Uttar Pradesh Rajarshi Tandon
Open University

Master of Computer Science

MCS-116

Computer Graphics

Uttar Pradesh Rajarshi Tandon
Open University

Master of Computer Science

MCS-116
Computer Graphics

BLOCK
1

RASTER GRAPHICS AND CLIPPING

UNIT-1

Introduction to Computer Graphics

UNIT-2

Graphics Primitives

UNIT-3

2-D Viewing and Clipping

Course Design Committee

Prof. Ashutosh Gupta
Director (In-charge)
School of Computer and Information Science, UPRTOU Prayagraj

Prof. Suneeta Agarwal
Department of CSE.
Motilal Nehru National Institute of Technology, Allahabad, Prayagraj

Dr. Upendra Nath Tripathi **Author**
Associate Professor
Department of Computer Science
Deen Dayal Upadhyaya Gorakhpur University, Gorakhpur

Dr. Ashish Khare
Associate Professor
Dept. of Computer Science, Allahabad University

Ms. Marisha
Assistant Professor (Computer Science)
School of Science, UPRTOU Prayagraj

Mr. Manoj Kumar Balwant
Assistant Professor (Computer Science)
School of Science, UPRTOU Prayagraj

Course Preparation Committee

Dr. Divya Kumar **Author**
Assistant Professor
Department of Computer Science & Engineering
Motilal Nehru National Institute of Technology Prayagraj

Prof. Abhay Saxena **Editor**
Dean, School of TCM
Dev Sanskriti Vishwavidyalaya, Haridwar-Uttarakhand

Prof. Ashutosh Gupta **Director (In-Charge)**
School of Computer & Information Sciences
UPRTOU Prayagraj

Mr. Manoj Kumar Balwant **Coordinator**
Assistant Professor (computer science)
School of Sciences, UPRTOU Prayagraj

ISBN :

Reprinted and Published by Vinay Kumar, Registrar Uttar Pradesh Rajarshi Tondon Open University Prayagraj - 2024.

Printed By: Chandrakala Universal Pvt. Ltd. 42/7 Jawahar Lal Neharu Road, Prayagraj.

UNIT-1 INTRODUCTION TO COMPUTER GRAPHICS

Structure :

1.1 INTRODUCTION

In the on going development of research era we create 2-D and 3-D pictures in computer graphics for research purposes. Hardware devices also plays a very important role in picture generation, many hardware device algorithms are created live storage of models and images of objects which are consequently used in several fields of computational mathematics engineering and many others.

The rest of the unit is structured as follows. In the first section lies the list of objectives of the unit section, 1.2 discuss the basics of computer graphics. Section 1.3 and 1.4 explain the graphics primitives and 2D viewing. Section 1.5 captures the summary of the unit and Section 1.6 and the unit deals with some Technical Questions for students to work out.

1.2 OBJECTIVES

After the end of this unit, you should be able to understand:

- The basics of computer graphics and various graphics primitives.
- Display devices, Refreshing & Video Displaying Devices, Input and output devices, etc.
- Different drawing and clipping techniques.

1.3 COMPUTER GRAPHICS

Computer graphics can be termed as the process which gets transformed and displays the information in a visual form. Computer graphics has become a

very significant feature in the present scenario, beyond technology, as in user interfaces, TV and commercial videos are Motion Pictures.

The fundamental concept of Computer Graphics places the two-way communication between users. The computer will receive signals from the input device and accordingly picture is transformed as soon as the command is given.

➢ **Computer Graphics**

Computer Graphics has three components as shown in Figure 1.1 below.

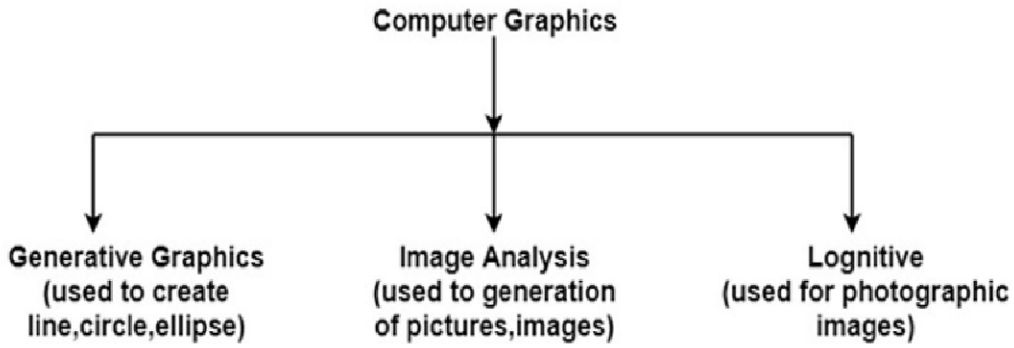


Figure 1.1: Three components of Computer Graphics [2]

Generative Graphics: It is an autonomous system that is used to create whole or part of images. Generally and autonomous system is one that is not human and independent features of images that would otherwise required decisions made directly .

Image Analysis : Image analysis is a way for providing theoretical fundamentals for solving problems that appears in areas like as biology, chemistry, astronomy etc. In computer graphics the main purpose of a computer is to create an image. Image processing to restore, resize, modify original pictures like TV scans machine perception of visual information. With image digitization comes the concepts of pixels (px), resolution and aspect ratio for considering Image quality which is used in advanced graphics.

Lognitive /Cognitive: Understanding the mental processes of visual perception provides your mind with the tools for the formation of images that establishes links with your viewer to convey the story you are telling.

