k, y_k - 1)$ and
$p2_{k+1} = p2k + r_x^2 - 2r_x^2 y_{k+1}$
and increment using same as region 1.

Step 8: Determine symmetry points in the other 3 quadrants and plot the points.
Step 9: Repeat steps 7 and 8 while y>0
Step 10: Stop.

Example

Given $r_x = 8$ and $r_y = 6$ trace the ellipse using midpoint ellipse drawing algorithm. Initial values of decision parameters with increments

$2r_y^2 x = 0$; increment $2r_y^2 = 72$

$2r_x^2 y$, increment $-2r_x^2 = -128$ until $2r_y^2 x > 2r_x^2 y$

Region 1 : The initial point for region 1, $(x_0, y_0) = (0, 6)$ given $r_x = 8$, $r_y = 6$

$p1 = r_y^2 - r_x^2 r_y + (1/4) r_x^2 = -332$

k	$P1_k$	(x_{k+1}, y_{k+1})	$2r_y^2 x_{k+1}$	$2r_x^2 y_{k+1}$
0	-332	(1, 6)	72	768
1	-224	(2, 6)	144	768
2	-44	(3, 6)	216	768
3	208	(4, 5)	288	640
4	-108	(5, 5)	360	640
5	288	(6, 4)	432	512
6	244	(7, 3)	504	384

If $p1 < 0$ then $p1 = p1 + r_y^2 + 2r_y^2 x_{k+1}$
Else
$p1 = p1 + r_y^2 + 2r_y^2 x_{k+1} - 2r_x^2 y_{k+1}$
when $p1 < 0$ only x is incremented else both x is incremented and y decremented.

Region 2 : The initial point for region 2, $(x_0, y_0) = (7, 3)$ $r_x = 8$, $r_y = 6$
The initial decision parameter p2
$P2 = f(7+1/2, 2) = -151$

k	$P2_k$	(x_{k+1}, y_{k+1})	$2r_y^2 x_{k+1}$	$2r_x^2 y_{k+1}$
0	-151	(8, 2)	576	256
1	233	(8, 1)	576	128
2	745	(8, 0)	-	-

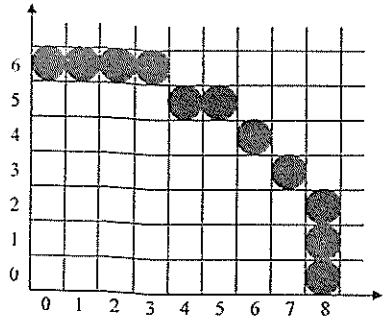


Figure 1.42 Positions along the elliptical path centered on the origin with rx = 8 and ry = 6.

1.18 Attributes of Output primitives

Any parameter that affects the way a primitive is to be displayed is referred to as an attribute parameter. Such as color, size is fundamental characteristics.

1.18.1 Line Attributes

The attributes of a straight line segment are

- Type.
- Width.
- Color.

Line Type

The different types of line types are

- Solid lines.
- Dashed lines.
- Dotted lines.
- Dashed dotted line.

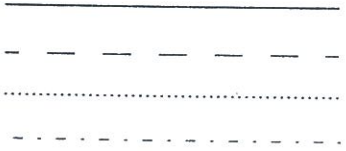


Figure 1.43 Line Types

Line Type attributes in a PHIGS application program, a user invokes the function

setLinetype(lt)

Where parameter 'lt' is assigned a positive integer value of 1, 2, 3, 4 to generate solid, dashed dotted or dash-dotted.

Line Width

Implementation of line width options depends on the capabilities of the output devices. We set the line width attribute with the command.

setLinewidthScaleFactor(lw)

Line width parameter 'lw' is assigned a positive number to indicate the relative width of the line to be displayed.

- A value of 1 specifies a standard width line.
- A value greater that 1 produces line thicker than the standard line.
- A value 0.5 produces line whose width is half that of the standard line.

We adjust the slope of the line ends to give them a better appearance by adding ***line caps.***

Problem when joining lines so line caps are used to adjust shape.

Different types of Line Caps

- Butt caps.
- Round caps.

- Projecting square caps.
- **Butt Caps:** This is obtained by adjusting the end positions of the component parallel lines so that the thick line is displayed with square end that are perpendicular to the line path.
- **Round Caps:** This is obtained by adding a filled semicircle to each butt cap.
- **Projecting square Caps:** This is obtained by extending the line and adds butt caps that are positioned one half of the line width beyond the specified endpoints.

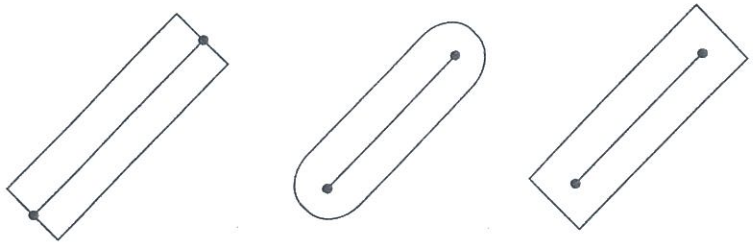


Figure 1.44 Thick lines drawn with (a) Butt Caps (b) Round Caps (c) Projection Square Caps

Line Color

When a system provides color options, a parameter giving the current color index is included in the list of system-attribute values.

The line color value in PHIGS with the function

setPolylineColourIndex(lc)

Non-negative integer values corresponds to allowed color choices, are assigned to the line color parameter 'lc'.

Example

```
setLinetype (2) ;
setLinewidthScaleFactor (2) ;
setPolylineColourIndex (5 ) ;
```

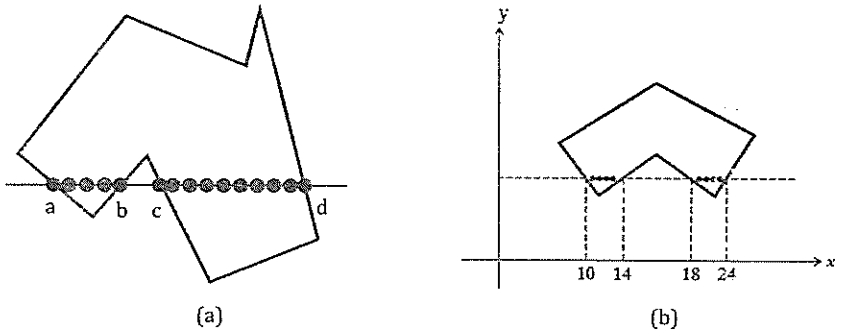
1.18.2 Area Filling

Scan – Line Algorithm for area filling:

Scan line filling algorithm is used for filling polygon areas. This procedure contains the following steps:

- For each scan-line locate the intersection of the scan-line with the edges.
- Sort the intersection points from left to right.
- Generate frame-buffer positions along the current scan-line between pairwise intersection

points.



(a) (b)

Figure 1.45 (a) Interior pixels along a scan line passing through a polygonarea
(b) Example : The polygon is filled for this particular scan line from r = 10 to14 and x = 18 to 24

- Draw the interior intersection points pair wise. (a-b), (c-d)
- Problem with corners. Same point counted twice or not?
- From fig. 1.45 a, b, c and d are intersected by 2 line segments each.
- Some scan line intersecting polygon vertex needs special handling. When a scan-line passes through a vertex, it intersects two polygon edges at that point and hence adds two points to the list of intersections for the scan line. In fig 1.46, scan line y' intersects the polygon at an even number of edges where as scan-line y intersects five polygon edges i.e, odd number of edges. Since scan-line y' generates even number of intersections, they can be paired to identify correctly the interior pixel spans.

olution:

o determine the correct interior point

- Make a clockwise or counter-clockwise traversal on edges.
- Check if y is monotonically increasing or decreasing.
- If direction changes, double intersection, otherwise single intersection.

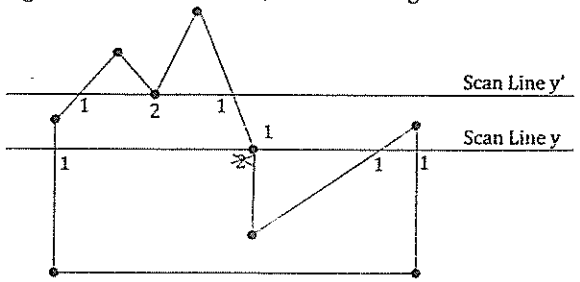


Figure 1.46 : Intersection points along scan lines that intersect polygon vertices

- Calculation can be performed by making use of *Coherence* properties of an image to be displayed. Properties of one part of a scene are related with the other in a way that can it be used to reduce processing of the other.
- To determine edges intersections, incremental co-ordinate calculation can be performed using the fact that the slope of the edge is constant from scan line to the next.

Area Filling Attributes

Fill an area with a solid color or a patterned fill and choices for the particular colors and patterns. These fill options can be applied to polygon regions or to areas defined with curved boundaries. In addition, areas can be painted using various brush styles, colors, and transparency parameters.

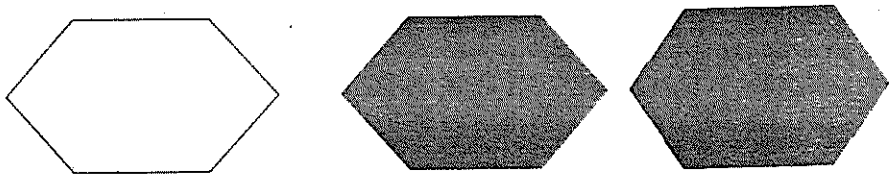
Fill Styles:

Areas are displayed with three basic fill styles:

- Hollow with a color border.
- Filled with a solid color.
- Filled with a specified pattern or design.
- A basic fill style is selected in a **PHIGS** program with the function

setInteriorStyle(fs)

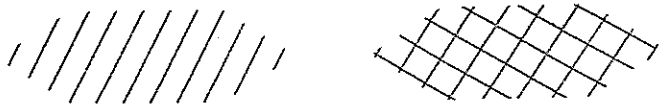
Values for the fill-style parameter fs include hollow, solid, and pattern.



Hollow (a) Solid (b) Patterned (c)

Figure 1.47 Polygon fill styles.

Another value for fill style is hatch, which is used to fill an area with selected hatching patterns-parallel lines or crossed lines.



Diagonal Hatch Fill — Diagonal Cross-Hatch Fill

Figure 1.48 Polygon fill using Hatch Patterns.

Hollow areas are displayed using only the boundary outline, with the interior color the same as the background color. A solid fill is displayed in a single color up to and including the borders of the region. The color for a solid interior or for a hollow area outline is chosen with

```
setInteriorColorIndex(fc)
```

where fill color parameter fc is set to the desired color code. Solid fill of a region can be accomplished with the scan-line procedures.

Other fill options include specifications for the edge type, edge width, and edge color of a region. These attributes are set independently of the fill style or fill color, and they provide for the same options as the line-attribute parameters (line type, line width, and line color). That is, we can display area edges dotted or dashed fat or thin, and in any available color regardless of how we have filled the interior.

1.18.3 Character Attributes

The appearance of displayed characters is controlled by attributes such

- Font (typeface)
- Style
- Color
- Size (width/height)
- Orientation
- Path
- Spacing
- Alignment

Attributes can be set both for entire character strings (text) and for individual characters defined as marker symbols.

Text Attributes

There are a many text options that is available to graphics programmers.

- **Font and Style**
  - The text can be set with a **font** (or typeface) such as New York, Courier, Helvetica, London, Times New Roman.
  - The characters can be set with a **font style** such as underline, boldface, *italics* and in outline or shadow styles.

    A font and associated style is selected in a program by setting an integer code for the text font parameter 'tf' in the function

```
setTextFont(tf)
```

- **Color** settings for displayed text

```
setTextColorIndex(tc)
```

  where text color parameter 'tc' specifies an allowable color code.

- We can adjust **text size** by scaling the overall dimensions (height and width) of characters or by scaling only the character width.

```
setCharacterHeight(ch)
```

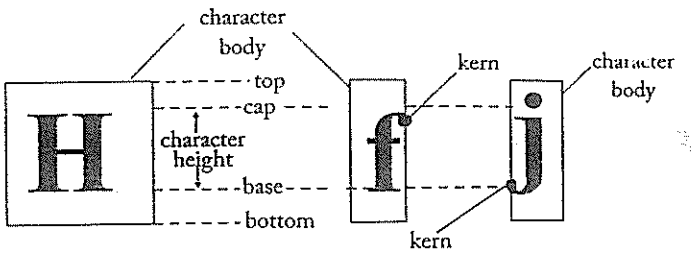



Figure 1.49 Character Body

Parameter 'ch' is assigned a real value greater than 0 to set the coordinate height of capital letters.

The distance between the *bottom line* and the *top line* of the character body is the same for all characters in a particular size and typeface, but the body width may vary.

- **Character height** is defined as the distance between the *baseline* and the *cap line* of characters. Kerned characters are such as f and j typically extend beyond the character-body limits.

Height 1

Height 2

Height 3

Figure 1.50 The effect of different character height.

- The **width** of text can be set with the function

```
setCharacterExpansionFactor(cw)
```

where the character-width parameter 'cw' is set to positive real value that scales the body width of characters.

width 0.5

width 1.0

width 2.0

Figure 1.51 The effect of different character width.

- **Spacing** between characters is controlled separately with

```
setCharacterSpacing(cs)
```

where the character-spacing parameter 'cs' can he assigned any real value.

The value assigned to cs determines the spacing between characters bodies.

- Negative values for cs overlap character bodies.
- Positive values insert space to spread out the displayed characters.
- The value 0 to *cs* causes text to be displayed with no space between character bodies.

Spacing 0.0

Spacing 0.5

Spacing 1.0

Figure 1.52 The effect of different character spacing

- The **orientation** for a displayed character string is set according to the direction of the character up vector:

setCharacterUpVector(upvect)

- Parameter 'upvect' in this function is assigned two values that specify the x and y vector components. For example, with upvect = (1, 1), the direction of the up vector is 45 degree and text would be displayed as shown in Fig. 1.53.
- To arrange character strings vertically or horizontally

setTextPath(tp)

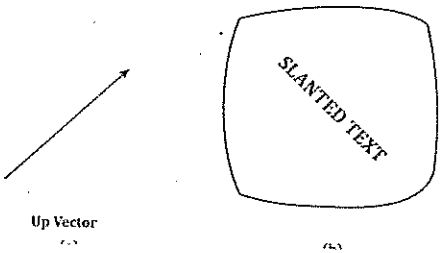


Figure 1.53

(a) Direction of up Vector

(b) Control Orientation of Display Text

where the text-path parameter 'tp' can be assigned the value: right, *left, up,* or down. Examples of text displayed with these four options are shown in fig 1.54(b).

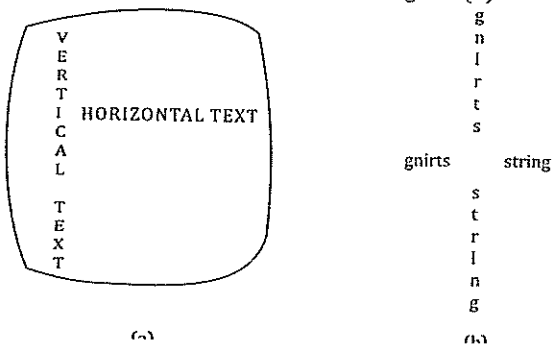


Figure 1.54 (a) Text path, vertical and horizontal.

(b) Text display with four text path options,

- **Alignment** attributes are set with

setTextAlignment(h, v)

where parameters 'h' and 'v' control horizontal and vertical alignment. Horizontal alignment is set by assigning h a value of *left,* centre, or *right..* Vertical alignment is set by assigning v a value of top, cap, *half, base,* or *bottom.*

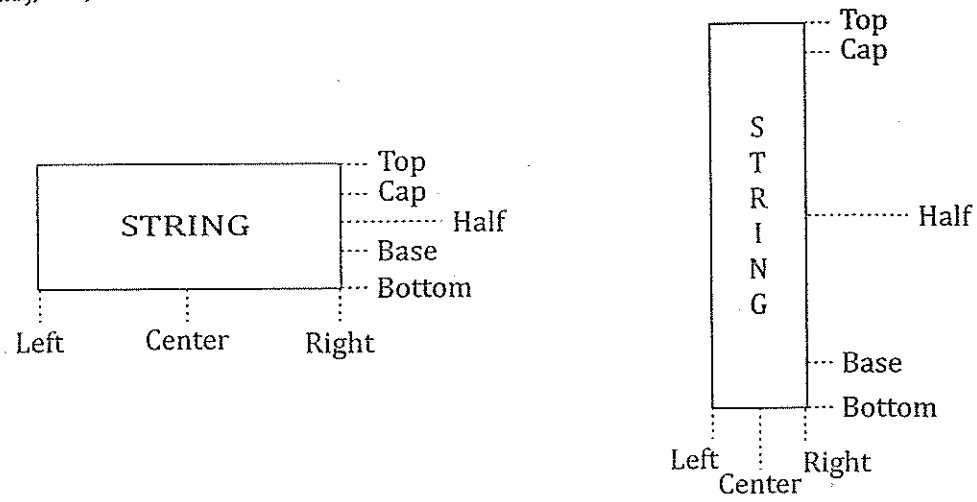


Figure 1.55 Alignment attribute values for horizontal and vertical strings.

- **Marker attribute** is a marker symbol for a single character that can be displayed in different colors and in different sizes.

We select a particular character to be the marker symbol with

setMarkerType(mt)

where marker type parameter 'mt' is set to an integer code. Typical codes for marker type are the integers 1 through 5, specifying, respectively, a dot (.), a vertical cross(+), an asterisk(*), a circle (o), and a diagonal cross(x). Displayed marker types are centered on the marker coordinates.

We set the marker size with

setMarkersizescaleFactor(ms)

Where parameter marker size 'ms' assigned a positive number. Values greater than 1 produce character enlargement; values less than 1 reduce the marker size.

Marker color is specified with

setPolymarkerColorIndex(mc)

where parameter 'mc' assigns a color code.

1.19 Review Questions

Short Answer Questions

1. Define Computer graphics.
2. What is meant by refreshing of the screen?
3. Define Random scan/Raster scan displays?
4. List out the disadvantages of Penetration techniques?
5. What do you mean by emissive and non-emissive displays?
6. What is persistence?
7. What is resolution?
8. What is Aspect ratio?
9. What is meant by Addressability?
10. Define pixel?
11. What is frame buffer?
12. What is bitmap and what is pixmap?
13. Where the video controller is used?
14. List out the different application of Computer Graphics?
15. Explain the Graphics software.
16. What is an output primitive?
17. What do you mean by 'jaggies'?
18. What is point in the computer graphics system?
19. Write short notes on lines?
20. Define Circle?
21. Define Ellipse?
22. Define polygon?
23. What is scan line algorithm?
24. Define coherence properties?
25. What is type face?
26. What are the various attributes of a line?
27. What is a Line cap?
28. List the different line caps?
29. What is kerned character?
30. What is character up vector?
31. Explain the steps involved in DDA circle algorithm
32. Explain any three text attributes.

Long Answer Questions

1. Explain the working of CRT, with a neat diagram.
2. Discuss the working of shadow mask CRT, with neat diagram.
3. Explain the working of emissive display?
4. Draw the architecture of a raster graphics system with the display processor. Explain raster scan display processor.
5. Explain raster scan display with neat diagram.
6. How are computer graphics used in training and education?
7. Explain some applications of computer graphics.
8. Explain the working of beam penetration method.
9. Explain the color model
10. What are the different factors affecting CRT.
11. Explain the different classifications of computer graphics.
12. Explain in detail about the display processor with raster system.
13. Explain in detail about the DDA line drawing algorithm?
14. Explain Bresenham's line drawing algorithm?
15. Explain Midpoint Circle algorithm?
16. Explain Ellipse generating Algorithm?
17. Explain Area fill attributes.
18. Give the different attributes for line in detail?
19. Explain the character attributes?
20. Define output primitives. Explain the line width and line color procedure.
21. Explain Scan-line algorithm for area filling.
22. Write the DDA circle algorithm. Illustrate it for a circle with radius 5 and center (0,0)

❑❑❑❑

NOTES

UNIT 2

2D-TRANSFORMATION, WINDOWING AND CLIPPING

CONTENTS

2.1 Geometric Transformation

Transformations are one of the primary concepts used in computer graphics for altering or manipulating the object. Animation is produced by moving the "camera" or the objects in a scene along animation paths. Many applications use the **Geometric Transformations** to change the position, orientation, and size of object.

The basic geometric transformation are:

- Translation
- Rotation
- Scaling

Other transformation are:

- Reflection
- Shear

In order to transform an object, we transform the points that define it.

For example:

- Polygon: Its vertices.
- Circle: Its center and, perhaps, its radius

2.2 Translation

A translation is applied to an object by repositioning it along a straight-line path from one coordinate location to another. We translate (or move) points to a new position by adding a translation distance, t_x and t_y to the original coordinate position (x, y) to a new position (x', y') Fig 2.1.

$$x' = x + t_x \qquad y' = y + t_y \quad \text{.....(1)}$$

The translation distance pair (t_x, t_y) is called a **translation vector** or **shift vector.**

The translation equation (1) can be expressed as a single matrix equation by using column vectors.

$$P' = \begin{bmatrix} x' \\ y' \end{bmatrix}, P = \begin{bmatrix} x \\ y \end{bmatrix}, T = \begin{bmatrix} t_x \\ t_y \end{bmatrix} \quad \text{.....(2)}$$

$$P' = P + T$$

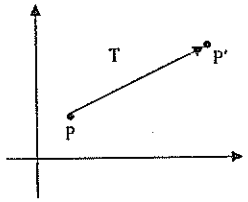


Figure 2.1 Translating a point from position P to position P' with translation vector T.

Example

Consider a triangle with 3 Coordinate Points (10, 5), (20, 5), (15, 10) with a translation distance (t_x, t_y) as (5, 3).

Now find the new coordinate positions with the equation:

$$x' = x + t_x$$
$$y' = y + t_y$$

The equation is applied to each of the line endpoints and redrawing the line between the new endpoint positions.

New Points

For (10,5) $x' = 10+5 = 15$, $y' = 5+3 = 8 \rightarrow$ (15,8)

For (20,5) $x' = 20+5 = 25$, $y' = 5+3 = 8 \rightarrow$ (25,8)

For (15,10) $x' = 15+5 = 20$, $y' = 10+3 = 13 \rightarrow$ (20,13)

The new coordinate points are: (20, 10), (30, 10), (25, 15)

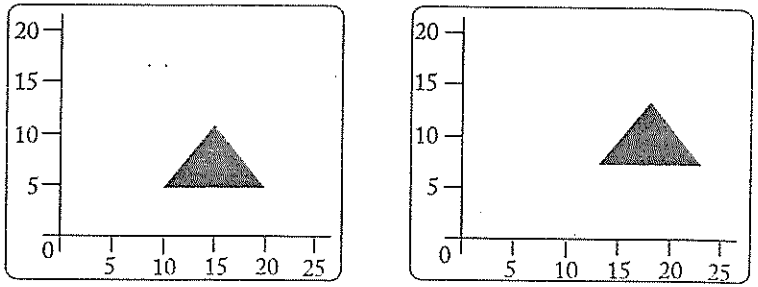


Figure 2.2 Moving a Polygon from Position (a) to Position (b)

Translation is a ***rigid-body transformation*** that moves objects without deformation. That is, every point on the object is translated by the same amount.

Polygons are translated by adding the translation vector to the coordinate position of each vertex and regenerating the polygon using the new set of vertex coordinates.

To change the position of a circle or ellipse, we translate the center coordinates and redraw the figure in the new location.

Exercise

1. **Example:**

   Consider a polygon with 4 Coordinate Points (0, 0), (4, 0), (2, 3), (2, 1) with a translation distance (t_x, t_y) as (0, 1). Find the new coordinate positions for the polygon.

2. **Example:**

   Consider a triangle with 3 Coordinate Points (20, 0), (60, 0), (40, 100) with a translation distance (t_x, t_y) as (100, 10). Find the new coordinate positions for the polygon.

2.3 Rotation

A two-dimensional rotation is applied to an object by repositioning it along a circular path in the xy plane. To rotate an object we need to specify a rotation angle θ and the position (x_r, y_r) of the **rotation point (or pivot point)** about which the object is to be rotated.

To rotate a line or polygon, we must rotate each of its vertices with the same angle θ Positive values for the rotation angle define counterclockwise rotations about the pivot point, and negative values rotate objects in the clockwise direction.

The transformation equations for rotation of a point position P when the pivot point is at the coordinate origin. In the fig. 2.3, r is the constant distance of the point P from the origin, angle Φ is the original angular position of the point P, and θ is the rotation angle.

Using standard *trigonometric identities*, we can express the transformed coordinates in terms of angles θ and Φ as

$$x' = r\cos(\Phi + \theta)$$
$$= r\cos\Phi\cos\theta - r\sin\Phi\sin\theta$$
$$y' = r\sin(\Phi + \theta)$$
$$= r\cos\Phi\sin\theta + r\sin\Phi\cos\theta \quad \text{.....(3)}$$

The original coordinates of the point in polar coordinates are:

$$x = r\cos\Phi,$$
$$y = r\sin\Phi \quad \text{.....(4)}$$

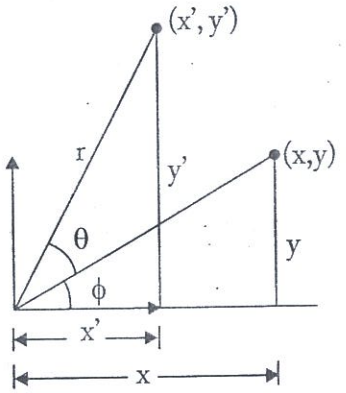


Figure 2.3 Rotation a point from position (x,y) to (x', y') with angle θ.

Substitute equation (4) into (3), The transformation equation for rotating a point at position (x, y) with an angle θ about the origin:

$$\boxed{x' = x\cos\theta - y\sin\theta}$$
$$\boxed{y' = x\sin\theta + y\cos\theta} \quad \text{.....(5)}$$

With the column vector representation, the rotation equation in the matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\boxed{P' = R.P} \quad \text{.....(6)}$$

Rotations are rigid-body transformations that move objects without deformation.

A straight line is rotated by applying the rotation equations to each of the line endpoints and redrawing the line between the new endpoint.

Polygons are rotated with the rotation angle for each vertex and regenerating the polygon using the new vertices.

Curved lines arc rotated by repositioning the defining points and redrawing the curves.

An ellipse can be rotated about its center coordinates by rotating the major and minor axes.

Example

Consider a polygon with 4 Coordinate Points (0, 0), (4, 0), (2, 3), (2,1) with rotation angle θ = 15 degree.

Now find the new coordinate positions with the equation:

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

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

New Coordinate Points

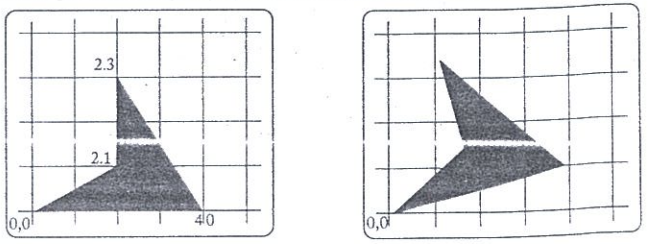
For (0,0) $x' = 0 * \cos 15 - 0 * \sin 15$
$y' = 0 * \sin 15 + 0 * \cos 15$ → (0,0)

For (4,0) $x' = 4 * \cos 15 - 0 * \sin 15$
$y' = 4 * \sin 15 + 0 * \cos 15$ → (3.86,1.04)

For (2,3) $x' = 2 * \cos 15 - 3 * \sin 15$
$y' = 2 * \sin 15 + 3 * \cos 15$ →(1.16, 3.42)

For (2,1) $x' = 2 * \cos 15 - 1 * \sin 15$
$y' = 2 * \sin 15 + 1 * \cos 15$ →(1.67, 1.48)

The new coordinate points are: (0, 0), (3.86, 1.04), (1.16, 3.42), (1.67, 1.48)



(a) (b)

Figure 2.4 Rotating a Polygon with θ= 15 degree

Exercise

1. **Example:**

   Consider a triangle with 3 Coordinate Points (20, 0), (60, 0), (40, 100) with rotation angle θ = 45 degree. Find the new coordinate positions for the triangle.

2. **Example:**

Consider a polygon with Coordinate Points (5, 2), (9, 2), (9, 6), (7, 8), (5, 6) with rotation angle θ = 45 degree. Find the new coordinate positions for the polygon.

2.4 Scaling

A scaling transformation is to alter the size of an object. This operation can be carried out for polygons by multiplying the coordinate values (x, y) of each vertex by scaling factors s_x and s_y to produce the transformed coordinates (x', y').

$$\boxed{\begin{aligned} x' &= x \cdot s_x \\ y' &= y \cdot s_y \end{aligned}} \qquad \text{.....(7)}$$

Scaling factor s_x scales objects in the x direction, while s_y scales in the y direction.

The scaling equations in the matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\boxed{P' = S.P} \qquad \text{.....(8)}$$

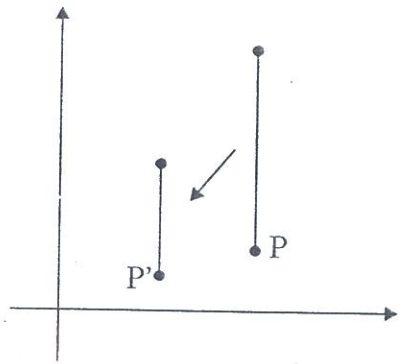


Figure 2.5 A line scaled using with scaling factor s_x =2 and s_y =1 and moves closer to the coordinate origin.

Uniform Scaling

Any positive numeric value can be assigned to the scaling factors s_x and s_y.

- Value less than 1 reduces the size of the object.
- Value greater than 1 produces an enlargement of the object.
- When the value of s_x =1 and s_y =1 does not change the size of the object.
- When s_x and s_y are assigned the same value then it called **uniform scaling.**
- Unequal values for s_x and s_y result in a **differential scaling.**

Objects transformed with equation (8) are both scaled and repositioned. It can control the location of a scaled object by choosing a position, called the **fixed point** that remains unchanged after the scaling transformation.

The Scaling factors are:

- With value less than 1 object moves closer to the coordinate origin.
- With value greater than 1 object moves away from the coordinate origin. Fig 2.5.

Example

Consider a Square with 4 Coordinate points (1, 0), (3, 0), (3, 2), (1,2) with s_x =2 and s_y = 1.

Now find the new coordinate positions with the equation:

$x' = x \cdot s_x$

$y' = y \cdot s_y$

		New Coordinate Points
For (1, 0)	x' = 1 * 2 =2 , y' = 0 * 1=0	→ (2,0)
For (3, 0)	x' =3 * 2 =6 , y' = 0 * 1=0	→ (6,0)
For (3, 2)	x' = 3 * 2 =6 , y' = 2 * 1=2	→ (6,2)
For (1, 2)	x' = 1 * 2 =2 , y' = 2 * 1=2	→ (2,2)

The new coordinate points are: **(2, 0), (6, 0), (6, 2), (2, 2)**

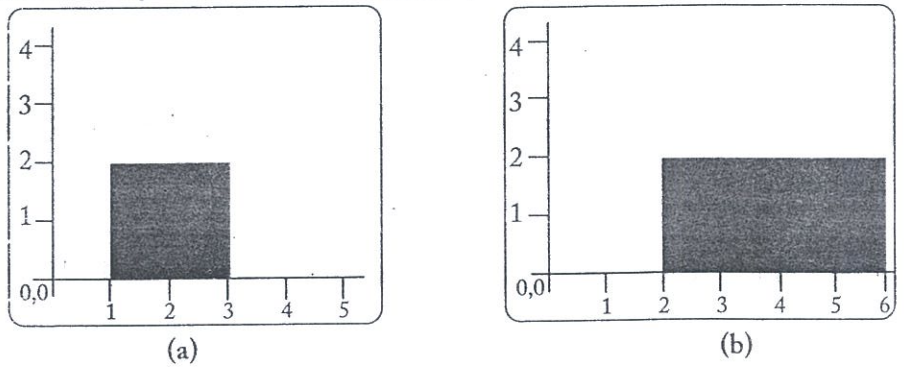


Figure 2.6 Turning a square (a) into a rectangle (b).

Exercise

1. **Example:**

Consider a triangle with 3 Coordinate Points (20, 0), (60, 0), (40, 100) with a scaling factor (s_x, s_y) as (2, 2). Find the new coordinate positions for the polygon.

2. **Example:**

Consider a polygon with 4 Coordinate Points (0, 0), (4, 0), (2, 3), (2, 1) with a scaling factor (s_x, s_y) as (0.5, 0.7)

2.5 Other Transformation

Basic transformations such as translation, rotation, and scaling are included in most graphics packages. Some packages provide a few additional transformations that are useful in certain applications.

- Reflection
- Shear

2.5.1 Reflection

A reflection is a transformation that produces a mirror image of an object. The reflection is generated relative to an axis of reflection by rotating the object 180 degree about the reflection axis. We can choose an axis of reflection in the xy plane or perpendicular to the xy plane.

The transformation matrix:

Reflection about the x – axis

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Reflection about the y – axis

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

.....(9) & (10)

- A reflection about x axis keeps x values the same but "flips" or changes the y values of coordinate positions.
- A reflection about y axis keeps y values the same but "flips" or changes the x values of coordinate positions.

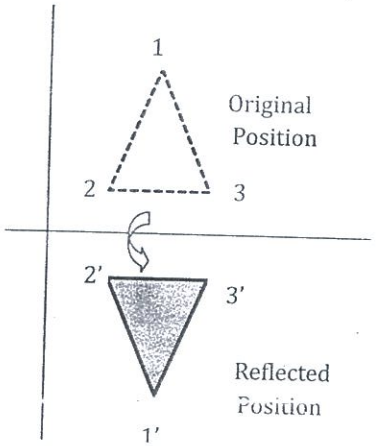


Figure 2.7 Reflection about x axis

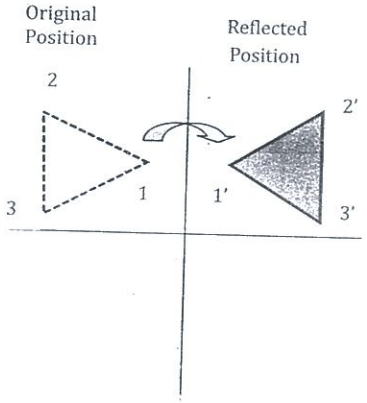


Figure 2.8 Reflection about y axis

Reflection relative to the coordinate origin:

When we flip both x and y coordinates of a point by reflecting relative to an axis that is perpendicular to the xy plane and that passes through the coordinate origin.

The matrix representation:

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

.....(11)

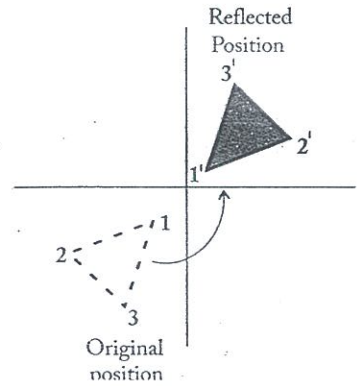


Figure 2.8(a) Reflection about coordinate origin

Reflection relative to a line y = x:

The matrix representation:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

.....(12)

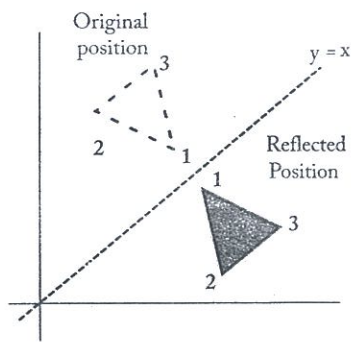


Figure 2.9 Reflection of an object with respect to the line y=x

We can derive this matrix by a sequence of rotation and coordinate- axis reflectionmatrices.

Method

Step 1 : Perform a clockwise rotation through a 45 degree angle, which rotates the line y = x onto the x axis.

Step 2 : Perform a reflection with respect to the x axis.

Step 3 : Rotate the line y = x back to its original position with a counterclockwise rotation through 45 degree.

2.5.2 Shear

A transformation that distorts the shape of an object such that the transformed shape appears as if the object were composed of internal layers that had been caused to slide over each other is called a shear.

Two types of shear transformation:

- An x-direction shear.
- An y-direction shear.

The shearing transformations are those that shift coordinate x values and those that shift y values.

X-direction Shear

An x-direction shear relative to the x axis is produced with the transformation matrix. This transformation changes only the x coordinates. Each point in the image is displayed horizontally by a distance that is proportional to the y coordinates.

$$\begin{bmatrix} 1 & sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{.....(13)}$$

The transformed coordinate positions are:

$$\boxed{\begin{aligned} x' &= x + sh_x \cdot y \\ y' &= y \end{aligned}} \quad \text{.....(14)}$$

Any real number can be assigned to the shear parameter sh_x.

Y-direction Shear

An y-direction shear relative to the y axis is produced with the transformation matrix. This transformation changes only the y coordinates. Each point in the image is displayed vertically by a distance that is proportional to the x coordinates.

$$\begin{bmatrix} 1 & 0 & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{.....(15)}$$

The transformed coordinate positions are:

$$\boxed{\begin{aligned} x' &= x \\ y' &= y + sh_y \cdot x \end{aligned}} \quad \text{.....(16)}$$

Example X-DIRECTION SHEAR

Turning the square into a parallelogram by setting $sh_x = 2$. Negative values for sh_x shift coordinate positions to the left.

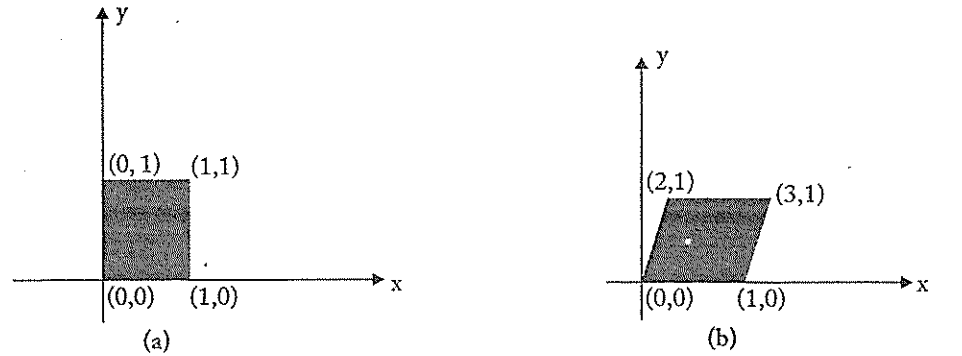


Figure 2.10 Changes the square into a parallelogram using x-direction shear with $sh_x = 2$

Example Y-DIRECTION SHEAR

Turning the square into a parallelogram by setting $sh_y = 2$.

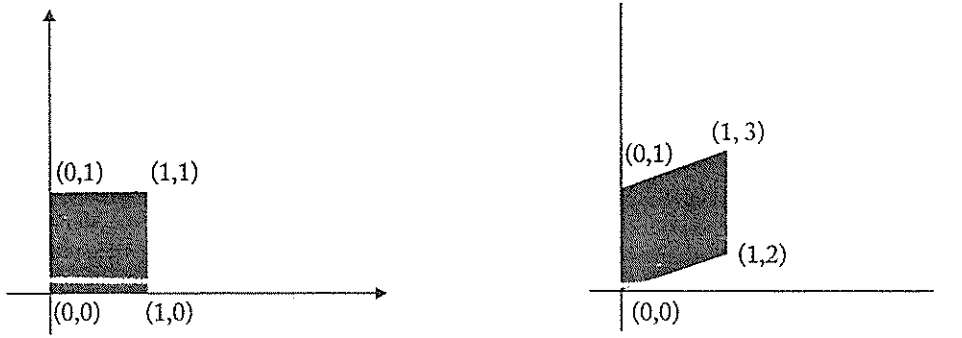


Figure 2.11 Changes the square into a parallelogram using y-direction shear with $sh_y = 2$

2.6 Matrix representation and Homogeneous Coordinates

An animation involves sequences of geometric transformations. We perform translations, rotations, and scaling to the picture components into their proper positions.

To produce a sequence of transformations with these equations, such as scaling followed by rotation then translation, we must calculate the transformed coordinate's one step at a time.

- First, coordinate positions are scaled
- Then the scaled coordinates are rotated
- Finally the rotated coordinates are translated.

A more efficient approach would be to combine the transformations so that the final coordinate positions are obtained directly from the initial coordinates, thereby eliminating the calculation of intermediate coordinate values.

To do this, we need to reformulate the equation by representing in the form of homogeneous coordinate. A Cartesian point (x, y) is converted to a homogeneous representation (x_h, y_h, h), equations containing x and y, such as f(x, y) = 0, become homogeneous equation in the three parameters x_h, y_h, h where we set h=1.

When we represent in homogeneous representation, the matrix addition associated with the translation can be eliminate. By using the homogeneous representation all geometric transformation equations can be represented as matrix multiplications in 3 X 3 matrix.

Translation

Translation is represented as a multiplication. The 3 x 3 matrix is:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \text{.....(17)}$$

$$\boxed{P' = T(t_x, t_y).P} \quad \text{.....(18)}$$

Rotation

Rotation transformation equations about the coordinate origin are written as

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \text{.....(19)}$$

$$\boxed{P' = R(\theta).P} \quad \text{.....(20)}$$

Scaling

Scaling transformation relative to the coordinate origin is now expressed as the matrix multiplication.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \text{.....(21)}$$

$$\boxed{P' = S(s_x, s_y).P} \quad \text{.....(22)}$$

In many systems, rotation and scaling functions produce transformations with respect to the coordinate origin.

2.7 Composite Transformation

The matrix representations can set up a matrix for any *sequence of transformations* as a **composite transformation matrix** by calculating the matrix product of the individual transformations. Forming products of transformation matrices is referred to as a **concatenation,** or **composition,** of matrices. Each successive transformation matrix pre-multiplies the product of the preceding transformation matrices.

Translation

Two successive translation vectors (t_{x1}, t_{y1}) and (t_{x2}, t_{y2}) are applied to a coordinate position P, the final transformed location P' is calculated as

$$\begin{bmatrix} 1 & 0 & t_{x2} \\ 0 & 1 & t_{y2} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & t_{x1} \\ 0 & 1 & t_{y1} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_{x1}+t_{x2} \\ 0 & 1 & t_{y1}+t_{y2} \\ 0 & 0 & 1 \end{bmatrix} \quad \text{.....(23)}$$

Or

$$\boxed{T(t_{x2}, t_{y_2}) \cdot T(t_{x1}, t_{y1}) = T(t_{x1}+t_{x2}, t_{y1}+t_{y2})} \quad \text{.....(24)}$$

Two successive translation operations are **additive.**

Rotation

Two successive rotation applied to point P produces the transformation position P'.

$$\begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_1 & -\sin\theta_i & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) & 0 \\ \sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{.....(25)}$$

Or

$$\boxed{R(\theta_2) \cdot R(\theta_1) = R(\theta_1+\theta_2)} \quad \text{.....(26)}$$

Two successive rotation operations are ***additive.***

Scaling

Concatenating transformation matrix for two successive scaling produces the following composite scaling matrix.

$$\begin{bmatrix} S_{x2} & 0 & 0 \\ 0 & S_{y2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} S_{x1} & 0 & 0 \\ 0 & S_{y1} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} S_{x1}.S_{x2} & 0 & 0 \\ 0 & S_{y1}.S_{y2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{.....(27)}$$

Or

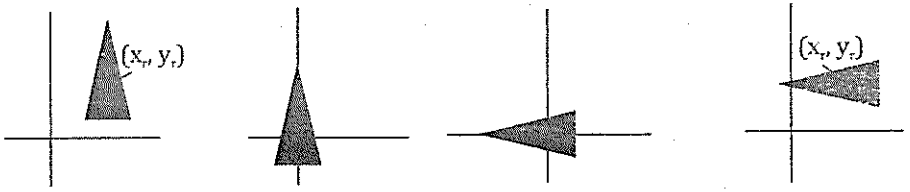
$$\boxed{S(s_{x2}, s_{y2}) \cdot S(s_{x1}, s_{y1}) = S(s_{x1} \cdot s_{x2}, s_{y1} \cdot s_{y2})} \quad \text{.....(28)}$$

Two successive scaling operations are ***multiplicative***

2.8 General Pivot Point Rotation

A graphics package that provides a rotate function, for rotating an object about the coordinate origin. We can generate rotations about any ***selected pivot point*** (x_r, y_r) by performing the following sequence of ***translate - rotate – translate*** operations:

- Translate the object so that the pivot-point position is moved to the coordinate origin.
- Rotate the object about the coordinate origin.
- Translate the object so that the pivot point is returned to its original position (Inverse translation).



(a) Original position of object and pivot point

(b) Translation of object so that pivot point (x, y) is at origin

(c) Rotation about origin

(d) Translation of object so that the pivot point is returned to position (x_r, y_r)

Figure 2.12 A transformation sequence for rotating an object about a specific pivot point

The composite transformation matrix for this sequence is obtained with the concatenation.

$$\begin{bmatrix} 1 & 0 & x_r \\ 0 & 1 & y_r \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_r \\ 0 & 1 & -y_r \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta & -\sin\theta & x_r(1-\cos\theta) + y_r \sin\theta \\ \sin\theta & \cos\theta & y_r(1-\cos\theta) - x_r \sin\theta \\ 0 & 0 & 1 \end{bmatrix}$$

this can be expressed in the form

$$T(x_r, y_r) \cdot R(\theta) \cdot T(-x_r, -y_r)$$

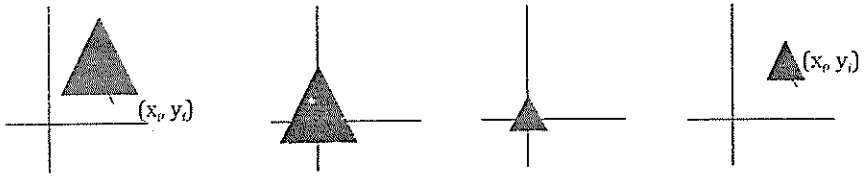
$$\boxed{= R(x_r, y_r, \theta)} \quad \text{.....(29)}$$

Where $T(-x_r, -y_r) = T^{-1}(x_r, y_r)$.

2.9 General Fixed Point Scaling

A transformation sequence to produce scaling with respect to a ***selected fixed position*** (x_f, y_f) using a scaling function that can only scale relative to the coordinate origin. We perform the following sequence ***translate - scaling – translate*** operations:

- Translate object so that the fixed point coincides with the coordinate origin.
- Scale the object with respect to the coordinate origin.
- Translate the object so that the pivot point is returned to its original position (Inverse translation).