Applications of Computer Graphics

1. **Display of information** - To produce different kinds and types of data 2D, 3D Graphics. A graphic tools are used for reporting complex data.
2. **Design** - Computer-Aided Design for mechanical engineering architectural systems animations and presentation graphics are used to produce different kinds of data.
3. **Entertainment** - It is an interface through which a user can have information.

4. **Simulation** - Photorealistic methods, morphing and animations are very useful in commercial art. For films and video monitor, 24 frames per second and 30 frames per second are required respectively.

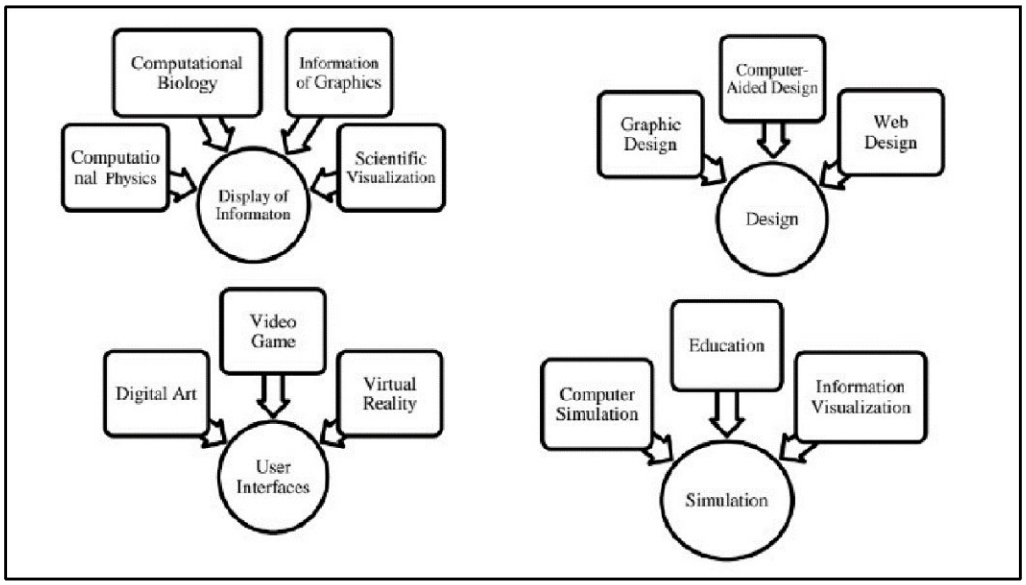


Figure 1.1: Applications of Computer Graphics [2]

Graphics Hardware

The graphics hardware is responsible for running computer games and recording video data, so the load on the CPU does not increase.

Graphics Processing Unit: GPU is a specialized CPU designed for executing video-related mechanisms.

Video Memory: Video memory is comparatively expensive then Computer memory and it is design to run faster to meet the requirements of the graphics core. There are different kinds of video memory available like DDR2 , GDDR2 and even GDDR3.

Integrated and Dedicated Graphics: The major difference between integrated and dedicated graphics lies on the location where Graphics Processing Unit is located and, the GPU has an access to its own video memory.

1.4 INPUT AND OUTPUT DEVICES

Input devices are used to feed data in the computer and convert them into a machine language which a computer can understand and use for the human purposes

Output devices produced the outcome of the processing data given to the machine and convert them into a human usable / readable forms.

Keyboards: keyboard is an input device primarily used to enter data into the system . Non graphic data such as image labels associated with the graphic display are inputted into the system by this device. Some efficient features are also provided like to enter screen coordinates, graphic functions etc.

General purpose keyboards have some common features like function and cursor control keys.

Touch Panel : A touch panel is an equipment through which users can directly interact with the computer system by touching it. A touch panel is basically used when the options are represented in graphical manner like graphical icons and we have to select or make choices among them and in it the input can be given using optical, electrical pointing devices.

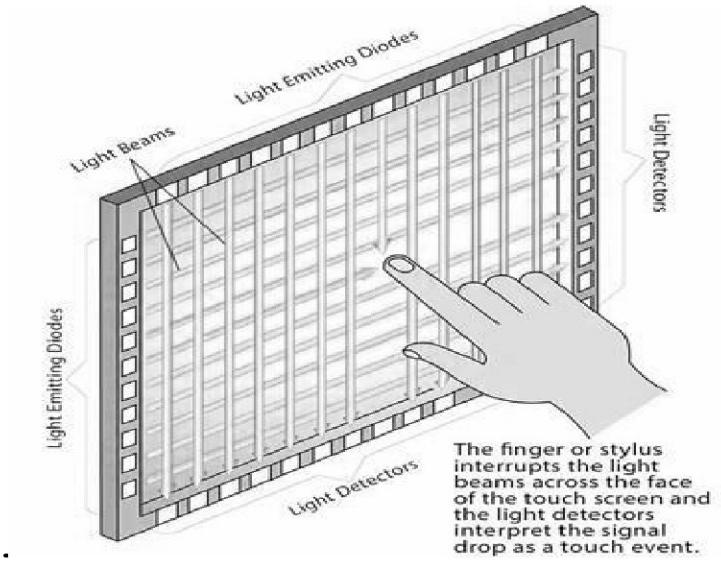


Fig1.2 Touch panel device

Optical touch panels use a line of infrared light emitting diodes for one horizontal edge and one vertical edge of the frame, Light detectors placed at opposite edges are used to check which beam are interrupted when the panel is touched and the beams which are crossed are interrupted and at that particular screen position, horizontal and vertical coordinates are identified and screen position is selected.

Light Pen : A light pen is a light sensitive input device basically designed to make selection among various things- for example selecting text, drawing pictures and user interaction with the computer screen. It functions good with CRT monitors because of the scanning pattern i.e. one pixel at a time as it can keep track of the scan time by the electron beam and check the pens position based on the latest time stamp of the scanning.