(a) Original position of object and fixed point

(b) Translate object so that fixed point (x_f, y_f) is at origin

(c) Scale object with respect to origin

(d) Translate object so that the fixed point is returned to position (x_f, y_f)

Figure 2.13 A transformation sequence for scaling an object with respect to a specified fixed position (x_f, y_f)

The composite transformation matrix for this sequence is obtained with the concatenation of the matrices.

$$\begin{bmatrix} 1 & 0 & x_f \\ 0 & 1 & y_f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_f \\ 0 & 1 & -y_f \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} S_x & 0 & x_f(1-S_x) \\ 0 & S_y & y_f(1-S_y) \\ 0 & 0 & 1 \end{bmatrix}$$

$$T(x_f, y_f) \cdot S(s_x, s_y) \cdot T(-x_f, -y_f)$$

$$= S(x_f, y_f, s_x, s_y) \quad \text{.....(30)}$$

2.10 Raster Method for Transformation

Picture information is stored in raster systems as *pixel patterns* in the ***frame buffer***. Therefore, some transformations can be carried out by rapidly by moving rectangular arrays of stored pixel values from one location to another within the frame buffer.

Raster functions that manipulate rectangular pixel arrays are referred as ***raster ops***. Moving a block of pixels from one location to another is also called a *block transfer of pixel values.*

On a bi-level system, the operation is called a ***bitBlt*** (bit-block transfer), when the function is hardware implemented.

On multilevel systems (multiple bits per pixel), the term ***pixBlt*** is used for block transfers.

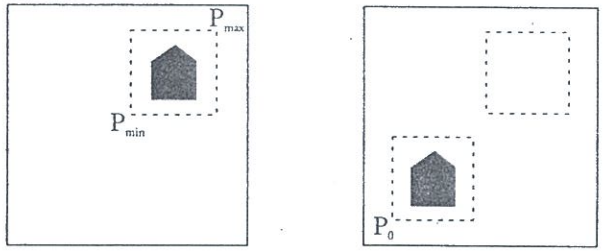


Figure 2.14 Translating an object from screen position (a) to position (b) by moving a rectangular block of pixel values.

In the above picture translation performed as a block transfer of a raster area. All bit settings in the rectangular area shown are copied as a block into another part of the raster.

This translation is followed by:

- First read pixel intensities from a rectangular area of a raster into an array.
- Copy the array back into the raster at the new location.
- The original object could be erased by filling its rectangular area with the background intensity. (Assuming the object does not overlap other objects in the scene).

Raster functions often provided in graphics packages are:

- **Copy** - move a pixel block from one raster area to another.
- **Read** - save a pixel block in a designated array.
- **Write** - transfer a pixel array to a position in the frame buffer.

Rotations in 90-degree increments are accomplished with block transfers.

Rotate an object 90 degree counterclockwise:

- First reversing the pixel values in each row of the array.
- Then interchange rows and columns.
- Rotate the object 180 degree by reversing the order of the elements in the array.
- Then reversing the order of the rows.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix} \quad \begin{bmatrix} 3 & 6 & 9 & 12 \\ 2 & 5 & 8 & 11 \\ 1 & 4 & 7 & 10 \end{bmatrix} \quad \begin{bmatrix} 12 & 11 & 10 \\ 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix}$$

(a) (b) (c)

Figure 2.15 Rotating an array of pixel values.

Figure 2.14 Rotating an array of pixel values is shown in fig 2.15.

- The original array.
- Array orientation after a 90 degree counterclockwise rotation.
- Array orientation after a 180 degree rotation.

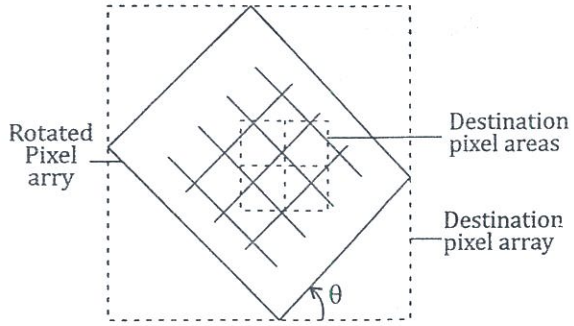


Figure 2.16 Raster rotation for a rectangular block of pixels

Other rotations are performed by first mapping rotated pixel areas onto destination positions in the frame buffer, then calculating overlap areas. Scaling in raster transformations is also accomplished by mapping transformed pixel areas to the frame-buffer destination positions.

2.11 Viewing Transformation

A mechanism for displaying a view of a picture on an output device. A graphics package allows a user to specify ***which*** part of the picture is to be displayed and ***where*** that part is to be placed on the display device.

For a 2 D picture, a view is selected by specifying a subarea of the total picture area. A user can select a single area for display, or several areas for simultaneous display.

The picture parts within the selected areas are then mapped onto specified areas of the device coordinates. When multiple view areas are selected, they are placed in separate display locations.

Transformations from *world to device coordinates* involve translation, rotation, and scaling operations, and procedures for deleting those parts of the picture that are outside the limits of a selected display area.

A world-coordinate area selected for display is called a **window**. An area on a display device to which a window is mapped is called a **viewport**. The window defines ***what*** is to be viewed; the viewport defines ***where*** it is to be display. Usually Windows and viewports are rectangles in standard position.

The mapping of a part of a world-coordinate scene to device coordinates is referred to as a **viewing transformation** or window-to-viewport transformation or windowing transformation.

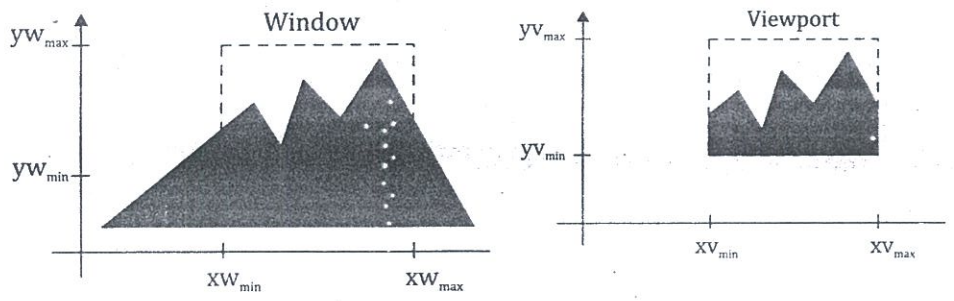


Figure 2.17 A Viewing Transformation using Rectangles for the Window and Viewport.

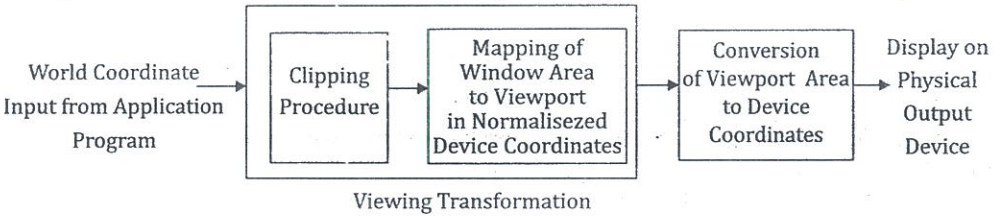


Figure 2.18 The 2 D viewing pipeline

- We construct the scene in world coordinates using the output primitives and attributes.
- To obtain a particular orientation for the window, we can set up a two-dimensional viewing-coordinate system in the world-coordinate plane, and define a window in the viewing-coordinate system.
- All parts of the picture that lie outside the viewport are clipped, and the contents of the viewport are transferred to device coordinates.

Clipping procedures are fundamental importance in computer graphics. They are used in drawing packages to eliminate parts of a picture inside or outside of a designated screen area, and in many other applications.

2-D Viewing Transformation

- Convert from Window Coordinates to Viewport Coordinates.
- (xw, yw) → (xv, yv).
- Maps a world coordinate window to a screen coordinate viewport.
- Window defined by: (xw1, yw1), (xw2, yw2).
- Viewport defined by: (xv1, yv1), (xv2, yv2).
- Basic idea is to maintain proportionality.

2.12 Window-to-viewport Coordinate transformation

A window is specified by four world coordinates xw_{min}, xw_{max}, yw_{min}, yw_{max}. Similarly a viewport is described by four normalized device coordinates xv_{min}, xv_{max}, yv_{min}, yv_{max}. The objective of the window to viewport mapping is to convert the world coordinate (xw, yw) of any point to its corresponding normalized device coordinates (xv,yv).

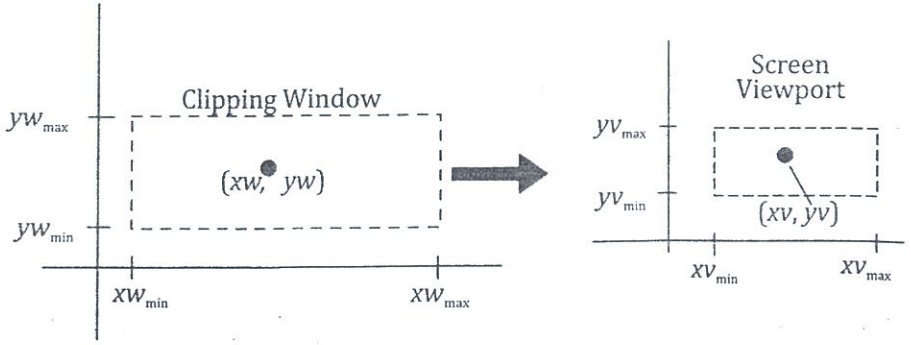


Figure 2.19 A point (xw, yw) in window is mapped to viewport at (xv, yv)

In order to maintain the same relative placement of the point in the viewport as in the window, we require

$$\frac{XV - XV_{min}}{XV_{max} - XV_{min}} = \frac{XW - XW_{min}}{XW_{max} - XW_{min}}$$

$$\frac{YV - YV_{min}}{YV_{max} - YV_{min}} = \frac{YW - YW_{min}}{YW_{max} - YW_{min}} \quad \text{.....(1)}$$

Solve these expression for the viewport position (xv, yv) ,We can rewrite above equation as:

$$XV = \frac{(XV_{max} - XV_{min})}{(XW_{max} - XW_{min})} * (XW - XW_{min}) + XV_{min}$$

$$\boxed{XV = XV_{min} + (XW - XW_{min}) * Sx} \quad \text{.....(2)}$$

Similarly for Y

$$YV = \frac{(YV_{max} - YV_{min})}{(YW_{max} - YW_{min})} * (YW - YW_{min}) + YV_{min}$$

$$\boxed{YV = YV_{min} + (YW - YW_{min}) * Sy} \quad \text{.....(3)}$$

Where $Sx = (XV_{max} - XV_{min}) / (XW_{max} - XW_{min})$

$Sy = (YV_{max} - YV_{min}) / (YW_{max} - YW_{min})$ .....(4)

The conversion is performed with the following sequence of transformation:

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

Relative proportion of objects are maintained if the scaling factor are the same (Sx = Sy). Otherwise, world objects will be stretched or contracted in either the x or y direction when displayed on the output device.

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.

2.13 Clipping Operations

Elimination of parts of scene outside a window or viewport is called clipping.

Any procedure that identifies those portions of a picture that are either inside or outside of a specified region of space is referred to as a ***clipping algorithm****, or simply* ***clipping****.*

Applications of Clipping are

- Extracting parts of a defined scene for viewing.
- Locating visible surfaces in 3D views.
- Anti-aliasing line segments or boundaries of objects.
- Creating objects using solid modeling procedures.
- Displaying a multiwindow environment.
- Drawing and pointing operations which allow parts of a picture to be selected for operations such as copying, moving, erasing or duplicating.
- The clip window can be a general polygon or any curved boundary.

Different types of clipping are

- Point Clipping
- Line Clipping
- Polygon Clipping
- Curve Clipping
- Text Clipping

2.14 Point Clipping

Assume that the clip window is a rectangle.

1. **Given:**
   - A Point (x, y)
   - Clipping rectangle (window or viewport)

     $(X_{min}, Y_{min}, X_{max}, Y_{max})$
2. **Point test:**

   If the inequality are satisfies

$$xw_{min} <= x <= xw_{max}$$
$$yw_{min} <= y <= yw_{max}$$ .....(5)

   then the point (x, y) lies inside the clip area.

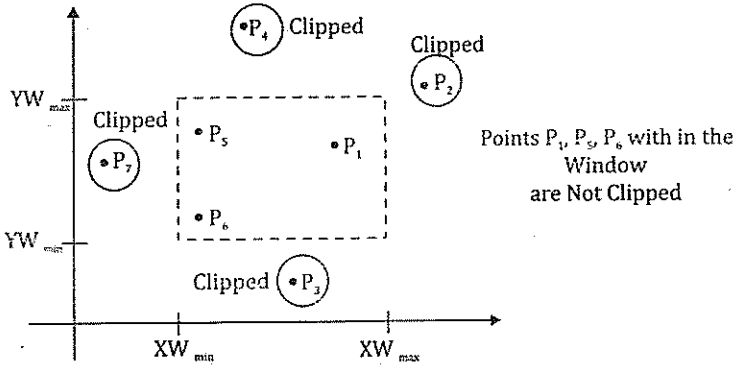


Figure 2.20 Point Clipping

If any of the four inequalities does not hold, the point is outside the clipping rectangle.

2.15 Line Clipping

A line clipping procedure:

- Test a given line segment whether it lies completely inside the clipping window.
- If it does not, determine whether it lies completely outside the window.
- If we cannot identify a line completely inside or completely outside, then perform intersection calculations with one or more clipping boundaries.
- We process the lines through the "inside-outside" tests by checking the line endpoints.

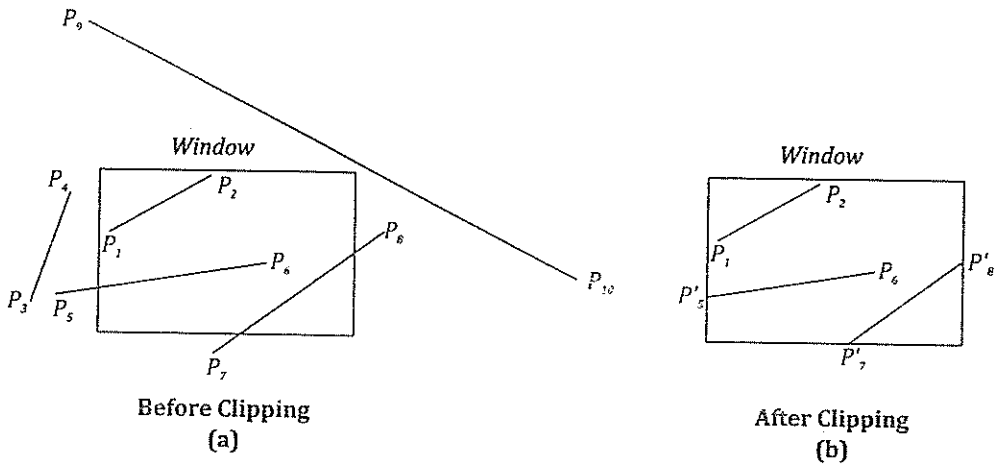


Figure 2.21 Line Clipping

- As in Fig 2.21 A line with both endpoints inside all clipping boundaries, such as the line from P1 to P2 is saved.
- A line with both endpoints outside any one of the clip boundaries like line P3, P4 which is outside the window, then the line is clipped.
- All other lines cross one or more clipping boundaries, and may require calculation of multiple intersection points.
- Find intersections with the rectangle boundaries using the parametric line:

$$x = x0 + u(x1 - x0)$$
$$y = y0 + u(y1 - y0) \quad \text{.....(6)}$$

Example

Intersection with the XWmin boundary:

$$XW_{min} = x_0 + u(x_1 - x_0)$$

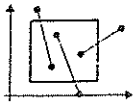
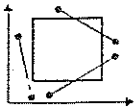
$$\text{where } u = \frac{XW_{min} - x0}{x1 - x0}$$

If $0 <= u <= 1$, intersection at XW_{min} and

$$Y' = y0 + \frac{XW_{min} - x0}{x1 - x0}(y1 - y0)$$

Check for intersections with the other boundaries based on the new line given by (XWmin, y0) and (x1, y1) (assuming (x0, y0) was outside).

Examine the end-points of each line to see if they are in the window or not

Situation	Solution	Example
Both end-points inside the window	Don't clip	
One end-point inside the window, one outside	Must clip	
Both end-points outside the window	Don't know!`	

Clipping line segments with these parametric **tests** require a good deal of computation, and faster approaches to clipping are possible.

2.15.1 Cohen - Sutherland Line Clipping

The **Cohen–Sutherland** algorithm is a *line clipping algorithm.* This is the efficient algorithm which performs initial tests on a line to determine whether intersection calculations can be avoided. The algorithm divides a 2D space into 9 regions, of which only the middle part (viewport) is visible.

- Every line end point in a picture is assigned a four-*digit binary code*, called a ***region code.***
- Region code identifies the location of the point relative to the boundaries of the clipping rectangle.

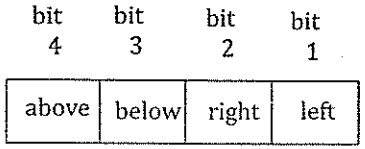


Figure 2.22 Region code

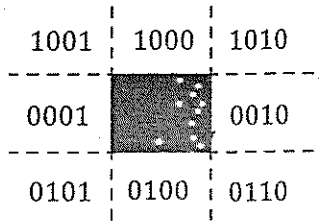


Figure 2.23 Binary Regional Codes assigned to the line end points

Observation - *All lines fall into one of three categories:*

1. Both endpoints lie inside the clip rectangle - Accept entire line.
2. Both endpoints outside clip rectangle on the same side of one of its borders - Reject entire line.
3. Neither 1 nor 2 - Clip part of line outside one of borders and repeat.

Steps for Cohen-Sutherland Algorithm

- A 4-bit region code number is assigned to an endpoint (x, y).
- Bit 1 is set to 1 if $x < xw_{min}$. Other three bits can be determined similarly.
- A value of 1 in any bit position indicates that the point is in the relative position; otherwise the bit position is set to 0.
- Once the region codes for all line endpoints are known, we can determine which line are completely inside the clip window or completely outside.
- The lines that are completely within the window boundaries have a region code of 0000 for both endpoints, we accept these lines.
- Any lines that have a 1 in the same bit position in the region codes for each endpoint are completely outside the clipping rectangle, we reject these lines.
- Lines are tested for total clipping by performing the logical AND operation with both region codes.
- If the result is not 0000, the line is completely outside the clipping region (Reject).
- If the result is 0000, the line is neither completely outside nor inside.
- So the algorithm checks the line endpoints against window boundaries in order left, right, bottom and top.
- Intersection points are calculated using equation parameters.
- For a line with endpoint (x_1, y_1) and (x_2, y_2), the y coordinate of the intersection point with a *vertical boundary* can be calculated as:

$$y = y_1 + m(x - x_1) \qquad \text{where, } x = xw_{min} \text{ or } xw_{max}$$

and slope of the line is $m = \dfrac{(y_2 - y_1)}{(x_2 - x_1)}$

Similarly the intersection point with a horizontal boundary, the x coordinates can be calculated as :

$$x = x_1 + \frac{(y - y_1)}{m} \qquad \text{where, } y = yw_{min} \text{ or } yw_{max}$$

Example

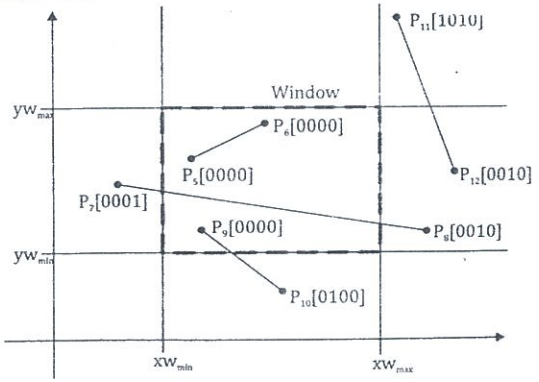


Figure 2.24 Line Clipping

- Every end-point is labelled with the appropriate region code.
- Lines completely contained within the window boundaries have region code 0000 for both end-points, so it is not clipped.
- Any lines with a common set bit in the region codes of both end-points can be clipped. The AND operation can efficiently check this.

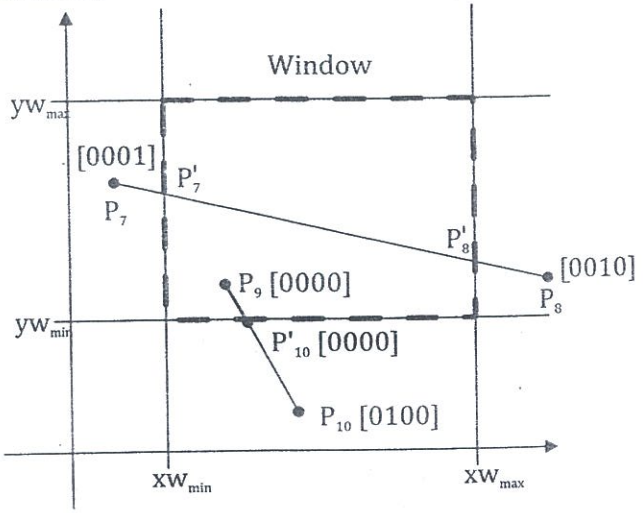


Figure 2.25 Clipping for line P_9 & P_{10}

- Consider the line P_9 to P_{10} in the fig 2.25
- Start at P_{10}
- From the region codes of the two end-points we know the line doesn't cross the left or right boundary
- Calculate the intersection of the line with the bottom boundary to generate point P_{10}'
- The line P_9 to P_{10}' is completely inside the window so it is ***Accept***.
- Then Consider the line P_7 to P_8 in the figure
- Start at P_7
- From the two region codes of the two end-points we know the line crosses the left boundary so calculate the intersection point to generate P_7'
- Consider the line P_7' to P_8
- Start at P_8
- Calculate the intersection with the right boundary to generate P_8'
- P_7' to P_8' is inside the window so is ***Accept***.

Example

$P_1(7, 2)$ $P_2(3, 6)$

$x_1 = 7$ $y_1 = 2$

$x_2 = 3$ $y_2 = 6$

Calculation:

$P_1(7, 2) P_2(3, 6)$

$x\omega_{min} = 4,\ x\omega_{max} = 8$
$y\omega_{min} = 3,\ y\omega_{max} = 6$ } Window boundry

Slope of line is

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{6-2}{3-7} = \frac{4}{-4} = 1$$

$\therefore m = 1$

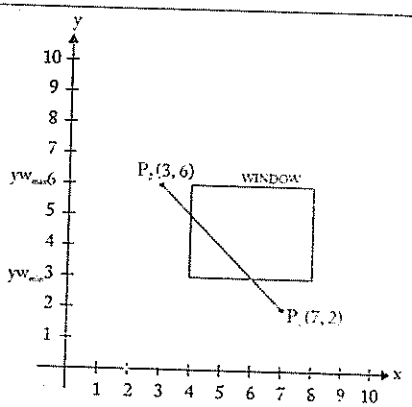


Figure 2.26 Before Clipping

Vertical Boundary

$y = y_1 + m(x - x_1)$ Where $x = x\omega_{min}$ (or) $x\omega_{max}$

Let $x = x\omega_{min} \Rightarrow x = 4$

Sub $x_1 = 7$, $y_1 = 2$ and $x = 4$ in eqn

$y = y_1 + m(x - x_1)$

$y = 2 + (-1)(4 - 7)$

$y = 2 + (-1)(-3)$

$\boxed{y = 5}$

$\therefore x = 4, y = 5 \Rightarrow \boxed{(4, 5)}$ intesecting with window boundary

Horizontal Boundary

$$x = x_1 + \frac{y - y_1}{m}$$

Let $y = y\omega_{min} \Rightarrow y = 3$

Sub $x_1 = 7, y_1 = 2, y = 3$

$$x = x_1 + \frac{y - y_1}{m} = 7 + \frac{(3-2)}{-1} = 7 + \frac{1}{-1} = -1 \qquad \boxed{\therefore x = 6}$$

$\therefore x = 6, y = 3 \Rightarrow (6, 3)$ intersecting with window boundary

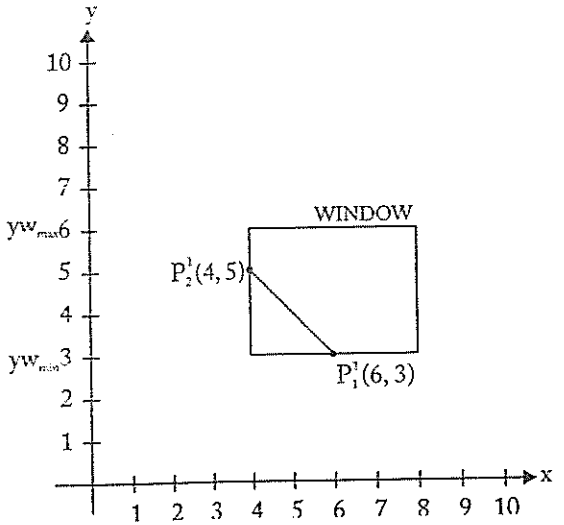


Figure 2.27 After Clipping

2.15.2 Midpoint Subdivision Algorithm

The line is divided at its midpoint into two shorter line segments. The clipping categories of the two new line segment are next computed using the region codes. Lines which cross one or two boundaries, we have to clip and fall in category 3. Category 1 lines are completely in the visible region and lines in category 2 are not visible and lie outside the window. Each segment in category 3 is again divided into shorter segments and categorized.

Category 1 : Lines are completely in the visible region.

Category 2 : Lines are not visible and lie outside the window

Category 3 : Line is again divided into shorter segments and categorized.

The bisection process and categorization is continued until each line segment that encompasses an intersection point across window boundaries 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 giving by $x_m = (x_1 + x_2)/2$ and $y_m = (y_1 + y_2)/2$. In the Fig. 2.28 we can determine the two intersection points M1 and M2 with 7 bisections using the midpoint subdivision algorithm.

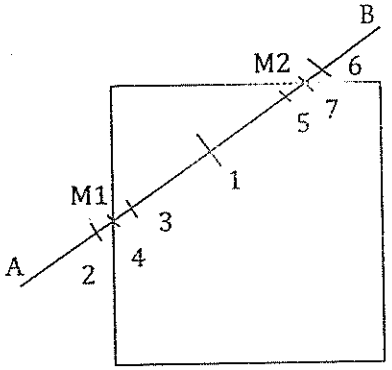
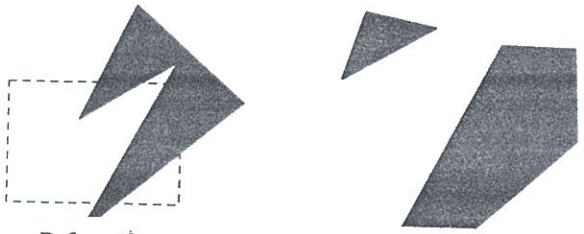


Figure 2.28 Midpoint subdivison

2.16 Area Clipping or Polygon Clipping

To clip polygons, we cannot use the line clipping algorithm, we need to modify the line-clipping procedures. A polygon processed with a line clipper may be displayed as a series of unconnected line segments.

For polygon clipping, we require an algorithm that will generate one or more closed areas that can fill the area. The output of a polygon clipper should be a sequence of vertices that defines the clipped polygon boundaries.



Before Clipping After Clipping

Figure 2.29 Display a clipped polygon

2.16.1 Sutherland-Hodgeman Polygon Clipping

Sutherland-Hodgeman clipping is used for polygon clipping. The algorithm operates on the vertices of the polygon. The simple problem is to clip a polygon against a single clipping edge. We clip the polygon on all four edges by clipping the entire polygon against one edge, then taking the resulting polygon and clipping against a second edge and so on for all four edges Fig 2.30

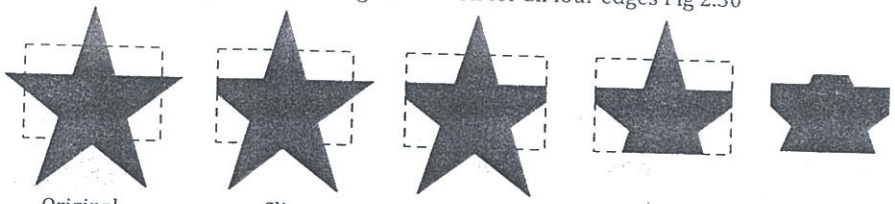


Figure 2.30 Clipping a Polygon against Successive Window Boundaries

There are four possible cases when processing vertices in sequence. Each pair of adjacent polygon vertices is passed to a window boundary clipper,

We make the following tests:

Case 1: If the first vertex is outside the window boundary and the second vertex is inside, both the intersection point of the polygon edge with the window boundary and the second vertex are added to the output vertex list.

Case 2: If both input vertices are inside the window boundary, only the second vertex is added to the output vertex list.

Case 3: If the first vertex is inside the window boundary and the second vertex is outside, only the edge intersection with the window boundary is added to the output vertex list.

Case 4: If both input vertices are outside the window boundary, nothing is added to the output list.

These four cases are illustrated in Fig. 2.31 for successive pairs of polygon vertices.

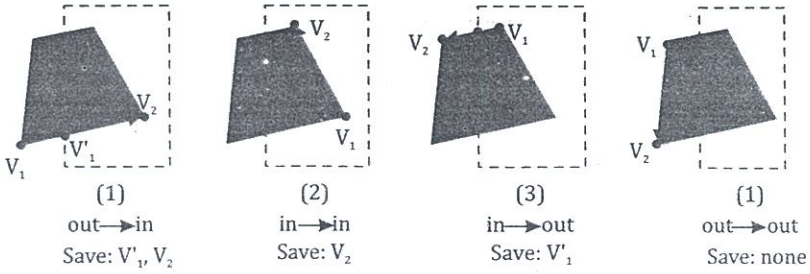


Figure 2.31 Successive processing a pairs of polygon vertices against the left window

Once all vertices have been processed for one clip window boundary, the output list of vertices is clipped against the next window boundary.

Example

We illustrate the example by processing the area against the left window boundary.

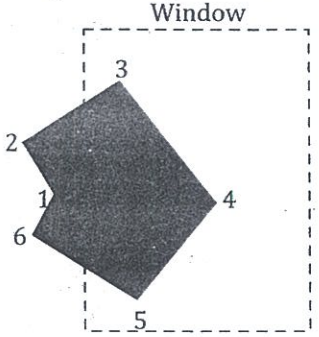


Figure 2.32 clipping a polygon against the left boundary

Traversal	Type	Action
Vertices 1 à 2	out - out	don't save
Vertices 2 à 3	in – out	save intersection point 1' and 2'
Vertices 3 à 4	in – in	save 3'
Vertices 4 à 5	in - in	save 4'
Vertices 5 à 6	in – out	save intersection point 5'
Vertices 6 à 1	out – out	don't save

Using the five saved points, repeat the process for the next window boundary.

2.17 Text Clipping

There are several techniques that can be used to provide text clipping in a graphics package. The different methods are:

- All-or-none string-clipping.
- All-or-none character clipping.
- Clip components of individual characters.

1. **All-or-none string-clipping:** If all of the string is inside a clip window we accept. Otherwise the string is discarded. This procedure is implemented by considering a boundary rectangle around the text pattern. The boundary positions of the rectangle are then compared to the window boundaries, and the string is rejected if there is any overlap.

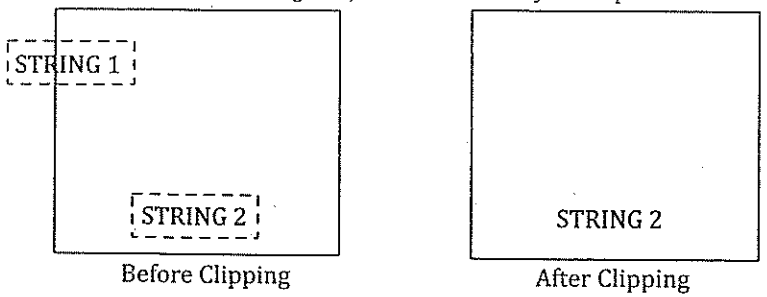


Figure 2.33 Text clipping using a boundary rectangle about the entire string

2. **All-or-none character clipping:** Here we discard only those characters that are not completely inside the window. In this case, the boundary limits of individual characters are compared to the window.Any character that either overlaps or is outside a window boundary is clipped.

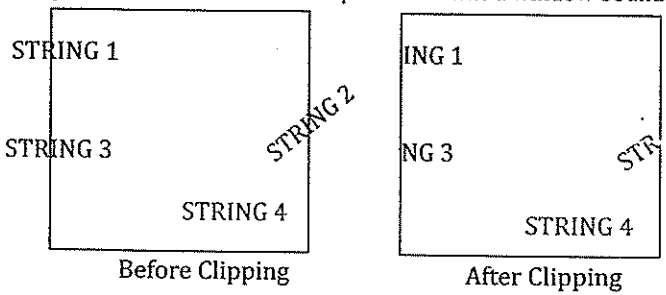


Figure 2.34 Text clipping using a boundary rectangle about individual characters

3. **Clip components of individual characters:**

In this method text clipping is to clip the components of individual characters. We now treat characters in much the same way that we treated lines. If an individual character overlaps a clip window boundary, we clip off the parts of the character that are outside the window.

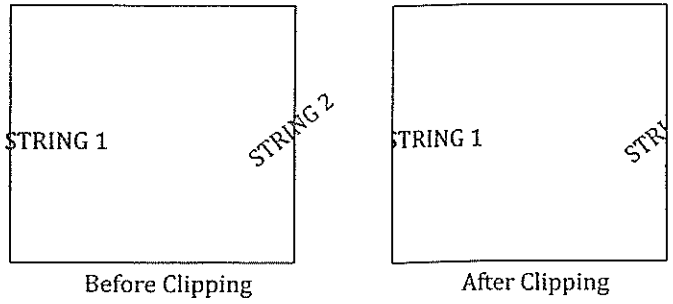


Figure 2.35 Text clipping on the components of individual characters

2.18 Curve Clipping

Areas with curved boundaries can be clipped. Curve-clipping procedures will involve nonlinear equations. The bounding rectangle for a circle or other curved object can be used first to test for overlap with a rectangular clip window. If the bounding rectangle for the object is completely inside the window, we save the object. If the rectangle is determined to be completely outside the window, we discard the object.

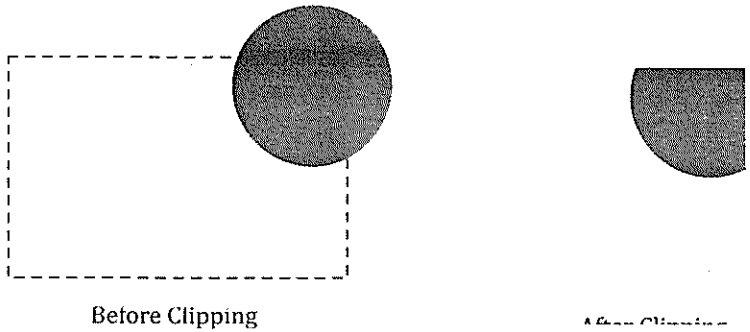


Figure 2.36 Clipping a filled circle

But if the bounding rectangle test fails, we can look for other computation-saving approaches. For a circle, we can use the coordinate extents of individual quadrants and then octants for preliminary testing before calculating curve-window intersections. For an ellipse, we can test the coordinate extents of individual quadrants.

2.19 Exterior Clipping

Till now procedures where used to clip a picture to the interior of a region by eliminating everything outside the clipping region. That is, we save only the **inside** 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.

Exterior clipping is used in other applications that require overlapping pictures. Examples are the design of page layouts in advertising or publishing applications or for adding labels or design patterns to a picture.

2.20 Review Questions

Short Answer Questions

1. Define 2 D Transformation?
2. What is translation?
3. What is rotation?
4. What is scaling?
5. Explain shear transformation?
6. Define reflection?
7. What is the need of homogeneous coordinates?
8. Distinguish between uniform scaling and differential scaling?
9. Successive 2 D scaling are multiplicative explain.
10. Distinguish between bitBlt and pixBlt?
11. Prove that successive 2D rotation are additive.
12. What do you mean by composite transformation? How it is useful?
13. Give a 3x3 transformation matrix to reduce an object to half its original size.
14. Explain about pivot point rotation.
15. Distinguish between window and view port?
16. Define clipping?
17. What is exterior clipping?
18. List out the various Text clipping?
19. Explain about clipping operations?
20. What do you main by interior and exterior clipping?
21. What is point clipping?
22. Explain about regional code.
23. How is curve clipping performed?

Long Answer Questions

1. Perform a 45 degree rotation of triangle A(0,0), B(1,1), C(5,2) about the origin Define
2. D transformation and explain with suitable illustrations translation, rotation.
3. Explain with suitable example uniform scaling.
4. Explain 2 D shear with suitable examples.

5. Discuss the raster method of transformation.
6. Explain two dimensional composite transformations?
7. What is 2 D reflection and explain in detail.
8. Give the sequence of steps that are required to rotate and scale an object through a general point. Also give its matrix representation.
9. Consider a polygon with 4 Coordinate Points (0, 0), (4, 0), (2, 3), (2, 1) with a scaling factor (s_x, s_y) as (0.5, 0.7) show how the object is scaled.
10. Explain the steps involved in fixed point scaling with example.
11. How is window-to-viewport transformation carried out.
12. Describe the Cohen Sutherland algorithm for line clipping with suitable illustrations.
13. Explain Sutherland Hodgeman polygon clipping
14. Explain about viewing transformation.
15. Explain the mid point subdivision method of clipping a line segment.
16. What are the different types of text clipping explain with example.

❑❑❑❑

NOTES

UNIT 3

3D GRAPHICS

CONTENTS

3.1 Three Dimensional Graphics

Three dimensional computer graphics are graphics that use a three-dimensional representation of geometric data that is stored in the computer for the purposes of performing calculations and rendering 2D images. In a three dimensional model we can view the structure from different viewpoint. 3D graphics programs allow objects to be created on an X-Y-Z scale (width, height, depth).

For example the architect can view the building structure from different angles or the automobile engineer can view the automobile design in different views. Every point in 3D is located using three coordinate values instead of two coordinates. In order to define points with three coordinate, we define a third axis, called the z-axis.

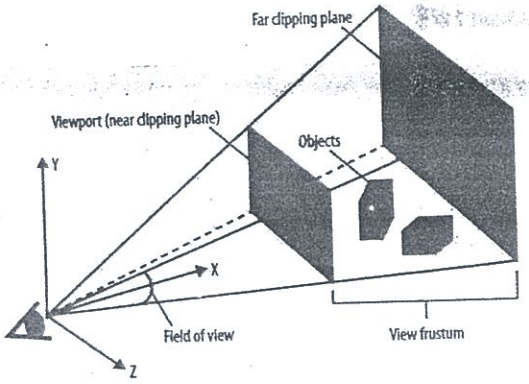


Figure 3.1 In 3D Graphics, Objects are created on a 3-Dimensional Stage where the current view is derived from the Camera Angle and Light Sources, similar to the Real World.

3.2 Three Dimensional Co-ordinate System

The conventional orientation for the coordinate axes in a 3D Cartesian reference system is in fig. 3.3(a). This is called a right-handed system because the right hand thumb points in the positive z-axis direction.

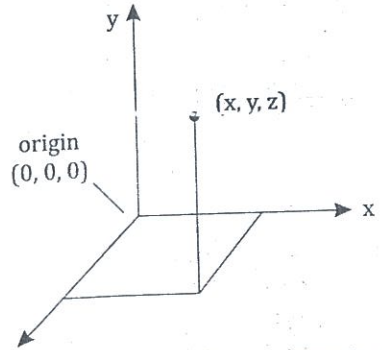


Figure 3.2 Coordinate representation of a point P at position(x, y, z) in Right Handed Cartesian reference system.

If the thumb of the right hand points in the positive Z direction, as one curls the fingers around the z-axis from the positive x-axis to the positive y-axis (90 degree) then the coordinates are called a **right handed system.**

If the thumb of the left hand points in the positive z direction when we imagine grasping the z-axis, so that the fingers of the left hand curl from the positive x-axis to the positive y-axis through 90 degree then the coordinates are called **left handed system.**

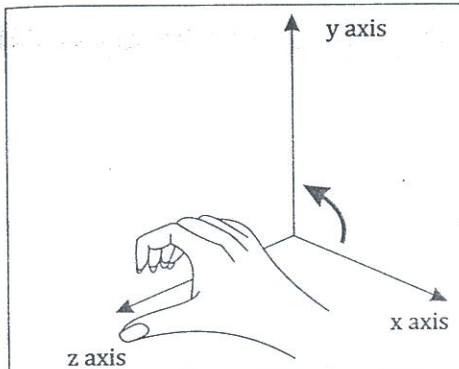


Figure 3.3 (a) Right Handed System

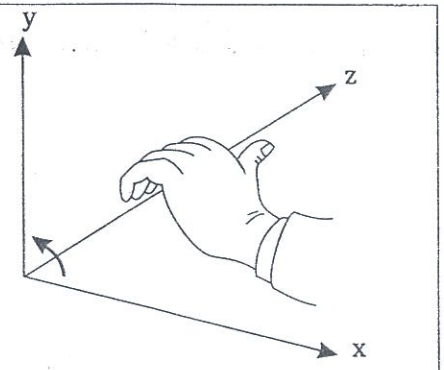


Figure 3.3 (b) Left Handed System

Normally the right handed system is used in mathematics but in computer graphics the left handed system is used since objects behind the display screen will have positive value.

3.3 Three Dimensional Display Techniques

On a graphics display it is impossible to produce an image that is perfectly realistic representation of an actual scene.

Figure 3.4 Three Dimensional View of an Object.

The techniques that should be taken into account:

- The different kinds of aspects needed by the application.
- The amount of processing required to generate the image.
- The capability of the display hardware.
- The amount of detail recorded in the image.
- The perceptual effects of the image on the observer.

The technique for achieving realism in three dimensional graphics is ***projection***. Projection can be defined as a mapping of a point P (x, y, z) onto its image P' (x', y', z') in the projection plane or view plane. Two types of projection Parallel and perspective.

The classification of projections is:

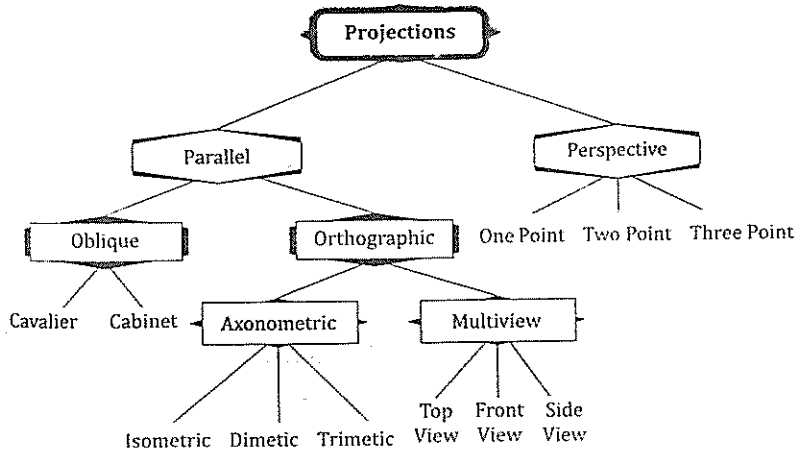


Figure 3.5 Classification of Projections

3.3.1 Parallel Projection

Parallel projection involved generating a view of a solid object by *projecting points* on the object surface along *parallel lines* onto the display plane. By selecting different viewing positions, we can project visible points of the object on the display plane and obtain different two dimensional views of the object.

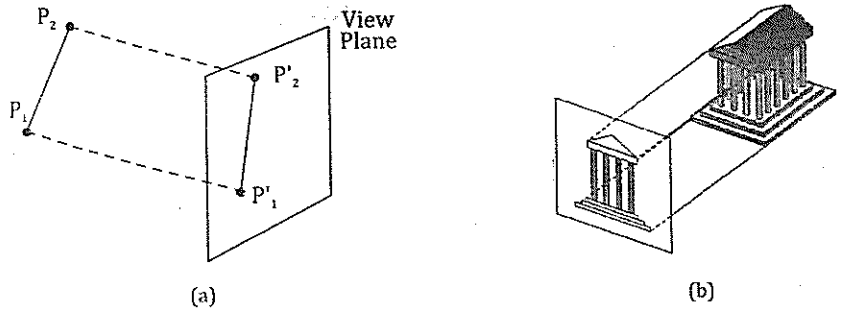


Figure 3.6 Parallel Projection of an Object to the View Plane.

In parallel projection parallel lines in the world coordinate scene, project parallel lines on the 2D display plane. The parallel projection technique is used in engineering and architectural drawings to represent an object with a set of views.

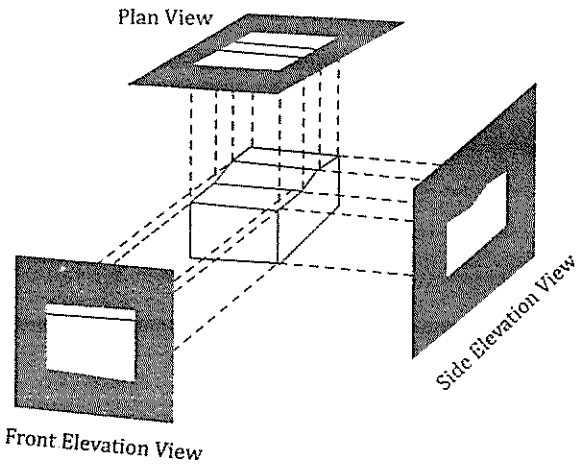


Figure 3.7 Three Parallel-Projection Views of an Object, different Viewing Positions

Parallel projection preserves relative proportions of 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 3D object.

3.3.2 Perspective Projection

For a perspective projection, object positions are transformed to the view plane along lines that *converge* to a point called projection reference point.

This causes objects far from the viewing position to be displayed smaller than the original objects. When the object is closer the size is the same. For example, aeroplane and a Ship.

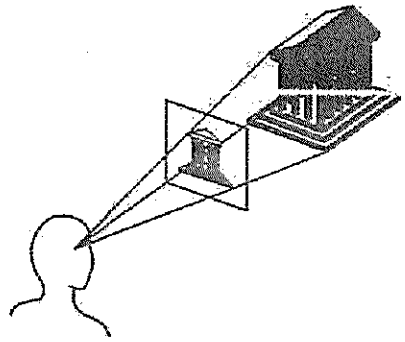


Figure 3.8 Perceptive Projection of an Object.

In perspective projection, parallel lines in a scene that are not parallel to the display plane are projected into converging lines. Scenes displayed using perspective projection appears more realistic. Since this is the way that our eyes and camera lens form images.

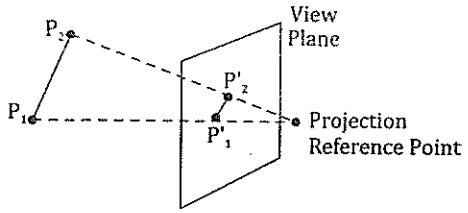


Figure 3.9 (a) Perspective Projection of an Object to the View Plane.

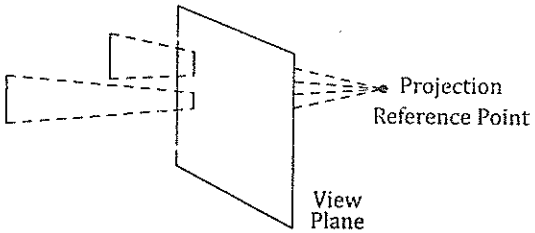


Figure 3.9 (b)Perspective Projection of Equal Size Objects at Different Distance from the View Plane

Object positions are transformed to the view plane along lines that converge to a point. In perspective projection, the further an object is from the viewer, the smaller it appears. This provides the viewer with *depth cue*, indicating which part of the image is closer and which part is farther from the user. The lines of projection are not parallel. They all converge at a single point called the center of projection and the intersection of these converging lines (Fig 3.8).