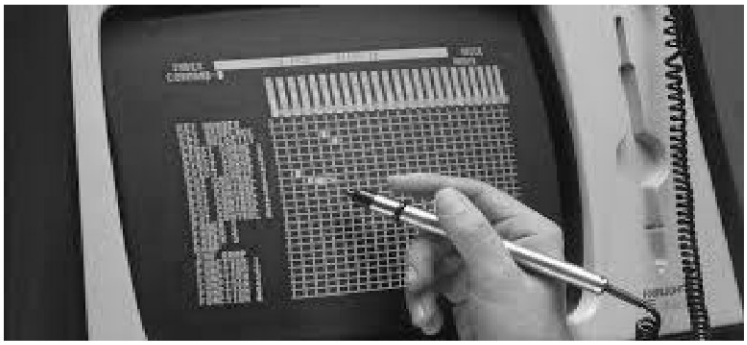


Fig.1.3 Light Pen Input device

Graphics Tablet: A graphics tablet (also called a digitizer, digital drawing tablet, pen tablet, drawing tablet, drawing pad, or digital art board) is a computer input device that enables a user to hand-draw images, graphics, and animations with a special pen-like stylus.

Fig. 1.4 network security policies

Plotters: Plotter is a computer output device and a vector graphical printer that provide hard copy of the output data which is given to the system in the form of instructions. It is used in various purposes like to print design of things such as ships, cars and buildings on a piece of paper with a pen, the major difference between plotters and printer is that the plotters are more precise and used in engineering where precision plays a very important role and also they are more costlier than printers .

Fig. 1.5 Types of Plotters

Film Recorders: A film recorder is a device which creates a 35mm slide or are films negative from a digital file. It works by recreating an image on a CRT which is already present in the Recorder Previous film recorder connected two computers by plugging in a controller board cable to the recorder

Fig.1.6 Film Recorder

1.5 DISPLAY DEVICES

The display device is a term used to describe the device used to view images or text. The picture is an example of a flat-panel display and the most common display used with computers today. The video monitor in a graphical system is the primary output device. Cathode Ray Tube (CRT) is the main part of the video monitor.

1.5.1 VIDEO DISPLAY DEVICES

Cathode-Ray Tubes (CRT) - still the most common video display device presently

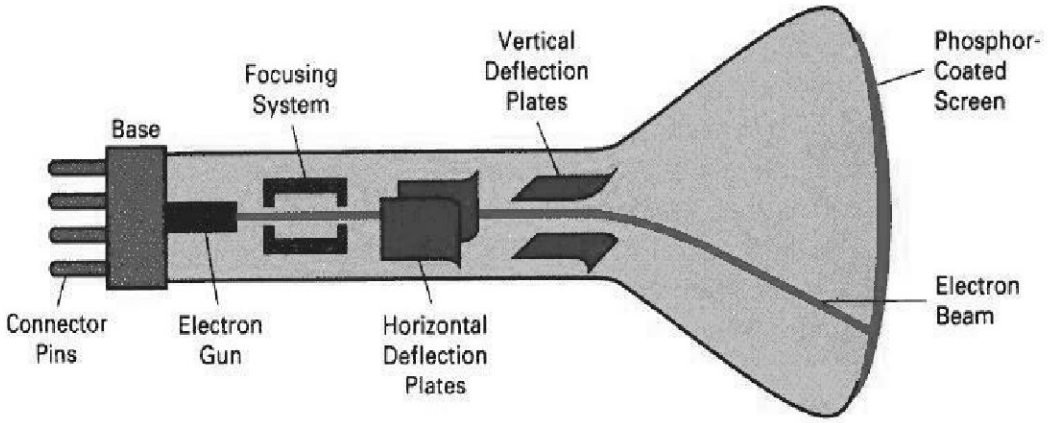


Fig 1.6 Electrostatic deflection of the electron beam in a CRT

A beam of electrons is emitted by an electron gun, which hits on the phosphor-coated screen passing through focusing and deflection systems. **Resolution** is termed as the total number of points portrayed on a CRT. Different coloured light spots were emitted by Different phosphors, combined to get a range of colours. **Shadow-mask** method is the common techniques for colour CRT display.

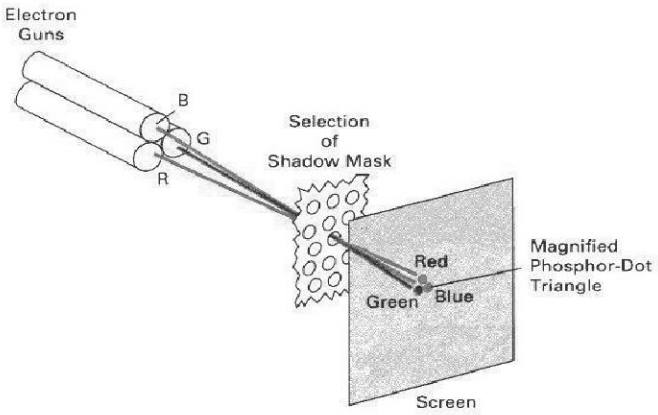


Fig.1.7 Illustration of a shadow-mask CRT

The phosphor, emits the light which fades very fastly , so the need for redrawing the picture occurs in a repeated manner. Two different mechanisms of redrawing are as: Raster-Scan and Random-Scan.

Raster Scan: In this technique, the electron beam is struck across the screen from top to bottom and one row. When it passes across the row, the intensity of the beam is turned on and off through to create the pattern of illuminated spots. This process of scanning is termed as refreshing, and when the complete scanning of a screen is done, the screen is called a **frame**.

The refreshing rate of a frame is termed as the **frame rate**, which is 60 to 80 frames per second. The **Frame buffer** is a memory area where the picture is stored. All the intensity values for coordinates or screen points are stored in this frame buffer, and this screen point is termed as a **pixel** (picture element).

Intensity of pixel defines the 1 (on) and 0 (off)on the black and white systems.

On color systems, bitmap is where the frame buffer stores the values of the pixels.

Pixmap is the each entry of 1 bit data of the pixel in a bitmap and color of the pixel are represented by the no. of bits occupied by an entry in the pixmap. For example - To display colours , the number of bits to represent each entry is 24 means 8 bits per red/green/blue channel and every channel we require 2^8=256 levels of intensity values. Intensity values of the screen coordinates are stored in this memory area and refresh buffer is used to retrieve the stored intensity values and accordingly painted one row (scan line) at a time on the screen as shown in the following fig.1.8 Cited below-

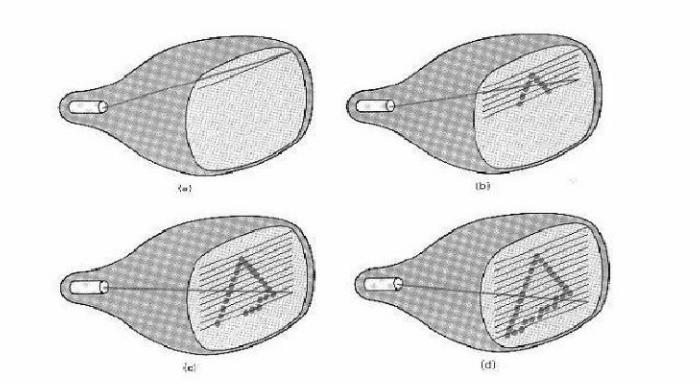


Fig 1.8 Raster Scan System

Random Scan: In this scanning technique, the part of the screen is focussed rather than scanning from left to right and top to bottom as in raster scan by the electron beam

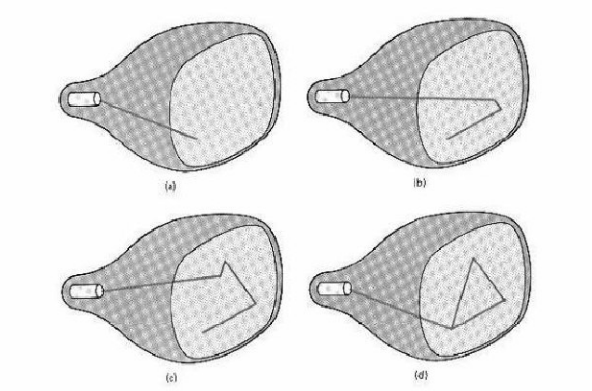


Fig 1.9 Random Scan Systems

It is also termed as vector display. In refresh display file pictures are stored as a set of line-drawing commands. To display a picture each component line is drawn using the commands given by the system cycles.

Direct-View Storage Tubes

A direct-view storage tube (DVST) in this picture information is stored in the form of charge distribution behind the phosphor coated screen. It has two electron guns are used - primary gun stores the picture pattern, the other is the flood gun which displays the picture. Compared to refresh CRT, DVST monitor also has advantages as well as disadvantages as it does not require refresh so it is very easy to view Complex images at high resolutions without flickering which makes it beneficial.

Disadvantages of DVST systems are that they normally do not display colors and that only specific selected part of a picture cannot be erased. To erase a section in a picture, the whole screen is erased and the modified picture is redrawn.

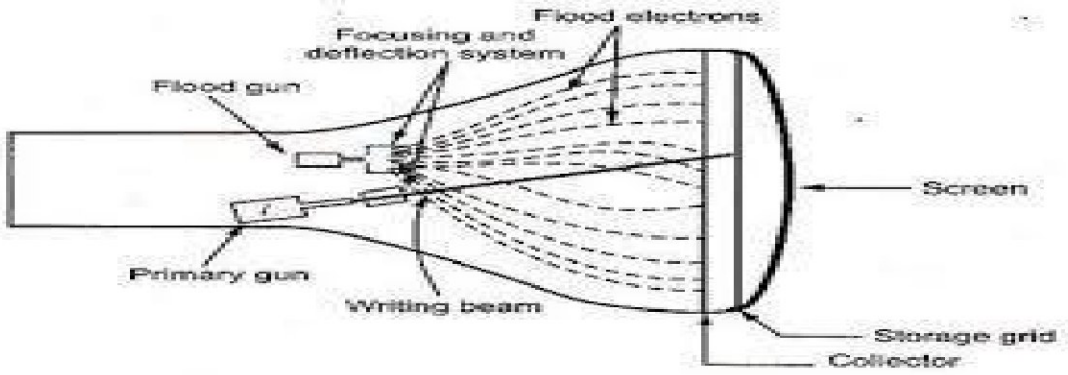


Fig.1.10 Direct view storage tubes

Flat-Panel Displays

The flat-panel display refers to the video devices that represented in reduced form in terms of requirements as comparison to the CRT. A major feature of these displays is that they are much slimmer than CRT. Flat-panel displays can be specified into two categories: **emissive displays** and **non-emissive displays**.

In the emissive display devices, the conversion of electrical energy into light is done. Some examples of emissive displays are Plasma panels, thin-film electroluminescent displays, and Light-emitting diodes. Non-emissive displays optical effects are used to convert sunlight or other light from one or the other sources into a graphical pattern. The liquid-crystal device is the most important example of non-emissive displays.