3.3.3 Orthogonal Projections (Orthgraphic Projection)

A technical drawing known as an orthographic projection shows several perspectives of an item projected on various reference planes while being observed perpendicular to the corresponding reference plane. The orthographic projection is a form of parallel projection in which all the projection lines are orthogonal to the projection plane, resulting in every plane of the scene appearing in affine transformation on the viewing surface. Orthographic projections are most often used to procedure the front, side, and top views of an object, which are called evaluations. The different reference planes are Horizontal Plane (HP), Vertical Plane (VP) and Side or Profile Plane (PP). The different views are Front View (FV) which is projected on vertical projection (VP), the Top View (TV) which is projected on horizontal projection (HP) and the Side View (SV) which is projected on profile plane (PP). When geometric objects are formed by the intersection of lines with a plane, the plane is called the projection plane and the lines are called projections. Orthographic parallel projections are done by projecting points along parallel lines that are perpendicular to the projection line. Perspective projection is another type of projection in computer graphics. It is the result of projecting an object onto a picture plane from a fixed point. Perspective projection is used to create a sense of depth and distance in a 2D image.

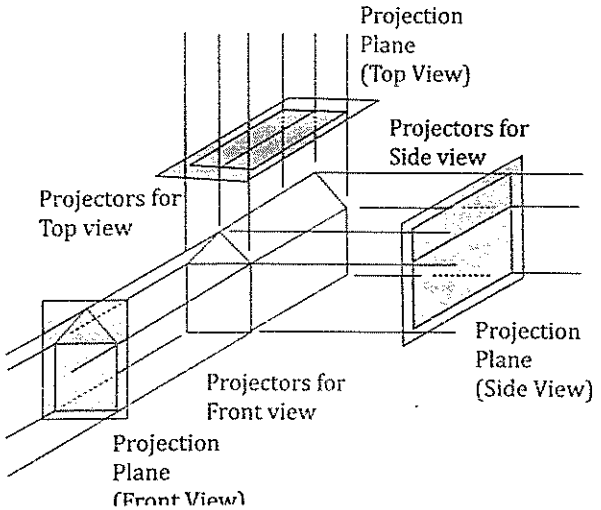


Figure 3.10 Orthographic projections of an object, displaying plane and elevation views

Engineering and architectural drawings commonly employ this orthographic projection, because lengths and angles are accurately depicted and can be measured from the drawings. We can also form orthographic projections that display more than one face of an object. Such views are called axonometric orthographic projections. The most commonly used axonometric projection is the isometric projection. We generate an isometric projection on by aligning the projection plane so that it intersects each coordinate axis in which the object is defined (called the principal axes) at the same distance from the origin. In the Figure 3.10, Orthographic projections of an object, displaying plane and elevation views is shown.

3.3.4 Intensity Cueing or Depth Cueing

Depth information is important so that we can easily identify, for a particular viewing direction, which is the front and which is the back of displayed objects. There are several ways in which we can include depth information in the two-dimensional representation of solid objects.

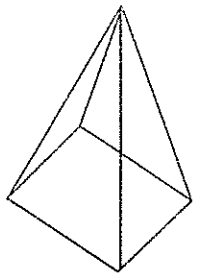


Figure 3.11 Depth Cueing

A simple method for indicating depth with wireframe displays is to *vary the intensity of objects* according to their distance from the viewing position. The lines closest to the viewing position are displayed with the highest intensities, and lines farther away are displayed with decreasing intensities. Depth cueing is applied by choosing maximum and minimum intensity (or color) values and a range of distances over which the intensities are to vary.

3.3.5 Visible Line and Surface Identification

Depth relationships in a wireframe display by identifying visible lines in some way. The simplest method is to highlight the visible lines or to display them in a different color. Another technique, commonly used for engineering drawings, is to display the non-visible lines as dashed lines. Another approach is to simply remove the non-visible lines. But removing the hidden lines also removes information about the shape of the back surfaces of an object.

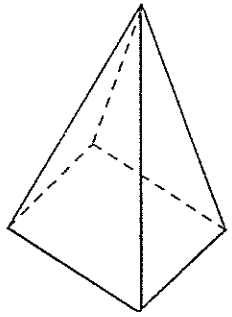


Figure 3.12 Hidden Surface Removal

3.3.6 Surface Rendering

When objects are to be displayed with color or shaded surfaces, we apply surface-rendering procedures to the visible surfaces so that the hidden surfaces are obscured. Lighting specifications include the intensity and positions of light sources and the general background illumination required for a scene. Procedures can then be applied to generate the correct illumination and shadow regions for the scene.

3.3.7 Exploded and Cutaway Views

Many graphics packages allow objects to be defined as hierarchical structures, so that internal details can be stored. Exploded and cutaway views of such objects can then be used to show the internal structure and relationship of the object parts. An alternative to exploding an object into its component parts is the cutaway view, which removes part of the visible surfaces to show internal structure.

3.3.8 Three Dimensional and Stereoscopic Views

To show realism to a computer-generated scene is to display objects using either three-dimensional or stereoscopic views. Three-dimensional views can be obtained by reflecting a raster image from a vibrating flexible mirror. The vibrations of the mirror are synchronized with the display of the scene on the CRT. As the mirror vibrates, the focal length varies so that each point in the scene is projected to a position corresponding to its depth.

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

3.3.9 Difference between Parallel and Perspective Projection

- **Parallel Projections:**
  - The center of projection is at infinity.
  - The projectors are parallel to each other.
  - Less realistic view because of no foreshortening.
  - Preserves the parallel lines.
- **Perspective Projections:**
  - The center of projection is a finite point.
  - The projectors intersect at the center of projection.
  - Visual effect is similar to human visual system which has 'perspective foreshortening'.
  - Does not preserve the parallel lines.

3.4 Three Dimensional Transformation

Three dimensions transformation are extended from two-dimensional methods by including considerations for the z - coordinate.

We translate an object by specifying a three-dimensional translation vector, which determines how much the object is to be moved in each of the three coordinate directions. Similarly we scale an object with three coordinate scaling factors. In two-dimensional rotations we consider only rotations about axes that were perpendicular to the *xy* plane. In three-dimensional rotation we can select any spatial orientation for the rotation axis. Most graphics packages handle three-dimensional rotation as a composite of three rotations, one for each of the three Cartesian axes.

3.4.1 Translation

In a three-dimensional homogeneous coordinate representation, a point is translated from position $P = (x, y, z)$ to position $P' = (x', y', z')$ with the matrix operation

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad \ldots(1)$$

Or $\boxed{P' = T.P}$ ...(2)

Parameters t_x, t_y, and t_z specifying translation distances for the coordinate directions *x*, *y*, and *z*, are assigned any real values. The matrix representation for equation (1) is

$$x' = x + t_x$$
$$y' = y + t_y$$
$$z' = z + t_z \quad \text{...(3)}$$

An object is translated in three dimensions by transforming each of the defining points of the object.

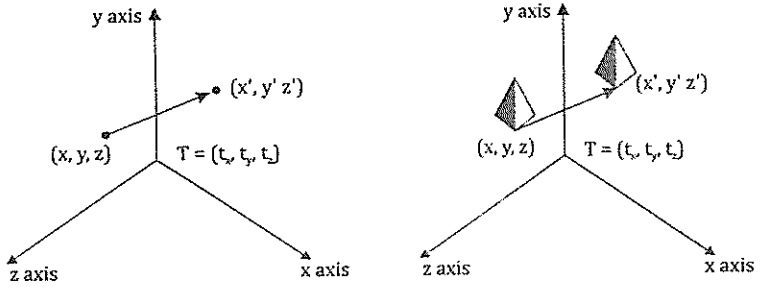


Figure 3.13 (a) Translating a point P to P'

(b) Translating an object with translation Vector T = (t_x, t_y, t_z)

3.4.2 Rotation

To rotate an object in three dimensional transformations we must decide two things:

- An axis of rotation.
- The angle θ to be rotated.

In two-dimensional applications, all transformations are carried out in the xy plane a three-dimensional rotation can be specified around any line in space. The easiest rotation axis are those that are parallel to the coordinate axis.

Positive rotation angles produce counter clockwise rotations about a coordinate axis, if we are looking along the positive half of the axis toward the coordinate origin.

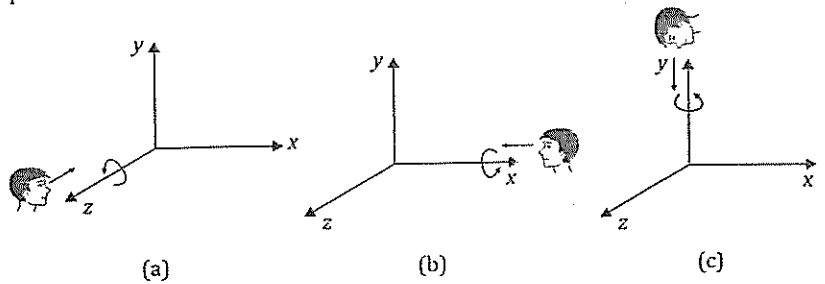


Figure 3.14 Positive Rotation about the Coordinate Axis.

Coordinate axis Rotation:

- Z – axis rotation
- X – axis rotation
- Y – axis rotation
  - **Z – axis rotation:**

The two-dimensional z-axis rotation equations are easily extended to three dimensions.

Transformation equation along z- axis:

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$
$$z' = z \quad \text{...(4)}$$

Parameter θ specifies the rotation angle.

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad \text{...(5)}$$

$$\boxed{P' = R_x(\theta).P} \quad \text{...(6)}$$

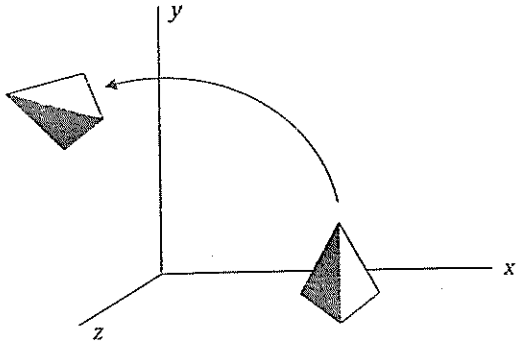


Figure 3.15 Rotation about z – axis

Transformation equations for rotations about the other two coordinate axis can be obtained with a **cyclic permutation** of the coordinate parameters x, y, and z in equation (4) we use the replacements as :

$$x \to y \to z \to x \quad \text{...(7)}$$

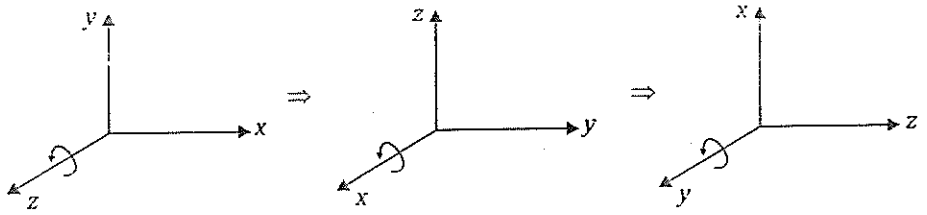


Figure 3.16 Cyclic Permutation of Cartesian Coordinate Axis

- **X – axis Rotation:**

Substituting cyclically permuting coordinates in equation (4) give us the transformation equations for a **x-axis rotation.**

$$y' = y\cos\theta - z\sin\theta$$
$$z' = y\sin\theta + z\cos\theta$$
$$x' = x \qquad \text{...(8)}$$

the matrix form is

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

$$\boxed{P' = R_x(\theta).P} \qquad \text{...(9)}$$

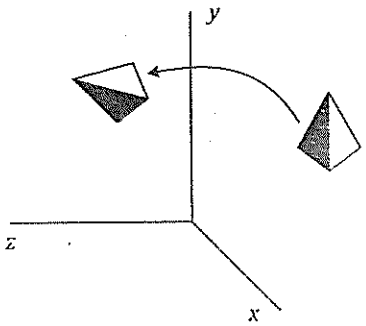


Figure 3.17 Rotation about x – axis

- **Y – axis Rotation:**

Substituting cyclically permuting coordinates in equation (8) give us the transformation equations for a **y-axis rotation.**

$$z' = z\cos\theta - x\sin\theta$$
$$x' = z\sin\theta + x\cos\theta$$
$$y' = y \qquad \text{...(10)}$$

the matrix form is

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad \text{...(11)}$$

$$\boxed{P' = R_y(\theta).P} \qquad \text{...(12)}$$

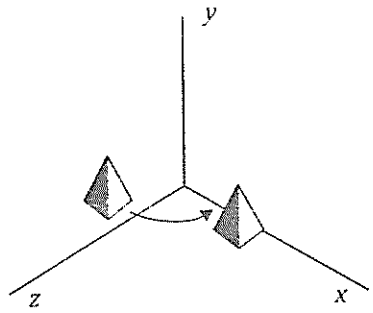


Figure 3.18 Rotation about y – axis

3.4.3 Scaling

The matrix expression for the scaling transformation of a position P = (x, y, z) relative to the coordinate origin can be written as

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad \text{...(13)}$$

Or

$$\boxed{P' = S \cdot P} \qquad \text{...(14)}$$

where scaling parameters s_x, s_y, and s_z are assigned any positive values.

$$x' = x.s_x$$
$$y' = y.s_y$$
$$z' = z.s_z \qquad \text{...(15)}$$

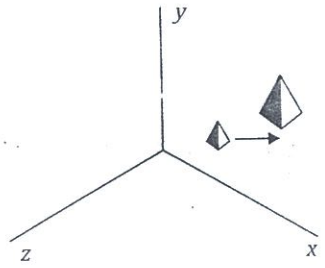


Figure 3.19 Scaling an 3D Object

Scaling an object with transformation changes the size of the object and repositions the object relative to the coordinate origin.

Scaling with respect to a selected fixed position (x, y, z) can be represented with the following transformation sequence:

1. Translate the fixed point to the origin.
2. Scale the object relative to the coordinate origin.
3. Translate the fixed point back to its original position.

This sequence of transformations is demonstrated in Fig. 3.20

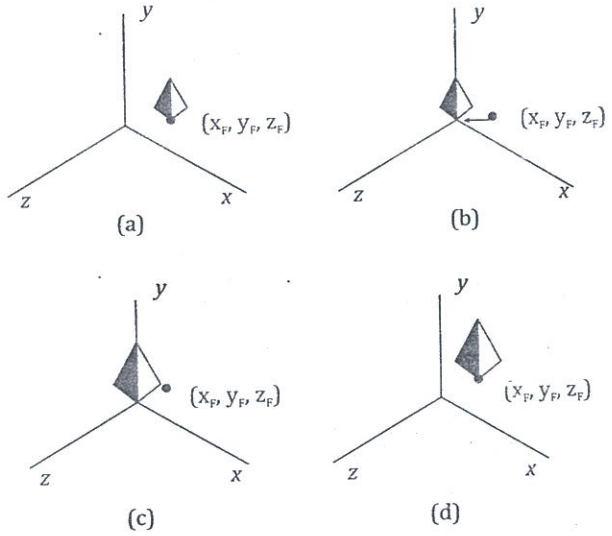


Figure 3.20 Scaling an Object relative to a Fixed Point

The matrix representation for an arbitrary fixed-point scaling can be expressed as the concatenation of these *translate-scale-translate* transformations as

$$T(x_f, y_f, z_f).S(s_x, s_y, s_z).T(-x_f, -y_f, -z_f) = \begin{bmatrix} S_x & 0 & 0 & x_f(1-S_x) \\ 0 & S_y & 0 & y_f(1-S_y) \\ 0 & 0 & S_z & z_f(1-S_z) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3.4.4 Reflection

A three-dimensional reflection can be performed relative to a selected reflection axis or with respect to a selected reflection plane. In general, three-dimensional reflection matrices are set up similarly to those for two dimensions. Reflections relative to a given axis are equivalent to 180-degree rotations about that axis. Reflections with respect to a plane are equivalent to 180-degree rotations in four-dimensional space. When the reflection plane is a coordinate plane (either xy, xz, or yz), the transformation can be a conversion between Left-handed and right-handed systems.

An example of a reflection that converts coordinate specifications from a right-handed system to a left-handed system (or vice versa) as shown in figure 3.21.

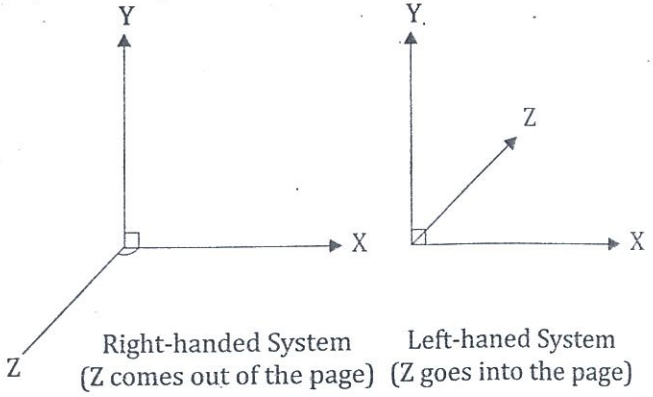


Figure 3.21: Conversion of Coordinate Specifications from a Right-handed to a Left-handed System can be carried out with the Reflection Transformation

This transformation changes the sign of the z coordinates, Leaving the x and y-coordinate values unchanged. The matrix representation for this reflection of points relative to the xy plane is

$$RF_z = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformation matrices for inverting x and y values are defined similarly, as reflections relative to the yz plane and xz plane, respectively. Reflections about other planes can be obtained as a combination of rotations and coordinate-plane reflections.

3.5 Polygon Surfaces

The boundary representation for a three-dimensional object is a *set of surface polygons* that enclose the object interior. Graphics systems store all object descriptions as sets of surface polygons. This simplifies and speeds up the surface rendering and display of objects. All surfaces are described with linear equations.

A polygon representation for a polyhedron (it is a closed polygonal mesh) precisely defines the surface features of the object. But for other objects, surfaces are *tessellated* (or tiled) to produce the polygon-mesh. The surface of an object is represented as a polygon mesh. (The wire frame model is called a polygonal net or polygonal mesh.)

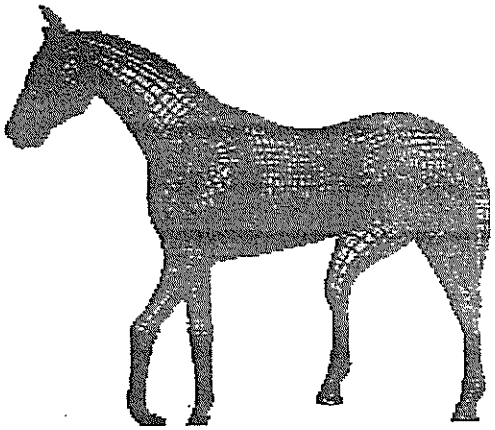


Figure 3.22 Wire-frame Model

The wireframe outline can be displayed quickly to give a general indication of the surface structure. The polygon-mesh can be improved by dividing the surface into smaller polygon.

Polygon Tables

The graphics package organizes the polygon surface data into tables. The table may contain geometric, topological and attribute properties.

As information for each polygon is input, the data are placed into tables that are to be used in the subsequent processing, display, and manipulation of the objects in a scene.

Polygon data tables can be organized into two groups:

- Geometric tables.
- Attribute tables.
- **Geometric data tables:**
  - It contain vertex coordinates and parameters to identify the orientation of the polygon surfaces.
  - To stored geometric data three lists is created:
    1. A vertex table
    2. An edge table.
    3. A polygon table.
  - *Vertex table* stores the coordinate values of each vertex in the object.
  - *Edge table* consists of a pointer to each endpoint of that edge.
  - *Polygon table* defines a polygon by providing pointers to the edges that make up the polygon.

VERTEX TABLE
V_1: x_1, y_1, z_1
V_2: x_2, y_2, z_2
V_3: x_3, y_3, z_3
V_4: x_4, y_4, z_4
V_5: x_5, y_5, z_5

EDGE TABLE
E_1: V_1, V_2
E_2: V_2, V_3
E_3: V_3, V_1
E_4: V_3, V_4
E_5: V_4, V_5
E_6: V_5, V_2

POLYGON SURFACE TABLE
S_1: E_1, E_2, E_3
S_2: E_2, E_4, E_5, E_6

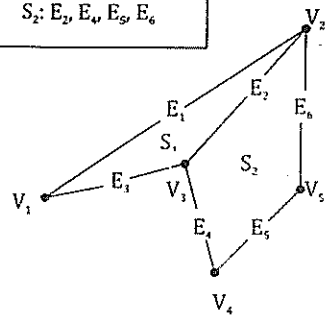


Figure 3.23 Geometric data table representing two adjacent polygon surfaces

The geometric data in three tables provides a convenient reference to the individual components (vertices, edges, and polygons) of each object.

We can add extra information to the data table of Fig 3.23 for fast information extraction.

For example, we could expand the edge table to include forward pointers into the polygon table so that *common edges* between polygons could be identified more rapidly, which is useful in rendering procedures for varying surface shading smoothly.

Edge Table

E_1 : V_1, V_2, S_1
E_2 : V_2, V_3, S_1
E_3 : V_3, V_1, S_1, S_2
E_4 : V_3, V_4, S_2
E_5 : V_4, V_5, S_2
E_6 : V_5, V_1, S_2

Figure 3.24 Edge Table for the Surfaces of Fig. 3.21

Attribute data table: Contains information for an object and parameters specifying the degree of transparency of the object and its surface reflectivity and texture characteristics

3.6 Octrees

Octrees are hierarchical tree structures used to represent solid objects in graphics systems.An octree is a tree data structure in which each internal node has exactly eight children. Octrees are most often used to partition a three dimensional space by recursively subdividing it into eight octants. Octree representations are used in Medical imaging and other applications for an object cross section view.

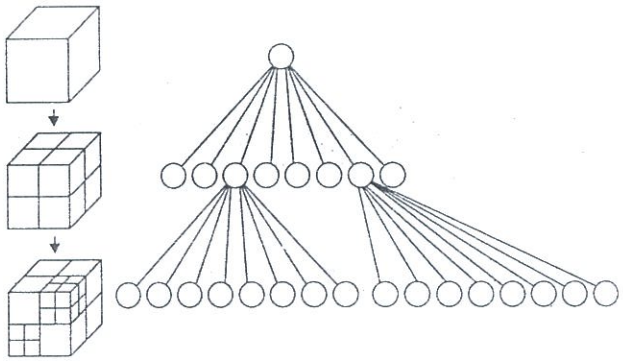
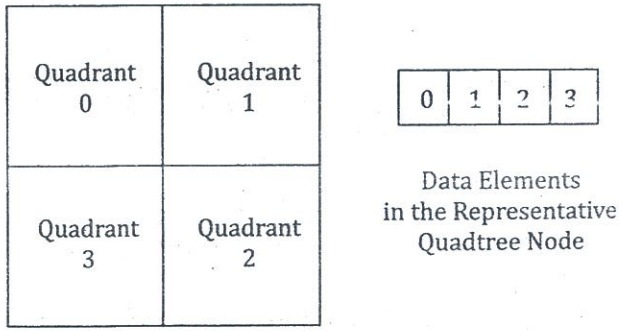


Figure 3.25 Left: Recursive Subdivision of a Cube into Octants.
Right: The Corresponding Octree.

Quadtree Encoding

The octree encoding procedure for a 3D space is an extension of an encoding scheme for 2D space, called quadtree encoding.