Emissive Displays

- **Plasma Panel**

It is a computer monitor display in which tiny bit of plasma is used to illuminate each pixel on the screen.

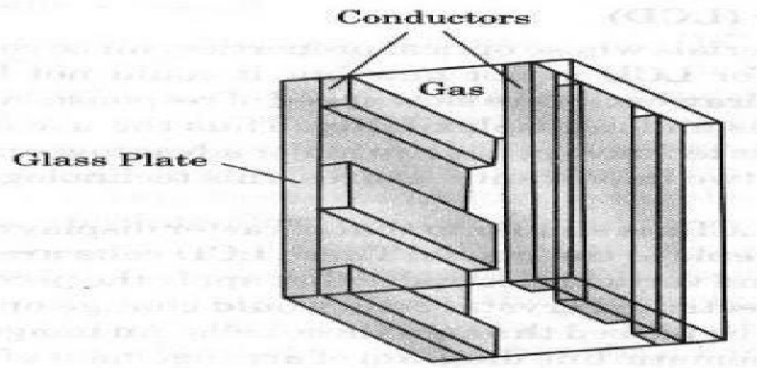


Fig 1.11 Schematic of Plasma panel

Plasma displays are much thinner in comparison to CRT displays and much brighter when compared to LCD. Plasma displays are also referred as "thin-panel" display and are used to display either analog or digital video signals.

Non-Emissive Display

➢ **LCD Panel**

LCD (liquid crystal display) is the display technology used in notebook and Computers which are smaller in size. It allows thinner displays as compared to cathode ray tube (CRT) technology. LCDs work on the technique in which light is blocked rather than emitting it. Consequently it requires less power than LED.

The working principle of LCDs is similar to that of raster display on a refresh CRT, as described earlier. Here the LCD cells replace the pixels. These LCD cells are installed in the matrix form with a number of vertical and horizontal electrodes which requires the necessary driving voltage from the crystal cells. Thus the electrified crystal cells would change optical properties which would control the amount of light to be passed through these cells. An image will be created on the screen and viewed by the user. Schematic line diagram of arrangement of the LCD panel is shown in figure below.

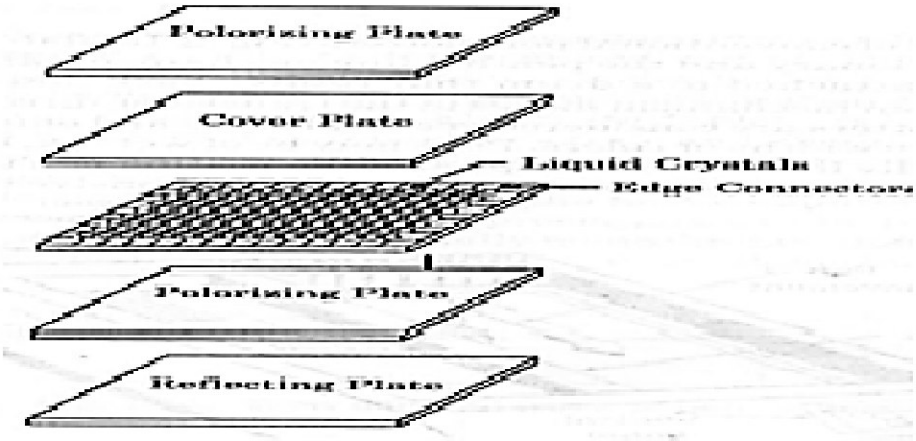


Fig. 1.12 Schematic of LCD

1.6 SUMMARY

In the above unit, we had surveyed the fundamentals of computer graphics, which fall under several categories like image analysis and the significant hardware and software features of computer graphics and systems. Hardware features include video monitors, hard-copy devices, keyboards, and other input-output devices use for graphics. The refreshing video devices like Raster and Random scan systems are explained. Raster refresh monitors are the necessary graphics display devices. The frame buffer is used to store intensity information for the position of each pixel.

Other video display devices, In particular, flat-panel display technology is on the stage at a fast rate, and these display devices may replace raster displays in the coming days/ or future. In the present scenario, flat-panel screens are mostly used in smaller computers. Some examples of this technology are plasma panels and liquid-crystal devices.

Hard-copy devices for workstations in graphics are printers and plotters. Printing techniques include dot matrix, laser, ink-jet, etc. Plotter methods involve

pen plotting and amalgamation of printer-plotter devices. Graphics software can also be categorized as applications packages or programming packages Applications based graphics software are CAD packages, drawing and painting programs, graphing packages, and visualization programs. Programming packages in computer graphics are PHIGS, PHIGS+, GKS, etc.

1.7 TECHNICAL QUESTIONS

1. Illustrate the operating characteristics for the given display technologies: raster refresh systems, vector refresh Systems?
2. Let a RGB raster scan system is to be designed using an 8 ×10-inch screen, given resolution of 100 pixels per inch in every direction. If bits per pixel are to be stored in frame buffer, what amount of storage in bytes is required for the frame buffer?
3. How long would it take to load a 640 by 480 frame buffer with 12 bits per pixel and 10^5 bits can be transferred per second? How long it take to load a 24-bit per pixel frame buffer with a resolution of 1280 by 1024 using this frame transfer rate?
4. Suppose there are two raster systems with resolutions of 640 × 480 and 1280 × 1024. How many pixels could be accessed per second in each of the given systems by a video display controller whose refresh rate is 60 frames per second? Calculate the access time per pixel in each given system?
5. Consider a video monitor that measures 12 × 9.6 inches with the resolution of 1280 × 1024 and the aspect ratio is 1, calculate the diameter of each screen point?
6. A non-interlaced raster monitor with a resolution of $a \times b$, a frame rate of f frames per second, a horizontal retrace time of T_{hori} and a vertical retrace time of T_{ver}. Calculate the fraction of the total refresh time per frame spent in retrace of the electron beam?
7. Determine the resolution defined in pixels per cm, in the x and y directions for the display monitor on the system.
8. Illustrate the functioning of CRT and shadow masking techniques?
9. Illustrate the advantages and disadvantages of a three-dimensional video monitor using a varifocal mirror with a stereoscopic system?
10. List the different Input and output components that are typically used with virtual reality systems. Also explain how users interact with a virtual scene displayed with different output devices, such as two-dimensional and stereoscopic monitors.
11. Illustrate the usage of large-screen displays.
12. Differentiate between a general computer graphics system designed for a programmer and other designed for some specific application like some architectural design?
13. Which of the following contained in computer graphic system? [15]