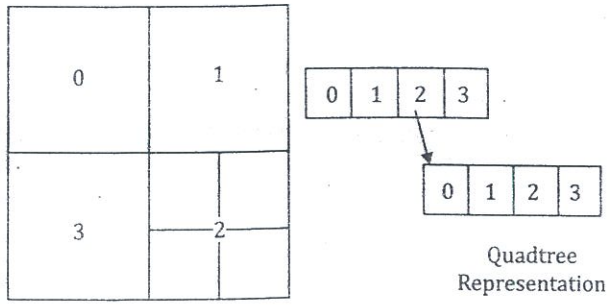
- Quadtrees are dividing a two-dimensional region (usually a ***square***) into quadrants.
- Each node in the quadtree has *four data elements*, one for each of the quadrants in the region.
- If all pixels within a quadrant have the same color (a *homogeneous* quadrant), the corresponding data element in the node stores that color.
- Suppose all pixels in quadrant 2 of Fig. 3.26 are found to be red. The color code for red is then placed in the data element 2 of the node.



Region of a Two-DimensionalSpace

Figure 3.26 Region of Two Dimensional Space divided into Numbered Quadrants.

- Otherwise, the quadrant is said to be *heterogeneous*, and that quadrant is itself divided into quadrants, Fig. 3.27.



Region of a Two-Dimensional Space

Figure 3.27 Region of two dimensional space with two levels of quadrant division.

- The corresponding data element in the node now flags the quadrant as heterogeneous and stores the pointer to the next node in the quadtree.
- An algorithm for generating a quadtree tests pixel-intensity values and sets up the quadtree nodes accordingly.
- If each quadrant in the original space has a single color specification, the quadtree has only one node.
- For a heterogeneous region of space, the successive subdivisions into quadrants continue until all quadrants are homogeneous.
- Fig: 3.28 shows a quadtree representation for a region containing one area with a solid color that is different from the uniform color specified for all other areas in the region.
- Quadtree encodings provide considerable *savings in storage* when large color areas exist in a region of space

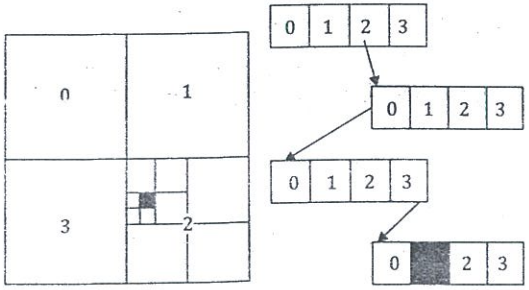


Figure 3.28 Quadtree Representations for a Region containing One Foreground Color Pixel on a Solid Background.

Octree Encoding:

An octree encoding scheme divides regions of three-dimensional space (usually ***cubes***) into octants and stores eight data elements in each node of the tree (Fig. 3.29).

Each elements of a three-dimensional space are called **volume elements**, or **voxels.** When all voxels in an octant are of the same type, this type value is stored in the corresponding data element of the node. Empty regions of space are represented by voxel type "void." Any heterogeneous octant is subdivided into octants, and the corresponding data element in the node points to the next node in the octree. Procedures for generating octrees are similar to those for quadtrees: Voxels in each octant are tested, and octant subdivisions continue until the region of space contains only homogeneous octants. Each node in the octree can now have from zero to eight immediate descendants.

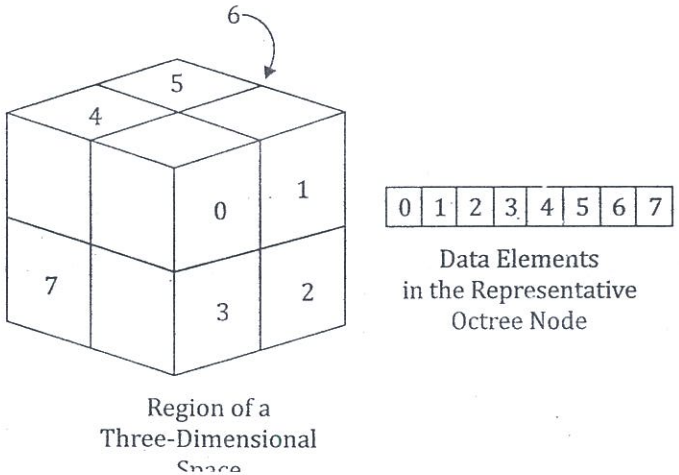


Figure 3.29 Region of a Three Dimensional Space divided into Numbered Octants.

Algorithm for Generating Octrees:

Step 1 : Accept object definitions. (such as a polygon mesh or solid geometry constructions)

Step 2 : Use the minimum and maximum coordinate values of the object.

Step 3 : The object in the three-dimensional space is tested, octant by octant, to generate the octee representation.

Step 4 : The established octree representation for the solid object can be applied with various manipulation routines.

Step 5 : Set operations – Union, Intersection or Difference can be applied to two octree representation.

Step 6 : A union operation, a new octree is constructed with the combined regions of both objects. Intersection operations are performed by selecting the common regions of overlap in the two octrees. Difference operation gives rise to the region occupied by one object but not the other.

Visible-surface or hidden surface identification is carried out by searching the octants from front to back. The first object detected is visible, so that information can be transferred to a quadtree representation for display.

3.7 Curves and Surfaces

Smooth curves and surfaces must be generated in many computer graphics applications. Special techniques are used to model the object to generate realistic images. There are several approaches to modeling curved surfaces. One is an analog of polyhedral models. Instead of using polygons, we model an object by using small curved surface patches placed next to each other. Another approach is to use surfaces that define solid objects, such as sphere cylinders and cones. A model can be constructed with these solid objects using building blocks. This process is called solid modeling.

Properties for designing curves:

1. **Control Point:** Control point or knots is a way to control the shape of a curve. It is to locate points through which the curve must pass or point that control the curve's shape in a predictable way. A curve is said to interpolate the control points if it passes through them.

Figure 3.30 Control Points Indicate by Dots

2. **Multiple Values:** In general, a curve is not a graph of a single valued function, irrespective of choice of coordinate system.

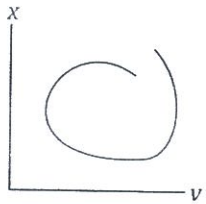


Figure 3.31 A Curve may be Multivalued with all Coordinate System

3. **Axis Independence:** The shape of an object must not change when the control points are calculated in a different coordinate system. For example, if the control points are rotated 90 degrees, the curve should rotate 90 degree but not change shape.
4. **Global or Local Control:** As a designer manipulates a control point, a curve may change shape only in the region near the control point, or change shape throughout.

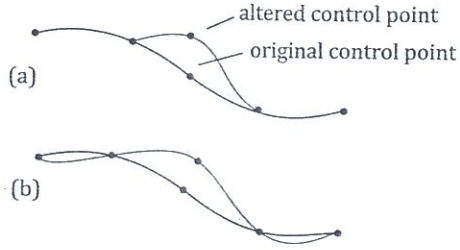


Figure 3.32 (a) Curves having Local Control Point (b) Curves with Global Control Point

5. **Variation-Diminishing Property:** A curve that oscillates about its control points is usually undesirable. Variation diminishing curves tend to smooth out a sequence of control points.
6. **Versatility:** A curve representation that allows limited variety of shapes. A framework that provides only arcs of circle. The flexible technique for the designer to control the versatility of a curve representation is by adding or removing control points.
7. **Order of Continuity:** A complex shape is created by joining several curves together end-to-end. The order of the curve determines the minimum number of control points necessary to define the curve. You must have at least order control points to define a curve. (So for a curve of order 4, you must have at least four control points.) To make curves with more than order control points, you can join two or more curve segments into a piecewise curve.

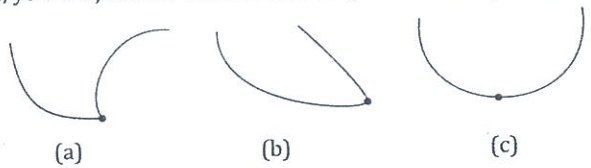


Figure 3.33 (a) Zero Order Continuity (b) First Order Continuity (slope) (c) Second Order Continuity (Curvature).

The order of the curve also affects how the curve behaves when a control point is moved.

- No continuity: The curves do not meet at all.
- Zero Order Continuity: It means the two curves meet.
- First Order Continuity: It requires the curve to be tangent at the point of intersection.
- Second Order Continuity: It requires that curvature must be the same.

The above requirements are the basis for the Bezier and B-spline formulations of curves and surfaces.

3.7.1 Bezier Curves

Bezier curves were widely publicized in 1962 by the French engineer Pierre Bezier, who used them to design automobile body design of the Renault car.

Bezier defines the curve P(u) in terms of the location of n+1 control points P_i

$$P(u)=\sum_{i=0}^{n} P_i B_{i,n}(u) \qquad ...(16)$$

where $B_{i,n}(u)$ is a blending function

$$B_{i,n}(u) = C(n, i)\, u^i (1-u)^{n-i} \qquad ...(17)$$

and C(n, i) is the binomial coefficient,

$$C(n,i)=\frac{n!}{i!(n-i)!} \qquad ...(18)$$

It could be expressed by writing equation for the x, y and z parametric function separately:

$$x(u)=\sum_{i=0}^{n} x_i B_{i,n}(u); \qquad y(u)=\sum_{i=0}^{n} y_i B_{i,n}(u); \qquad z(u)=\sum_{i=0}^{n} z_i B_{i,n}(u)$$

where the three dimensional location of the control point P_i is (x_i, y_i, z_i). Fig 3.34 shows Bezier curve in the plane, the z coordinate of each control point is zero. The curve shows four control points in fig. 3.34(a).

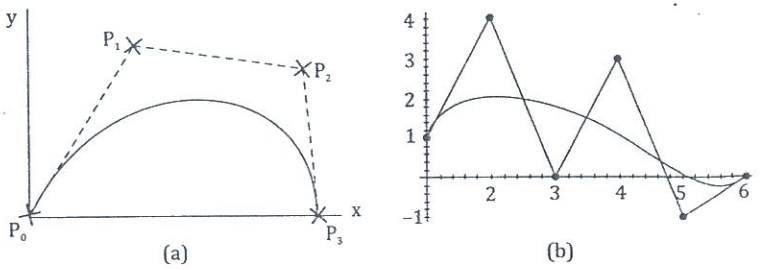


Figure 3.34 Bezier Curve (a) with 4 Control Points (b) with 6 Control Points.

A Bezier curve is a polynomial of degree, one less than the number of control points used. Three points generate a parabola, 4 points a cubic curve and so on.

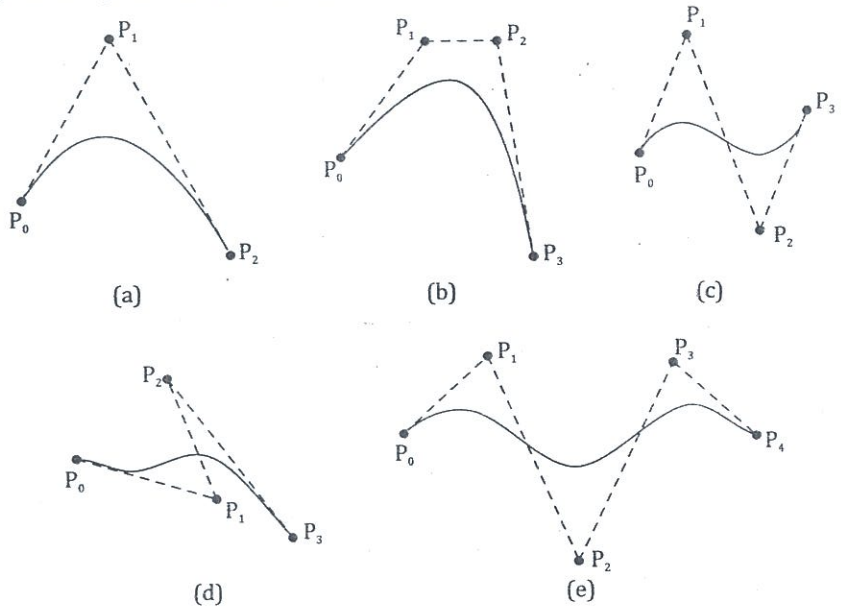


Figure 3.35 Bezier Curve generated from three, four, and five control points. Dashed Lines connect the Control Point Positions.

Properties of Bezier Curves

- A Bezier curve always passes through the first and last control point.
- It lies within the convex polygon boundary of the control points. From the properties if Bezier blending functions all are positive and their sum is always 1.

i.e $\sum_{i=0}^{n} B_{i,n}(u) = 1$ ...(19)

Cubic Bezier Curves

Cubic Bezier curves are generated with four control points. The four blending functions for cubic Bezier curves, obtained by substituting n = 3 in equation (17)

$B_{i,n}(u) = C(n, i)\, u^i (1-u)^{n-i}$

Substitute n = 3 and i = 0 to 3

$B_{0,3}(u) = C(3, 0)\, u^0 (1-u)^{3-0}$

$$\rightarrow \quad B_{0,3}(u) = \frac{3!}{0!(3-0)!} u^0 (1-u)^{3-0}$$

therefore $B_{0,3}(u) = (1-u)^3$

$B_{1,3}(u) = 3u(1-u)^2$

$B_{2,3}(u) = 3u^2(1-u)$

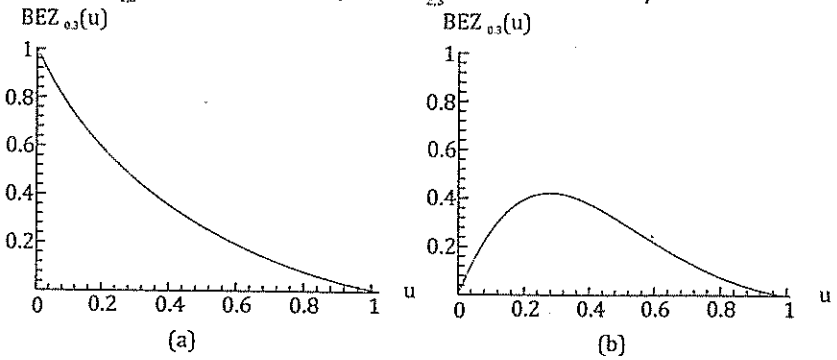
$B_{3,3}(u) = u^3$ ...(20)

Plots of the four cubic Bezier blending function are given in Fig: 3.36. The form of the blending function determines how the control points influence the shape of the curve for values of parameter u over the range from 0 to 1.

At u = 0, the function $B_{0,3}$ =1.

At u= 1, the function $B_{3,3}$ =1. Thus the cubic Bezier curves will always pass through the control points P0 and P3. $B_{1,3}$ and $B_{2,3}$ control the shape of the curve at intermediate values of u, so that the resulting curve tends towards points P_1 and P_2.

Blending function $B_{1,3}$ is maximum at u= 1/3 and $B_{2,3}$ is maximum at u = 2/3.



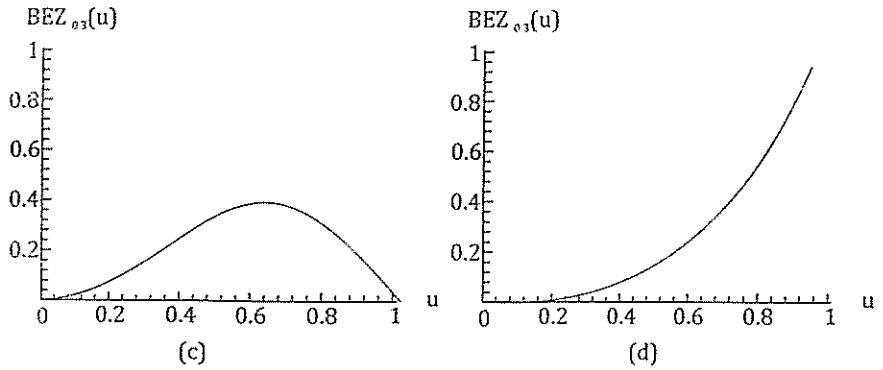


Figure 3.36 Four Blending Functions for Cubic Curves (n=3)

The Bezier curves do not allow for local control of the curve shape. If we reposition any one of the control points, the entire curve will be affected.

3.8 Hidden Surface Removal

In 3D computer graphics, hidden surface detection (also known as hidden surface removal (HSR), or visible surface detection (VSD) *is the process used to determine which surfaces and parts of surfaces are not visible from a certain viewpoint.*

Hidden line and hidden surface algorithms, share several characteristics and capitalize on various forms of coherence in order to reduce the computing required to generate an image.

- The algorithms use some form of *geometric sorting* to distinguish visible parts of an object from those that are hidden.
- These algorithms used on various forms of coherence in order to reduce the computing required for generating an image.

Coherence

- *Scan-line coherence* arises because the display of a scan line in a raster image is usually similar to the display of the preceding scan line.
- *Frame coherence* in a sequence of images designed to show motion, recognizes that successive frames are similar.
- *Object coherence* results from relationships between different objects or between separate parts of the same object.
- *Edge Coherence,* An edge may change visibility only where it crosses behind a visible edge or penetrates a visible face.
- *Face coherence,* Surface properties typically vary smoothly across a face allowing computations for one part of a face to be modified incrementally to apply to adjacent parts.
- *Area Coherence.* A group of adjacent pixels is often covered by the same visible face.

- *Depth Coherence.* Adjacent parts of the same surface are typically close in depth where as different surfaces at the same screen location are typically separated farther in depth.

Visible-surface detection algorithms are classified according to whether they deal with object definitions or with their projected images. These two approaches are called ***object-space methods*** and ***image-space methods***.

Object-Space Algorithm

- ★ It compares objects and parts of objects to each other within the scene definition to determine which surfaces visible.
- ★ It performs geometric calculation with as much precision as possible. Since the precision of the solution is greater than that of a display device, the image can be displayed enlarged many times without losing accuracy.
- ★ The computation time will tend to grow with the number of objects in the scene, whether visible or not.
- ★ The cost of object-space algorithm grows slowly as the complexity of the scene increases.
- ★ Object space methods are used in hidden line algorithm.
- ★ Object space method is used in *Back Face Detection.*

Image-Space Algorithm

- ❖ In an *image-space algorithm*, visibility is decided point by point at each pixel position on the projection plane.
- ❖ It performs calculations with only enough precision to match the resolution of the display screen used to present the image.
- ❖ The computation time will tend to grow with the complexity of the visible parts of the image.
- ❖ The cost of image-space algorithm grows slowly than that of object-space algorithms as the complexity of the scene increases.
- ❖ Image space methods are used in hidden surface algorithm.
- ❖ Image space method is used in *Depth Buffer method.*

3.8.1 Back Face Detection / Removal

Object-space method is used for identifying the back faces of a polyhedron, based on the "inside-outside" tests.

A point (x, y, z) is "inside" a polygon surface with plane parameters A, B, C, and D if

$$Az + By + Cz + D < 0 \qquad ...(21)$$

When an inside point is along the line of sight to the surface, the polygon must be a back face.

Test by considering the normal vector N to a polygon surface, which has Cartesian components (A, B, C).

If V is a vector in the viewing direction from the eye (or "camera") position then the polygon is a back face if

$$V \cdot N > 0 \qquad ...(22)$$

In a right-handed viewing system with viewing direction along the negative z_v axis (Fig. 3.37, the polygon is a back face if $C < 0$).

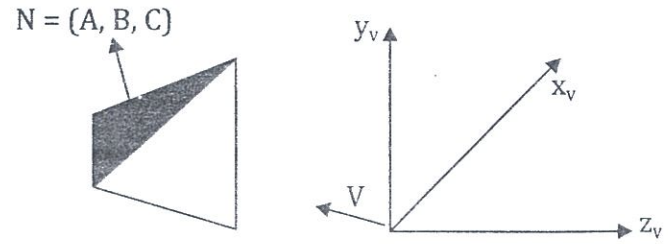


Figure 3.37 Polygon surface with plane parameter C< 0.

In a left-handed viewing system plane parameters A, B, C and D can be calculated from polygon vertex coordinates specified in a clockwise direction. Back faces have normal vectors that point away from the viewing position and are identified by C >= 0 when the viewing direction is along the positive z_v axis.

By examining parameter C for the different planes defining an object, we can immediately identify all the back faces.

For other objects, such as the concave polyhedron more tests need to be carried out to determine whether there are additional faces that are totally or partly obscured by other faces. In general, back-face removal can be expected to eliminate about half of the polygon surfaces in a scene from further visibility tests.

3.8.2 Depth Buffer Method

Image space method is used to detect visible surface in depth buffer method, which compares surface depths at each pixel position on the projection plane.

This procedure is also referred to as the *z-buffer method*, since object depth is usually measured from the view plane along the z axis of a viewing system.

Each surface of a scene is processed separately, one point at a time across the surface.

Fig: 3.38 shows three surfaces at varying distances along the orthographic projection line from position (x, y) in a view plane taken as the $x_v y_v$ plane. Surface S_1, is closest at this position, so its surface intensity value at (x, y) is saved.

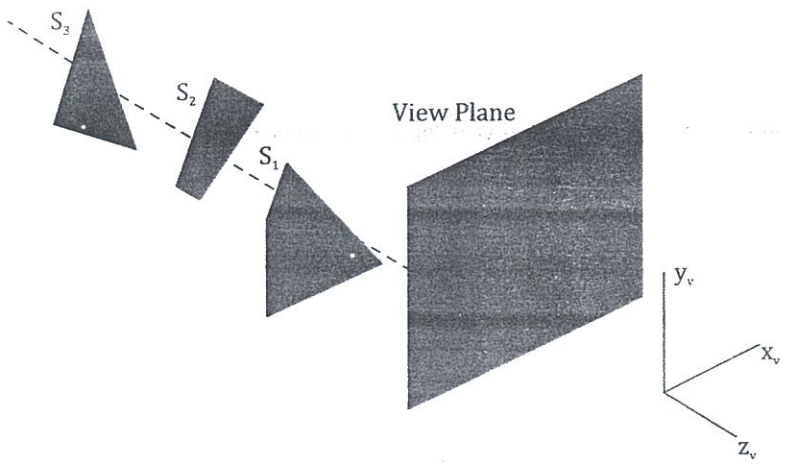


Figure 3.38 At view Plane Position (x, y) surface S_1 has the Smallest Depth from the View Plane.

The depth-buffer algorithm in normalized coordinates, so that z values range from 0 at the back clipping plane to z_{max} at the front clipping plane. The value of z, can be set either to 1 or to the largest value that can be stored on the system.

Two buffer areas are required, One a depth buffer is used to store depth values for each (x, y) position as surfaces are processed, and the second, refresh buffer stores the intensity values for each position.

Depth Buffer Algorithm:

Step 1: Initialize the depth buffer and refresh buffer for all buffer positions (x,y) depth(x, y) = 0, refresh(x, y) = $I_{backgnd}$

Step 2: For each position on each polygon surface, compare depth values to previously stored values in the depth buffer to determine visibility.

(a) Calculate the depth z for each (x, y) position on the polygon.

(b) If z > depth(x, y), then set

depth(x, y) = z, refresh(x, y) = I_{surf}(x,y)

where $I_{backgnd}$ is the value for the background intensity, and I_{surf}(x,y) is the projected intensity value for the surface at pixel position (x,y). After all surfaces have been processed, the depth buffer contains depth values for the visible surfaces and the refresh buffer contains the corresponding intensity values for those surfaces.

(c) If z < depth(x, y), this polygon is closer to the observer than others already recorded for this pixel.

Depth values for a surface position *(x,* y) are calculated from the plane equation for each surface:

$$Z = \frac{-Ax - By - D}{C} \quad \ldots(23)$$

From position (x, y) on a scan line, the next position across the line has coordinates *(x* + 1, y), and the position immediately below on the next line has coordinates *(x,* y - 1).

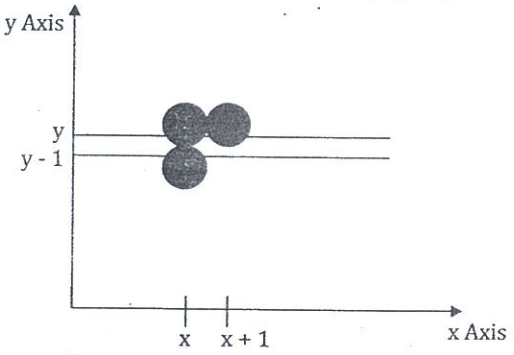


Figure 3.39 Point(x, y) on a Scan line.

The depth of position (x, y) has been determined to be z, then the depth z' of the next position (x + 1, y) along the scan line is obtained as

$$z' = \frac{-A(x+1) - By - D}{C} \quad \ldots(24)$$

$$z' = z - \frac{A}{C} \quad \ldots(25)$$

The ratio – A/C is constant for each surface.

On each scan line, we start calculating the depth on a left edge of the polygon that intersects that scan line (Fig. 3.40).

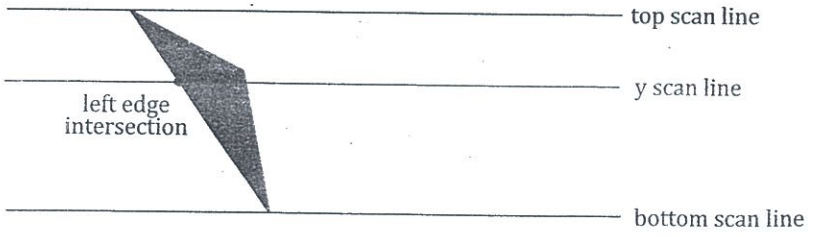


Figure 3.40 Scan Line Intersecting a Polygon Surface.

Depth values at each successive position across the scan line are calculated by Eq. (25).The y-coordinate extents of each polygon are determined, starting at a top vertex, we can recursively calculate x positions down a left edge of the polygon as x' = x - 1/m, where m is the slope of the edge.

$$z' = z + \frac{A/m + B}{C} \quad \text{...(26)}$$

$$z' = z + (B / C) \quad \text{...(27)}$$

pros:

- Easy implementation (directly in hardware) .
- No sorting of surfaces.
- O (# of objects ×pixels).
- Good rendering algorithm with polygon models.

Cons:

- Additional buffer (z-buffer)
- Aliasing (point-sampling).
- Difficult to deal with transparent object.

3.9 Scan-Line Method

Image space method used for removing hidden surfaces.

- An extension of scan-line polygon filling (with multiple surfaces).
- Idea is to intersect each polygon with a particular scan line. Solve hidden surface problem for just that scan line.
- Across each scan line, depth calculations are made for each overlapping surface to determine which is nearest to the view plane.
- When the visible surface has been determined, the intensity value for that position is entered into the refresh buffer.
- The cost of tiling scene is roughly proportional to its depth complexity.
- Efficient way to tile shallowly-occluded scenes.
- May need to split.

Tables are set up for the various surfaces that is an *edge table* and a *polygon table.*

The *edge table* contains coordinate endpoints for each line in the scene, the inverse slope of each line, and pointers into the polygon table to identify the surfaces bounded by each line.

The *polygon table* contains coefficients of the plane equation for each surface, intensity information for the surfaces, and possibly pointers into the edge table.

We define a *flag* for each surface that is set on or off to indicate whether a position along a scan line is inside or outside of the surface.

- Scan lines are processed from left to right. At the leftmost boundary of a surface, the surface flag is turned on; and at the rightmost boundary, it is turned off.
- Fig: 3.41 illustrate the scan-line method for locating visible portions of surfaces for pixel positions along the line.

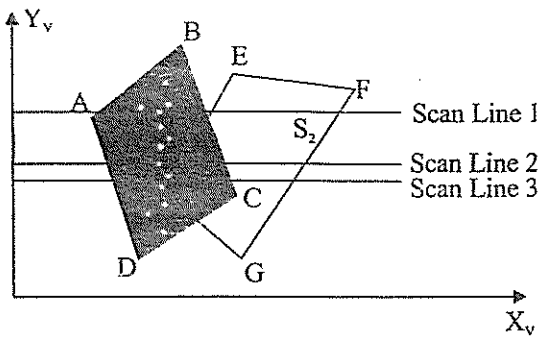


Figure 3.41 Scan Line Crossing the Projection of Two Surfaces, S_1 and S_2 in the View Plane.

Dashed lines indicate the boundaries of hidden surfaces.

* The list for scan line 1 contains information from the edge table for edges AB, BC, EH, and FG. For positions along scan line 1 between edges AB and BC, only the flag for surface S_1 is on.

  Depth calculations are not necessary, and intensity information for surface S_1 is entered from the polygon table into the refresh buffer.
* For edges EH and FG, only the flag for surface S_2 is on. The intensity values in the other areas are set to the background intensity, since there are no other positions of intersection along scan line 1.
* For scan lines 2 and 3 the active edge contains edges AD, EH, BC, and FG. Along scan line 2 from edge AD to edge EH, only the flag for surface S_1 is on. But edges between EH and BC, the flags for both surfaces S_1 and S_2 are on.
* Depth calculations are calculated using the plane coefficients for the two surfaces. For this example, the depth of surface S_1 is assumed to be less than that of S_2 so intensities for surface S_1 are loaded into the refresh buffer until boundary BC is encountered.

  The flag for surface S_1 goes off, and intensities for surface S_2 are stored until edge FG is passed.
* For Scan line 3 has the same list of edges as scan line 2. Since there are no changes in line intersections, depth calculations between edges *EH* and BC is not necessary.

3.10 Review Questions

Short Answer Questions

1. What is Polygon mesh?
2. What is Bezier Basis Function?
3. What is the use of control points?
4. What are the important properties of Bezier Curve?
5. Define Octrees?
6. Define Projection?

7. What are the steps involved in 3D transformation?
8. What do you mean by view plane?
9. What is view distance?
10. What you mean by parallel projection?
11. What do you mean by Perspective projection?
12. What is Projection reference point?
13. What are the different types of parallel projections?
14. What is the use of hidden line removing algorithm?
15. How realistic pictures are created in computer graphics?
16. What is Z-buffer algorithm for removing hidden faces?
17. What are the merits and demerits of various techniques for hidden surface removal?
18. What is coherence? Discuss various types of coherence that can be used to make visible surface algorithm more efficient.
19. Explain 3D and stereoscopic views.
20. Difference between a right handed system and left handed system.
21. What is intensity cueing? Explain
22. Polygon data tables can be organized into two groups, What are they explain?
23. Explain the properties of curve.

Long Answer Questions

1. Explain the three dimensional display methods?
2. Explain Bezier curves and surfaces.
3. Explain general three dimensional rotations.
4. Explain Back face detection method and Depth buffer method.
5. Explain the Z-Buffer algorithm for hidden surface removal.
6. Explain the advantages and disadvantages of Z-Buffer algorithm.
7. Explain the scan line algorithm for hidden surface removal.
8. Write the difference between image space and object space methods of hidden surface and hidden line algorithms.
9. What are Octrees? How are they used to represent 3D objects?
10. Explain about polygon surfaces in detail.
11. Explain 3D Transformation?
12. Explain three dimensional scaling.
13. Explain three dimensional translations.
14. Explain how cubic Bezier curves are generated.

❑❑❑❑

UNIT 4

GRAPHICAL INPUT DEVICES AND TECHNIQUES

- Input Device
- Pointing Device
  - Keyboard
  - Trackball and Spaceball
  - Touchpad
  - Touch Screen
  - Mouse
  - Joystick
  - Graphics Tablet
  - Light Pen
- Input of Graphical Data
- Interactive Picture Construction Techniques
  - Positioning Techniques
  - Grids
  - Rubber Band Method
  - Painting and Drawing
  - Constraints
  - Gravity Field
  - Dynamic Manipulation
- Pointing and Selection
  - Selection
  - Multiple Selections
  - Selection Feedback
- Menu Selection
- Selection by Naming
- Tablet
- Data Glove
- Digitizers
- Voice Systems
- Review Questions

4.1 Input Device

Before a computer can process your data, you need some method to input the data into the machine. The device you use will depend on what form this data takes (be it text, sound, artwork, etc.).

An input device is any peripheral (piece of computer hardware equipment) used to provide data and control signals to an information processing system (such as a computer). All the computers have keyboard which is an input device. The other input devices are Mouse, Trackball, Joystick, Touchpad, and Light Pen.

Two types of graphical interaction:

- Pointing Device – point's items already on the screen.
- Positioning Device – positions new items.

4.2 Pointing Device

A pointing device is an input interface that allows a user to input continuous and multi-dimensional data to a computer. Movements of the pointing device are echoed on the screen by movements of the pointer (or cursor) and other visual changes. The most common pointing device is the mouse.

Pointing Devices:

- Based on motion of an object
  - Mouse
  - Trackball
  - Joystick
- Based on touching a surface
  - Touchpad
  - Graphics Tablet
  - Touch Screen
  - Light Pen

4.2.1 Keyboard

The alphanumeric keyboard is used to enter text information into the computer, as when you type the contents of a report.

A 'keyboard' is a human interface device which is represented as a layout of buttons. Each button, or key, can be used to either input a linguistic character to a computer, or to call upon a particular function of the computer. Traditional keyboards use spring-based buttons, though newer variations employ virtual keys, or even projected keyboards.

A keyboard typically has characters engraved or printed on the keys and each press of a key typically corresponds to a single written symbol. However, to produce some symbols requires pressing and holding several keys simultaneously or in sequence. While most keyboard keys produce letters, numbers or signs (characters), other keys or simultaneous key presses can produce actions or computer commands.

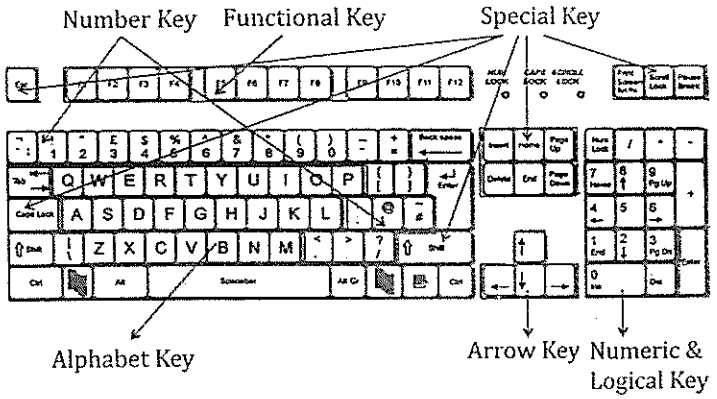


Figure 4.1 Keyboard

The keys on computer keyboards are often classified as follows:

- Alphanumeric keys.
- Punctuation keys.
- Special keys.

Alphanumeric keys are used for typing alphabets and numbers. Other keys such as Tab, Caps lock, scroll lock, num lock, backspace and enter key are used for specific task.

Punctuation keys are used to enter comma, period, semicolon and punctuations.

Special keys are function keys, control keys, and arrow keys, caps lock key.

Function keys are 12 keys and Esc key each has its own function. For example F1 is for help, F2 to rename a selected item or object. F3 to find all files, F4 Selects the go to a different folder, Esc is for exit.

Control Keys (CTRL) - The CTRL key is used in conjunction with another key. Holding it down while pressing another key will initiate a certain action. CTRL key combinations are defined by the application being used. For example CTRL+S will save the current file or document, and CTRL+P will print the current file or document.

Cursor movement keys – they are used to move the cursor on the display up, down, left, right. They allow the user to position the cursor at the point of display where data is to be entered or changed. Other keys are Page Up, Page Down, Home, End, Insert, Pause, Num Lock, Scroll Lock, Break, Print Screen.

Caps Lock - The Caps Lock key is a toggle key. Pressing it once turns it on. Pressing it again turns it off. Some computer keyboards have a light or indicator that shows when the Caps Lock is on and when it is off. When Caps Lock is on, every letter that is typed will be a capital letter. Unlike a typewriter, the Caps Lock key on a computer keyboard affects only letters. It has no effect on the number or symbol keys.

4.2.2 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. The mouse allows an individual to control a *pointer* in a graphical user interface (GUI). Utilizing a mouse a user has the ability to perform various functions such as opening a program or file and does not require the user to memorize commands like those used in a text-based command line environment such as MS-DOS. The Mouse was originally referred to as an X-Y Position Indicator for a Display System.

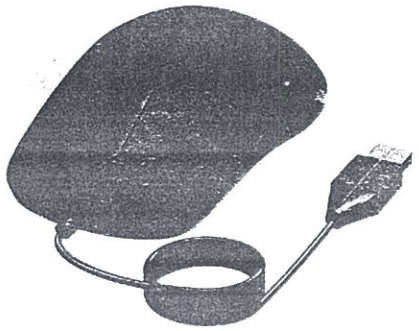


Figure 4.2 Mouse

The mechanical mouse uses a rubber coated ball on the underside. The movement of this ball sends electrical signals to the system unit which causes the cursor, or pointer to move correspondingly. An optical mouse uses diodes to emit light onto a metal pad, performing the same task but usually with greater accuracy.

The different actions performed by a mouse are:

Pointing :	Mouse pointer points to a particular object on the screen.
Single Click :	Pressing of a mouse button once and release it immediately selects an object.
Double Click :	Clicking the mouse button twice in rapid succession carriers out an operation such as opening a folder.
Right Click :	Right click displays a pop up menu corresponding to the selected object.
Drag and Drop :	Clicking on a certain object, holding down the mouse button and dragging the mouse to another location and then releasing the button.

Advantages of Mouse

- Easy to use.
- No need to type commands.
- Low cost.
- Easy to select menus and open them.

4.2.3 Trackball and Spaceball

A trackball is a ball that can be rotated with the fingers or palm of the hand, to produce screen-cursor movement. Potentiometers, attached to the ball, measure the amount and direction of rotation. Trackballs are often mounted on keyboards or other devices such as the Z mouse.

While a trackball is a two-dimensional positioning device, a space ball provides six degrees of freedom. Unlike the trackball, a space ball does not actually move. Strain gauges measure the amount

of pressure applied to the space ball to provide input for spatial positioning and orientation as the ball is pushed or pulled in various directions. Spaceballs are used for three-dimensional positioning and selection operations in virtual reality systems.

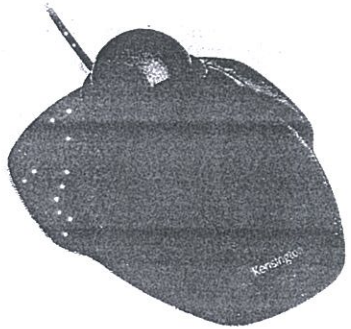


Figure 4.3 Trackball

4.2.4 Joystick

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 switches to signal certain actions once a screen position has been selected.

Figure 4.4 Joystick

4.2.5 Touchpad

Touchpad's are small touch-sensitive tablets commonly used on laptop computers. Touchpads typically respond in relative mode, because of the small size of the pad. Most touchpads support an absolute mode to allow Asian language input or signature acquisition, for example, but for these uses make sure the touchpad can sense contact from a stylus (and not just the bare finger).

Figure 4.5 Touchpad of a Laptop

4.2.6 Graphics Tablet

A graphics tablet consists of an electronic writing area and a special "pen" that works with it. Graphics tablet allows artists to create graphical images with motions and actions similar to using more traditional drawing tools. The pen of the graphics tablet is pressure sensitive, so pressing harder or softer can result in brush strokes of different width (in an appropriate graphics program).

The tablet provides a flat drawing area resting on a table top. The surface of the tablet contains 1024 lines parallel to the x-axis and 1024 lines parallel to the y-axis.

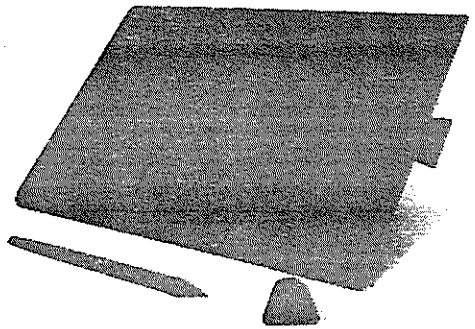


Figure 4.6 Graphics Tablet

Each individual line carries a digitally coded signal that can be picked up by the stylus. The stylus contains a sensitive amplifier which detects the pulses from the lines amplifies them and delivers them via coaxial cable to decoding logic, which in turn stores binary integer co-ordinates in the tablets buffer registers.

4.2.7 Touch Screen

Some computers, especially small hand-held PDAs, have touch sensitive display screens. The user can make choices and press button images on the screen. You often use a stylus, which you hold like a pen, to "write" on the surface of a small touch screen

Touch screens are used in applications such as:

- Mobile phones.
- ATM (Automated Teller Machine) of a bank where the withdrawal of money or checking of balance etc.,
- Information in Railway Station or airport.

Advantages of Touch Screen

- Requires minimum skill and easy to use.
- Occupies less space.
- Does not require maintenance.
- Used in ATM's

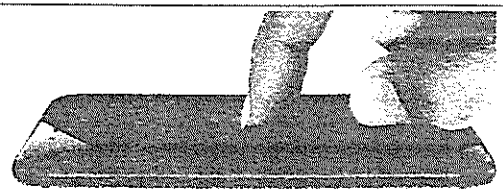


Figure 4.7 Touch Screen

4.2.8 Light Pen

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

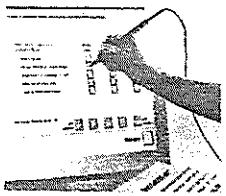


Figure 4.8 Light Pen

Advantages of Light Pen

- Eliminates usage of a mouse or touch pad.
- Can be used to take notes.
- Signature can be input using the stylus which can be stored as a image and used anywhere.

4.3 Input of Graphical Data

We can input the data using many devices. Picture specification needs values for coordinate positions, values for character string parameter, scalar values for the transformation parameters, values specifying menu options, and values for identification of pictures parts. Depending on the type of data to be input, a logical classification of input devices can be made.

Logical Classification of Input Devices

The different kinds of input devices are

Locator – a device for specifying a position (x, y).

Stroke – a device for specifying a series of coordinate positions.

String – a device for specifying text input.

Valuator – a device for specifying scalar values.

Choice – a device for selecting menu option.

Pick – a device for selecting picture components.

Locator

- Locator is a device for interactive selection of a coordinate point is by positioning the screen cursor. We can do this with a mouse, joystick, trackball, space ball, thumb wheels, dials, a digitizer stylus or hand cursor.
- When the screen cursor is at the desired location, a button is activated to store the coordinates of that screen point.
- Keyboards can be used for locator input in several ways. Keyboard usually has four cursor-control keys that move the screen cursor up, down, left, and right.
- Light pens have also been used to input coordinate positions. Light pens operate by detecting light emitted from the screen phosphors, some nonzero intensity level must be present at the coordinate position to be selected.

Stroke

- Stroke is a device used for specifying a series of coordinate positions.
- Stroke-device input is equivalent to multiple calls to a locator device. The set of input points is often used to display line sections.
- Physical devices used for generating locator input can be used as stroke devices.
- Continuous movement of a mouse, trackball, joystick, or tablet hand cursor is translated into a series of input coordinate values.
- The graphics tablet is one of the more common stroke devices.

String

- It is used for input of text values.
- Physical device used for string input is the keyboard.
- Input character strings are typically used for picture or graph labels.
- Physical devices can be used for generating character patterns in a "text-writing" mode.
- For this input, individual characters are drawn on the screen with a stroke or locator-type device. A pattern-recognition program then interprets the characters using a stored dictionary of predefined patterns.

Valuator

- Valuator device is used for specifying scalar values.
- Valuators are used for setting various graphics parameters, such as rotation angle and scale factors.
- A physical device used to provide valuator input is a set of control dials.
- Floating-point numbers within any predefined range are input by rotating the dials. Dial rotations in one direction increase the numeric input value, and opposite rotations decrease the numeric value.
- Any keyboard with a set of numeric keys can be used as a valuator device. A user simply types the numbers directly in floating-point format.
- Joystick, trackball, tablets, can be adapted for valuator input by interpreting pressure or movement of the device relative to a scalar range.

Choice

- Choice is a device for selecting menu option.
- A choice device is defined as one that enters a selection from a list (menu) of alternatives.
- Choice devices are a set of buttons; a cursor positioning device, such as a mouse, trackball, or keyboard cursor keys; and a touch panel.
- For screen selection of listed menu options, we can use cursor-control devices. When a coordinate position (x, y) is selected. A menu item with vertical and horizontal boundaries at the coordinate values x_{min}, x_{max}, y_{min}, y_{max} is selected if the input coordinates (x, y) satisfy the inequalities.

$$x_{min} <= x <= x_{max}$$
$$y_{min} <= y <= y_{max}$$

- A cursor-control device, such as a mouse, a selected screen position is compared to the area occupied by each menu choice.
- A keyboard can also be used to type in commands or menu options.

Pick

- Pick devices are used to select parts of a scene that are to be transformed or edited in some way.
- Devices used for object selection are the same as those for menu selection.
- With a mouse or joystick, we can position the cursor over the primitives.
- The position of the cursor is then recorded, and several levels of search may be necessary to locate the particular object that is to be selected.
- First, the cursor position is compared to the coordinate extents of the various structures in the scene. If the bounding rectangle of a structure contains the cursor coordinates, the picked structure has been identified.

- A cursor positioning is to use button input to highlight successive structures. A second button is used to stop the process when the desired structure is highlighted.
- We could also use a keyboard to type in structure names. This is a straightforward, but less interactive, pick-selection method.
- Descriptive names can be used to help the user in the pick process.

4.4 Interactive Picture Construction Techniques

There are several techniques that are incorporated into graphics packages to aid the interactive construction of pictures. We have seen that to enter the information about the objects, it is necessary to enter the coordinates of the object. The coordinate gives the information about the position or boundaries of the object. The coordinate information of the object can be entered using input devices such as locator and stroke. Once the information is entered it can be modified to rearrange or reshape the object. We see several techniques available in graphics packages to construct the interactive picture using input devices.

4.4.1 Positioning Techniques

We interactively select coordinate positions with a pointing device, usually by positioning the screen cursor. Positioning is some times known as locating. The user indicates a position on the screen with an input device, and this position is used to insert a symbol or to define the endpoint of a line.

Positioning involves the user in first moving the cursor or tracking cross to the desired spot on the screen and then notifying the computer by pressing a button or key. A single positioning operation can be used to insert a symbol as in Fig 4.9 and two in succession can be defined the endpoints of a line.

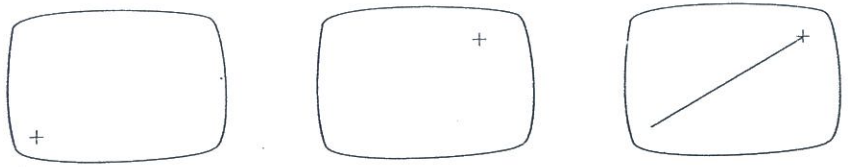


Figure 4.9 (a) Press Button for First Line Endpoint (b) Press Button at Second Line Endpoint (c) Line Displayed between the Two Chosen Endpoint

4.4.2 Constraints

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 or vertical alignment of straight lines. With this constraint, we can create horizontal and vertical lines without worrying about precise specification of endpoint coordinates.