a) Processor

b) Frame Buffer

c) Display Device

d) All of the above

14. Which of the following are the properties of the video display device?

a) Persistence

b) Resolution

c) Aspect Ratio

d) All of the above

15. Which of the following are the advantages of view storage tubes?

a) Refreshing is not necessarily

b) Without flicker, very complex images can be exhibit at very high resolution

c) Refreshing of the screen is not needed Refreshing of the screen is not needed

d) All of the above

16. Which of the following are characteristic of vector graphics?

a) Vector graphics are consisting of paths

b) Vector image do retain appearance regardless of estimate

c) Vector images are scalable

d) All of the above

17. Which of the following are characteristic of parallel projections?

a) Coordinate positions are changed to the view plane along parallel lines

b) Preserves the related proportions of objects

c) It is used in drafting to produce scale drawings of 3Dobjects

d) All of the above

18. What is Aspect Ratio?

19. What are the differences between vector and raster graphics?

20. What are the essential applications of computer-graphic?

21. What are the different kinds of display devices?

22. What are the differences between emissive and non-emissive display?

UNIT-2 GRAPHICS PRIMITIVES

Structure :

2.1 INTRODUCTION

There are many techniques for drawing and filling of any picture. The line drawing algorithms are used for approximate the line segment.

For drawing the rectangular shape on the display such that the illumination is on the border for generating circle. For filling the object on the display device, the polygon filling technique is used such that it picks a point inside the object and starts to fill until it hits the desired boundary, we can even draw the boundary and the colour inside it with the different colour.

Pixel array is useful for describing internally the shapes and colour of the objects, there are three channels for generating as well as filling the image with the different colours. We continue the discussion on generating as well as the filling of the objects using the computation.

Point

A Point has position in space. The characteristics of the point is that the point has the exact place or the location on the plane.

Line

A line is the geometrical 1D figure which has the property of zero thickness and can be extended in both the direction to infinite length. It is the basic element of the coordinate geometry.

2.2 OBJECTIVES

After the end of this unit, you should be able to understand:

- The basics of Line Drawing Algorithms which will be very helpful in edge detection.
- Circle Drawing Algorithms
- Polygon Filling Algorithm.

2.3 POINTS AND LINES

Point plotting is the technique in the computer graphics which needs just two points or depending on the application more than two points on the coordinate system. It is the process of taking the position of the point in the one coordinate system and then transform it using the application program that performs operations on the point and make it in the way so that it make some sense for the output device.

If we take random scan as an example system stores in the display list the useful and appropriate point plotting instructions and the deflection voltage has been converted from the instructions which are in the coordinate values and then these instructions allows them to plot on the screen locations using the electron beam on every refresh cycle.

If we take the raster scan as an example system point plotting is done on the basis of setting the value of the bits corresponding to a specific screen position and make that Bit to 1 in the specified frame buffer. So, it works on the principle such that whenever it encounters a bit value 1 it emits the burst of electron beam across the horizontal line.

Line drawing in the computer graphics is used to plot using the intermediate positions by the use of two points in the coordinate system along the path in which points known as the endpoints. The devices which comes under the analog device for e.g., vector pen or random-scan display, straight line can be drawn easily from one extreme point to the other extreme point.

For the algorithms defined as discussed in this chapter, there are some device-level algorithms in which the position of the object is there in the integer device coordinates. For the algorithms to work pixel positions are referenced includes preventing the data from being accessed by unapproved sources, shielding it from modification or harm and Deploying or Channelizing mechanisms to recover data from data losses and breaches.

```
setPixel (x, y)
```