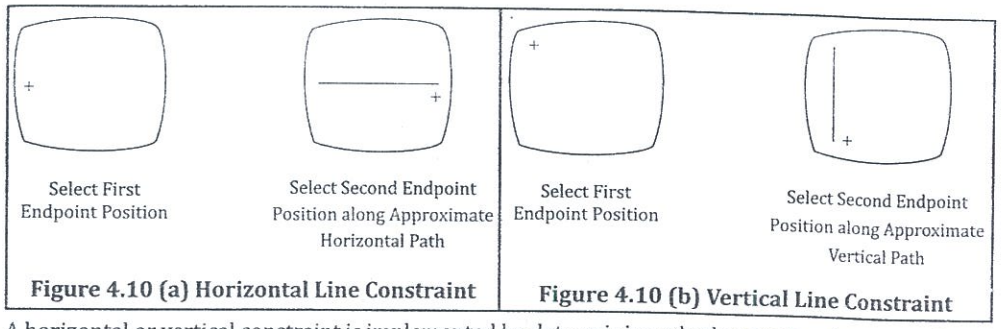


Figure 4.10 (a) Horizontal Line Constraint **Figure 4.10 (b) Vertical Line Constraint**

A horizontal or vertical constraint is implemented by determining whether any two input coordinate endpoints are more nearly horizontal or more nearly vertical. If the difference in the y values of the two endpoints is smaller than the difference in x values, a horizontal line is displayed. Otherwise, a vertical line is drawn.

4.4.3 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. 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 4.11 Line Drawing using a Grid.

4.4.4 Gravity Field

When constructing a picture, we sometimes need to connect lines at position 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.

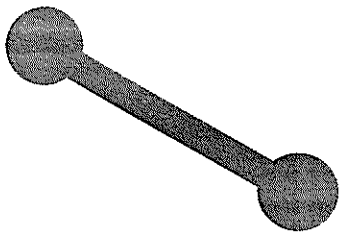


Figure 4.12 Gravity Field Around a Line.

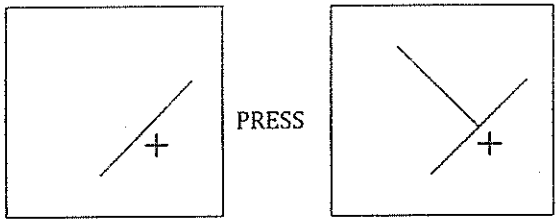


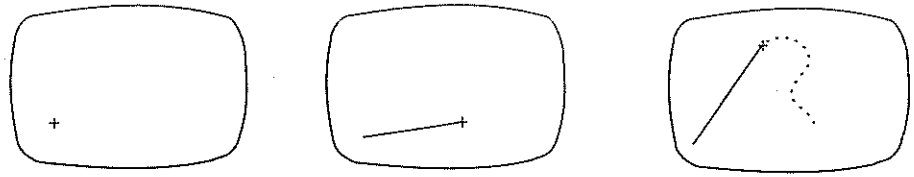
Figure 4.13 Using a Gravity Field to attach a Line Endpoint to another Line.

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.

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.

4.4.5 Rubber Band Method

Straight lines can be constructed and positioned using ***rubber-band*** methods, which stretch out a line from a starting position as the screen cursor is moved.



(a) Select first line endpoint

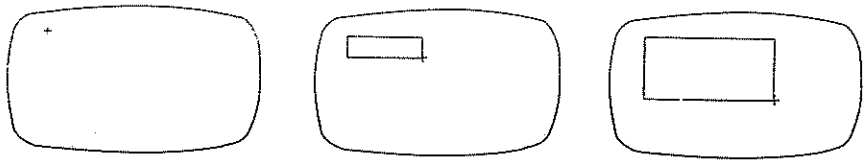
(b) As the cursor moves a line stretches out from the initial point

(c) Line follows cursor position unit the second endpoint is selected

Figure 4.14 Rubber Band Method for Drawing and Positioning a Straight Line Segment

We first select a screen position for one endpoint of the line. Then, as the cursor moves around, the line is displayed from the start position to the current position of the cursor. When we finally select a second screen position, the other line endpoint is set.

Rubber-band methods are used to construct and position other objects besides straight lines. Figure 4.15 demonstrates rubber-band construction of a rectangle, and Fig. 4.16 shows a rubber-band circle construction.

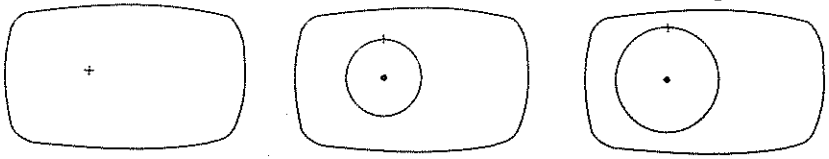


(a) Select position for one corner of the rectangle

(b) Rectangle stretches out as cursor moves

(c) Select final position for opposite corner of the rectangle

Figure 4.15 Rubber Band Method for Construction a Rectangle.



(a) Select position for the circle center

(b) Circle stretches out as the cursor moves

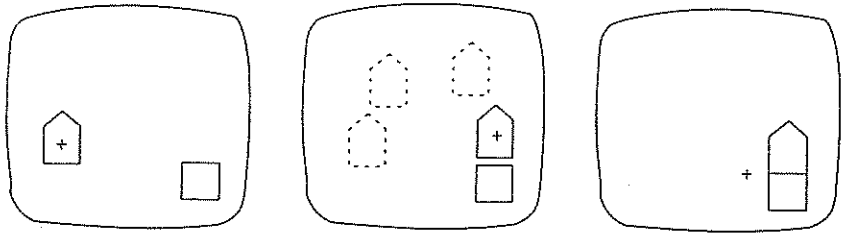
(c) Select the final radius of the circle

Figure 4.16 Rubber Band Method for Construction a Circle.

4.4.6 Dynamic Manipulation

Dragging

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



(a) Press button to begin displaying selected object at cursor position

(b) Object displayed at new positions, following cursor movement

(c) Press stop button to end dragging operation when object correctly positioned

Figure 4.17 Dragging a Picture to another Position

.4.7 Painting and Drawing

ptions for sketching, drawing, and painting come in a variety of forms. Curve drawing options can be rovided using standard curve shapes, such as circular arcs and splines, or with freehand sketching rocedures. Splines are interactively constructed by specifying a set of discrete screen points that ive the general shape of the curve.

freehand drawing, curves are generated by following the path of a stylus on a graphics tablet or ne path of the screen cursor on a video monitor. Once a curve is displayed, the designer can alter the urve shape by adjusting the positions of selected points along the curve path.

4.5 Pointing and Selection

nput devices allow the user to point information on the screen. In case where positioning does ccur, the user generally make extensive use of pointing techniques to select picture parts that need nodification and to issue commands with the aid of command menus.

.5.1 Selection

election is choosing objects from a set of alternatives. Programs that provide the user with the bility to interact with a image, which must provide a means of ***selecting*** parts of the image. The user an then move, delete or copy the selected part. Selection is a useful tool in the hands of the user for electing the image.

The selection techniques have problems.

- First problem is the fact that input devices provide only a pair of coordinates which is not sufficient for selection.
- A second problem in selection is to determine how much is being selected.

When the user points at the line as in figure 4.18, he may wish to select a point on the line, the ine itself, all four lines of the box to which the line belongs, or complete box. In order to solve this mbiguity, the user must define a certain grain of selection.

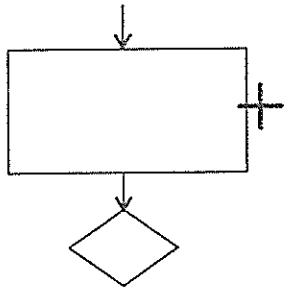


Figure 4.18 Ambiguous Selections

Several techniques for achieving unambiguous selections:

- Use of selection points.
- Defining a bounding rectangle.
- Multiple key for selection.
- Prefix commands.
- Modes.

Use of Selection Points

In order to select a graphical unit the user points to a specific spot, such as the center of a circle or an endpoint of a line. Selection points can be provided for symbols and larger sub pictures. By choosing selection points with care reduces overlap. Selection points can be emphasized by highlighting or increasing brightness.

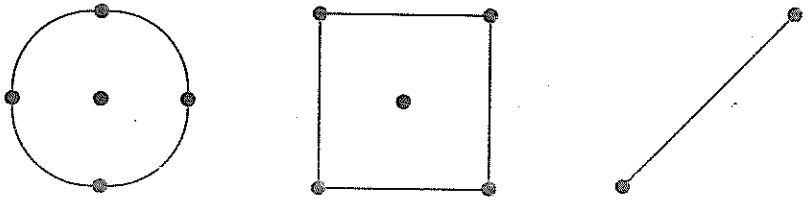


Figure 4.19 Selection Points shown by Highlighting.

Defining a Bounding Rectangle

The user can define two opposite corners of a rectangle and in this way select an object that lies within the rectangle. This technique is useful for multiple selection.

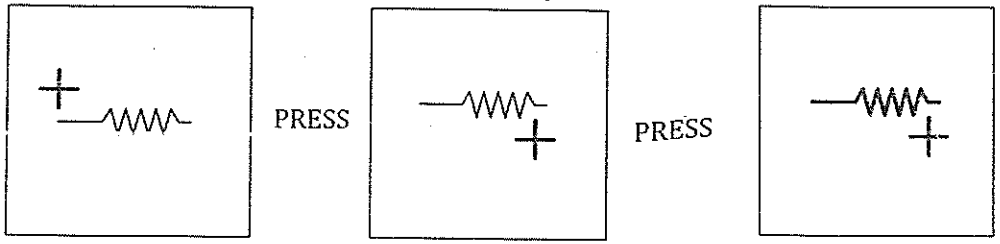


Figure 4.20 Selecting a Symbol by Defining a Bounding Rectangle.

Multiple Keys for Selection

When the user has positioned the cursor over the item to select, he can press one of several keys according to the type of item. The selection can be one key to select a line, another to select a point, and a third to select a symbol.

Prefix Commands

The type of item to be selected can be determined by the user's prior choice of command. The command is given before the selection and many specifies which type of item is to be selected. Three different DELETE command are available, DELETE POINT, DELETE LINE, and DELETE SYMBOL.

Modes

The user may be able to change the selection mechanism by setting different modes of operation. In one mode the program allow only line selection and in another the symbol selection.

4.5.2 Selection Feedback

Selection feedback is useful before the selection has been completed, the user can move the cursor around in the region of interest and see each possible selection highlighted in turn. When the item of his choice is highlighted, he completes the selection operation.

4.5.3 Multiple Selections

The user must be able to select more than one item at a time. Multiple selection capability can be provided in a number of ways.

- To indicate the first and last item in the sequence to be selected.
- To select the item by specifying a bounding rectangle.
- If neither technique is applicable, the user must select each one in turn. Further the program, must allow him to edit a set of selection if required.

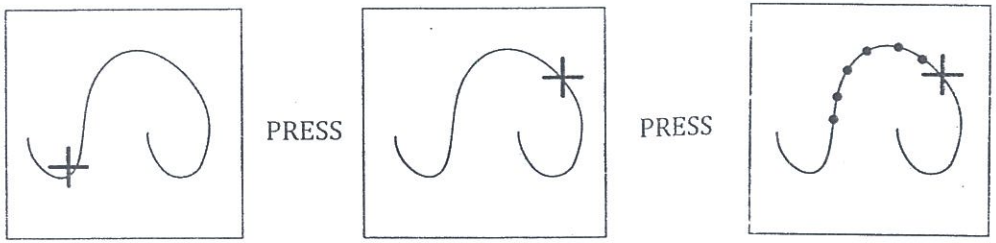


Figure 7.21 Selecting Points on a Curve by indicating the First and Last of the Sequence

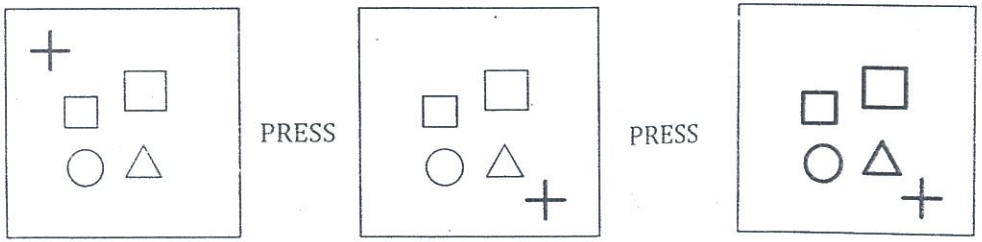


Figure 4.22 Multiple Selection by drawing a Bounding Rectangle.

4.6 Menu Selection

Menu selection is a very powerful technique for interactive input. The menu is displayed on the screen and the user points to the selection with a graphical input device. *Menus can be used for a variety of reasons:*

- They allow the user to choose an item for insertion. Fig 4.23 (a)
- To change the mode of operation of the program. Fig 4.23 (b)
- To issue a command. Fig 4.23(c)
- They protect the user from making an invalid selection, since only valid choices are included in the menu.

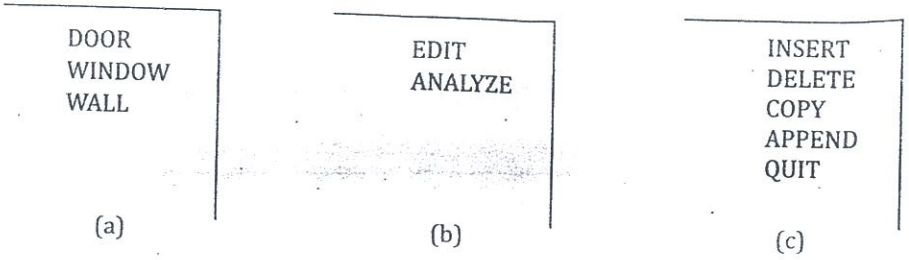
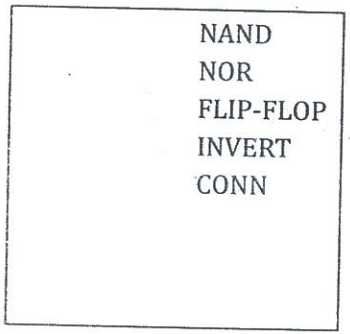
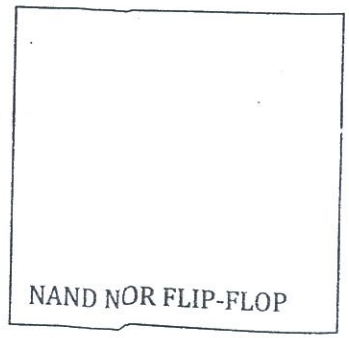


Figure 4.23 Menus (a) For Item Selection, (b) for Mode Change (c) for Command Selection

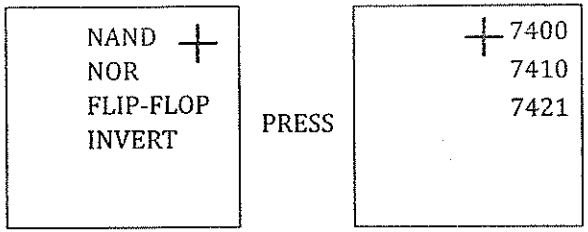
Menus are often constructed from short text strings displayed either pulled down on one side of the screen or along the bottom fig 4.24 (a) and fig 4.24 (b). It is always better to keep the menus short, dividing them into sets of submenus containing two or more selections. Fig 4.24 (c)



(a) Menu items arranged along a Vertical Screen Edge



(b) Menu item arranged along the Bottom of the Screen.



(c) Selection by Menus of a Two Level Menu

Figure 4.24 Menu Selections

Graphical symbols may be used in place of text for menu items. Command menus can be displayed as a set of graphical icons, each representing a command. Icons have the advantage that they use less screen space and therefore lead to more compact menus. Fig 4.25.

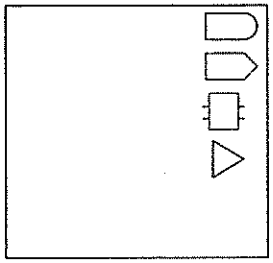


Figure 4.25 Item Menu Displayed Graphically

4.7 Selection by Naming

In this technique, the user must type the name of the choice. If the user knows the names of various objects, then referring to them by name would be reasonable and faster than pointing method. Further if the display is so cluttered that pointing or zooming is not possible, then naming would be the only choice. Typing allows users to make multiple selections through wild card characters. Selection by naming is also the best method for experienced and regular users. When naming is necessary a useful feedback after each keystroke would be to display the list of names in selection list that match the sequence of characters typed so far. As soon as an unambiguous match is typed the correct name must be automatically highlighted on the list. This technique is called auto completion.

When the name typed does not match one in the system, other name close to the typed name must be presented to the user as alternatives. Using voice recognition system, the user can speak a name, an abbreviation or code. Commands can be entered by voice, the data by keyboard or other means. This eliminates the need for special characters or modes to distinguish between data and commands.

4.8 Tablet

A drawing tablet or a pen tablet, a graphics tablet is a natural input device that converts information from a handheld stylus. The user uses the stylus like a pen, pencil, or paintbrush, pressing its tip on the tablet surface. The device can also be used in replacement of a computer mouse.

Similar to an artist drawing with a pencil and paper, a user draws on the graphics tablet with a stylus as shown in Figure 4.26. The computer will convert the drawing strokes into digital form, displaying them on the computer screen. The graphics tablet can also be used to capture user's signatures. This use is similar to the signature pads found at many retail stores, where you would write your signature after using a credit card to make a purchase.

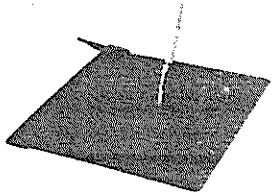


Figure 4.26 Tablet

4.9 Data Glove

A data glove is an input device that is essentially a glove worn on the hand that contains various electronic sensors that monitor the hand's movements and transform them into a form of input for applications such as virtual reality and robotics.

A data glove shown in Figure 4.27 that can be used to grasp a "virtual" object. The glove is constructed with a series of sensors that detect hand and finger motions. Electromagnetic coupling between transmitting antennas and receiving antennas is used to provide information about the position and orientation of the hand.

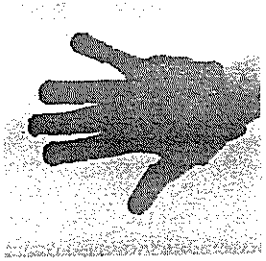


Figure 4.27 Data Glove

The transmitting and receiving antennas can each be structured as a set of three mutually perpendicular coils, forming a three-dimensional Cartesian coordinate system. Input from the glove can be used to position or manipulate objects in a virtual scene. A two-dimensional projection of the scene can be viewed on a video monitor, or a three-dimensional projection can be viewed with a headset.

4.10 Digitizers

A common device for drawing, painting, or interactively selecting coordinate positions on an object is a digitizer as show in figure 4.28. Digitization is the process of converting information into a digital format. In this format, information is organized into discrete units of data called bits that can be separately addressed, usually in multiple-bit groups called bytes.

These devices can be used to input coordinate values in either a two-dimensional or a three-dimensional space. Typically, a digitizer is used to scan over a drawing or object and to input a set of discrete coordinate positions, which can be joined with straight-line segments to approximate the curve or surface shapes. One type of digitizer is the graphics tablet or 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.

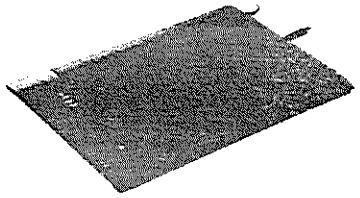


Figure 4.28 Digitizer

4.11 Voice Systems

Voice recognizers systems 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 spoke 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.

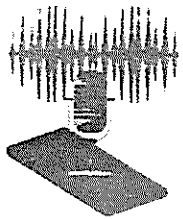


Figure 4.29 Voice Recognizers System

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.

4.12 Review Questions

Short Answer Questions

1. Give examples of touch screen.
2. Explain the working of Touch screen.
3. Give the advantage of mouse.
4. List the graphical input devices.
5. How does a graphic table work?

6. Explain what constraint is.
7. Explain the gravity field effects.
8. What is rubber band method?
9. What is dragging method?
10. Explain multiple selections?

Long Answer Questions

1. Explain about any three input devices.
2. Explain the logical classification of input devices.
3. Explain about light pen.
4. Explain about Keyboard.
5. Explain about mouse.
6. Explain the working about Touch screen.
7. Explain positioning techniques in detail?
8. Describe the technique of dragging.
9. Explain rubber band method and dragging?
10. What are the constraints available in interactive input techniques?
11. What is menu selection explain?
12. Explain selection by naming in computer graphics.

❑❑❑❑

NOTES

APPENDIX

MODEL QUESTION PAPERS

CONTENTS

Model Question Paper - 1

Section-A

I. Answer any Four questions. Each questions carries Two Marks (4 x 2 = 8)

1. Define Computer Graphics.
2. List the factors affecting CRT.
3. Define two dimensional transformations?
4. What is window and view port?
5. What do you mean by segment files?
6. Explain the gravity field effects.

Section - B

II. Answer any Four questions. Each questions carries Five Marks (4 x 5 = 20)

7. Explain raster scan display with neat diagram.
8. Explain the working of shadow mask CRT.
9. Successive two-dimensional scaling is multiplicative explain.
10. What you mean by parallel projection explain?
11. Explain rubber band method and dragging?
12. How does a graphic table work?

Section - C

III. Answer any Four questions. Each questions carries Eight Marks (4 x 8 = 32)

13. Explain the working of CRT, with a neat diagram. (8)
14. (a) Explain in detail the DDA line drawing algorithm with example (4)
    (b) How does beam penetration method works for displaying colors. (4)
15. Define 2 D transformation and explain with suitable illustrations translation, rotation. (8)
16. Describe the Cohen Sutherland algorithm for line clipping with suitable illustrations (8)
17. Explain Bezier curves and surfaces. (8)
18. Write short notes on:
    (a) Keyboard and Mouse. (4)
    (b) Touch panels. (4)

❑❑❑❑

Model Question Paper - 2

Section-A

I. Answer any Four questions. Each questions carries Two Marks (4 x 2 = 8)

1. Define Raster scan displays?
2. What is a Line cap?
3. Define translation?
4. List out the various Text clipping?
5. What is the use of control points?
6. List the graphical input devices.

Section - B

II. Answer any Four questions. Each questions carries Five Marks (4 x 5 = 20)

7. Explain random scan display with a neat diagram.
8. Prove that successive 2D rotation are additive.
9. Explain about regional code.
10. What do you mean by Perspective projection?
11. What is menu selection explain?
12. Describe the technique of dragging.

Section - C

III. Answer any Four questions. Each questions carries Eight Marks (4 x 8 = 32)

13. Discuss the working of CRT, with neat diagram. (8)
14. Explain Midpoint Circle algorithm? (8)
15. Explain two-dimensional shear with suitable examples. (8)
16. Explain Sutherland Hodgeman polygon clipping (8)
17. Explain the Z-Buffer algorithm for hidden surface removal. (8)
18. Write short notes on:
    a) Rubber band method (4)
    b) Mouse and Track ball (4)

❑❑❑❑

Model Question Paper - 3

Section-A

I. Answer any Four questions. Each questions carries Two Marks (4 x 2 = 8)

1. What is resolution?
2. What is kerned character?
3. Define reflection?
4. Explain about regional code.
5. How is curve clipping performed?
6. Give the advantage of mouse.

Section - B

II. Answer any Four questions. Each questions carries Five Marks (4 x 5 = 20)

7. Explain the color model
8. Explain Scan-line algorithm for area filling.
9. Give a 3x3 transformation matrix to reduce an object to half its original size.
10. Polygon data tables can be organized into two groups, what are they explain?
11. Define segment and draw the segment format diagram.
12. What is Positioning technique?

Section - C

III. Answer any Four questions. Each questions carries Eight Marks (4 x 8 = 32)

13. Explain Bresenham's line drawing algorithm? (8)
14. Give the sequence of steps that are required to rotate and scale an object through a general point. Also give its matrix representation. (8)
15. What are the different types of text clipping explain with example. (8)
16. Explain three-dimensional rotation. (8)
17. What are Octrees? How are they used to represent 3D objects? (8)
18. Write short notes on:
    a) Selection by name (4)
    c) Segment attributes. (4)

❑❑❑❑