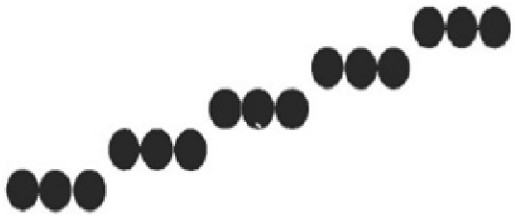


Figure 2.1 Stairstep effect

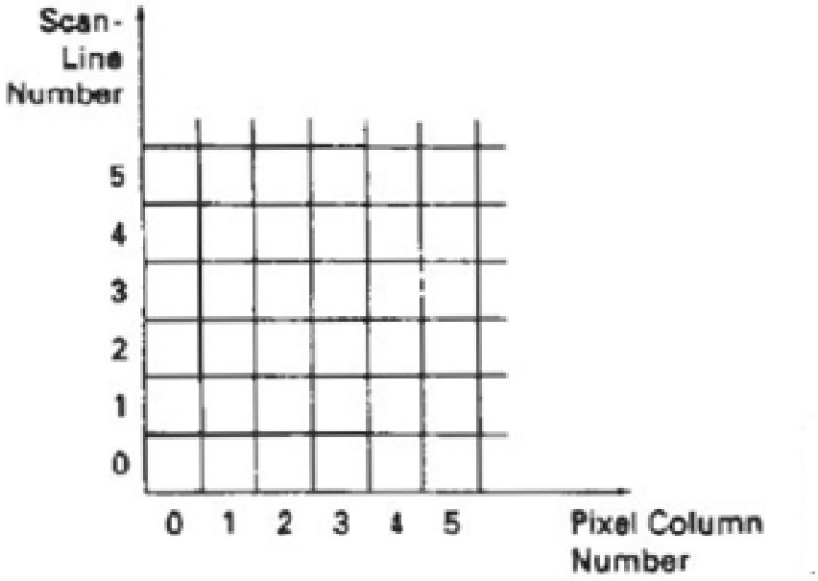


Figure 2.2 Pixel Positions referenced by scan-line

Pixel positions are referenced using the ability to retrieve the current frame buffer-intensity by setting this up for a specific location. There is a low-level function for it which help to accomplish.

getpixel (x,y)

2.4 LINE DRAWING ALGORITHMS

Line-Drawing Algorithms

The equation of the line on the cartesian coordinate system or the equation for the slope-intercept is :

$$y = m \, . \, x + b \quad (3.1)$$

Where m = slope of the line

b = intercept of the line.

So if we have the two points on the extreme of the line segment which are coined as (x1,y1) and (x2,y2) as we show that line using the points which can be shown in the figure below.

We find out the values of the slope and the intercept m and b respectively using the below equations:

$$m = \frac{y_2 - y_1}{x_2 - x_1} \tag{3.2}$$

$$b = y_1 - m \, . \, x_1 \tag{3.3}$$

The algorithms for finding out or to display the lines in the coordinate system are in the equation 3.1 and the appropriate calculations are in the equations respectively in the equation 3.2 ad 3.3

For the given coordinates the difference between the points or we can say the interval between the x coordinate between the points along a line the corresponding Δy can be fine out:

$$\Delta y = m \, \Delta x \tag{3.4}$$

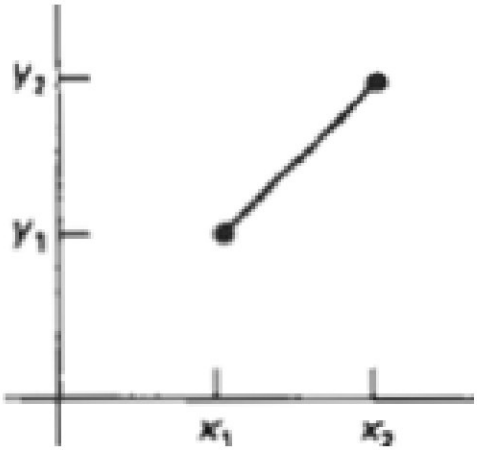


Figure 2.3 Line path between endpoint

In the same way we can calculate for the x using the interval of y coordinates in the below equation:

$$\Delta x = \Delta y / m \tag{3.5}$$

These equations are the basic form for finding out the different voltages in the analog devices.

So the idea is whenever the magnitude of the slope of the line is < 1 then it can coined as the small proportional to the deflection voltage and the respective deflection in the y direction is the Δy.

For the lines which are having the slope with the magnitude >1, Δy is set to be proportional to the Δx with the small vertical deflection as per the values of the points in the coordinates system.

The system which are associated with the raster scan, lines are plotted with the help of the specific pixels. Pixel separation defines the appropriate step sizes in the vertical as well as the horizontal directions.

So in the below figure we define the scan process for the lines whose target points are associated with x1 and x2.

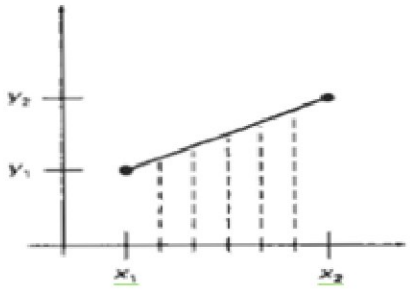


Figure 2.4 Segment with five samples.

In computer graphics line is a terminology which connect two points, so the concept strengthens with the help of the following three algorithms.

2.4.1 DDA ALGORITHM

The digital differential analyser which generally known as DDA line algorithm, which uses the concept of scan-conversion which calculate either the Δx or the Δy. The algorithm works in a way such that the unit intervals are plotted on the sample line in the one coordinate and determine respective integer point values nearest the five positions along the path for the other coordinates.

Now on taking the first line with the positive slope, we calculate the successive y values based on the sampling at the unit x intervals which is like interval for x is 1 in case the slope < 1.

$$Y_{p+1} = y_p + m \qquad (3.6)$$

The subscript that defines with the coordinate points *p* takes integer values starting from 1,

It works like for the first point it increases by 1 and keep on increase the value such that the final endpoint reaches, As we know that the value of slope can be anywhere in between the 0 and 1.

Now if we take or we calculate the slope of the line is greater than 1. Then we can calculate the succeeding x values with the help of the y interval, we keep on increase by 1 till the endpoint.

$$X_{p+1} = x_p + 1/m \qquad (3.7)$$

So the assumptions are that the line should be processed from the left to the right extreme point and these assumptions are made in the equation 3.6 and 3.7, and in case we want to change the direction of processing from the right to the left then the interval of x is taken as -1.

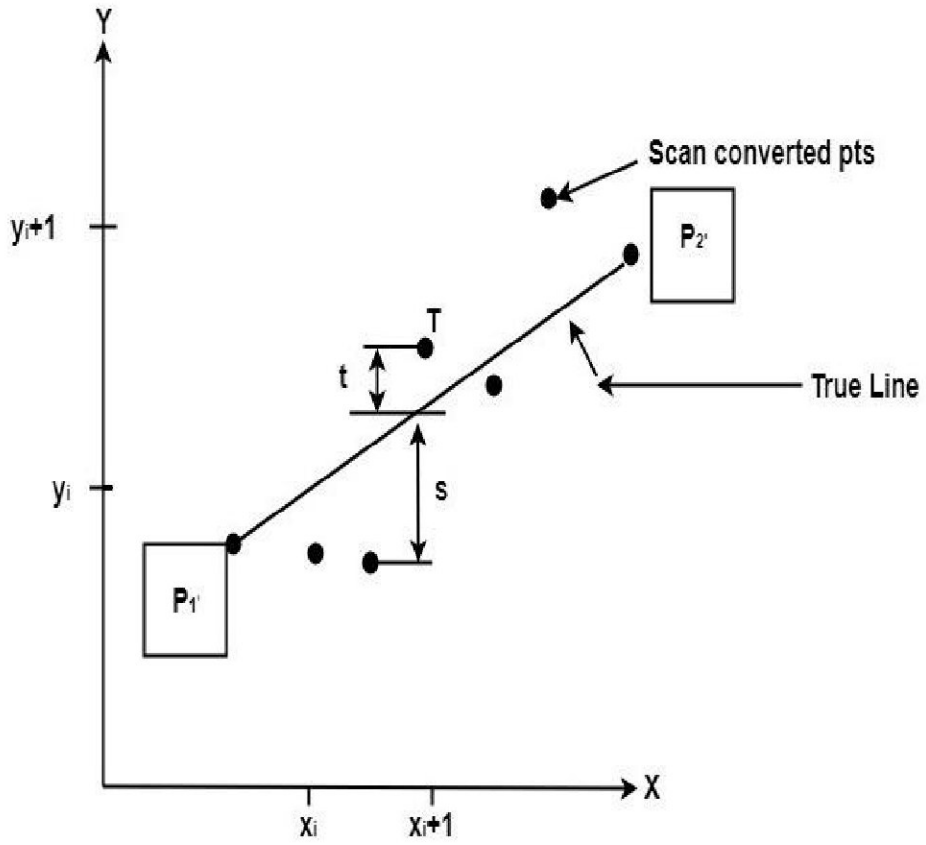


Fig 2.5 Scan converting line

$$Y_{p+1} = y_p - m \qquad (3.8)$$

Same applies to the x coordinate as when the slope is greater than 1.

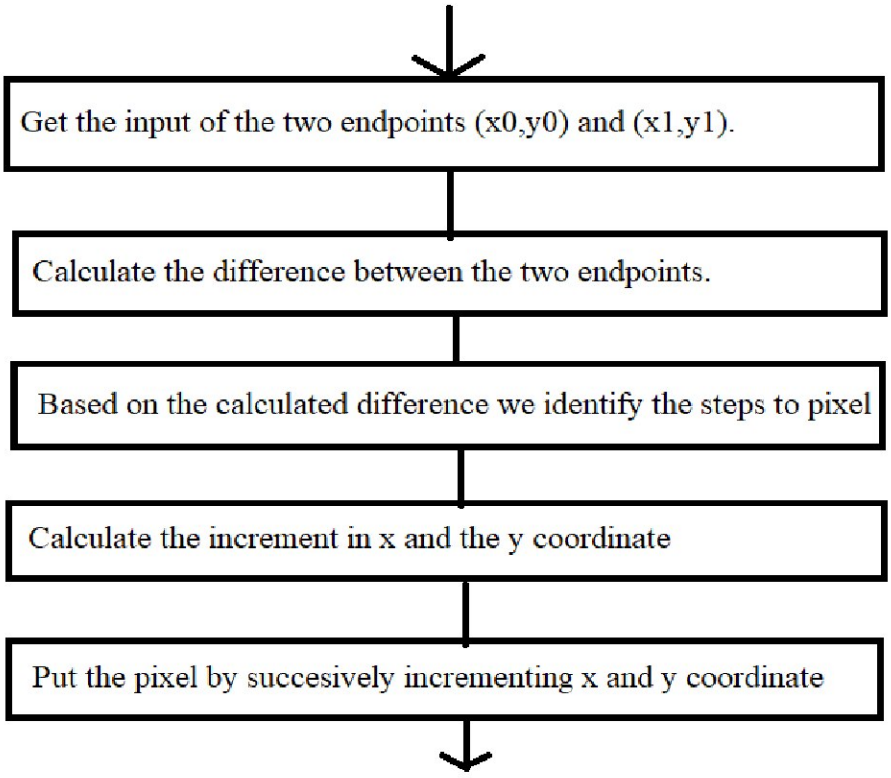
$$x_{k+1} = x_k - 1/m \qquad (3.9)$$

The equations from 3.6 to 3.9 calculate the pixel positions even for the negative slope along the line which gives the fruitful result. So the x interval is 1 in case the start point is at the left and the slope is less than 1 same applies to in case the start point is from the right then we set $\Delta x = -1$ same applies to the y interval.

The algorithm steps are below and the procedure of the algorithm can be summarized as that the algorithm need the two endpoints as an input in the form of pixel positions. The differences between the x and the y extremes are define as the dx and the dy as the parameter to the equation.

At start we have to determine the offset to generate the next coordinate pixel position and then we compare according to if magnitude of difference of x is greater than the difference of y then the other is set to 1.

Digital Differential Analyzer (DDA) algorithm is the simple line generation algorithm which is explained step by step here.



2.4.2 BRESENHAM'S LINE ALGORITHM

This algorithm is there to find the close approximation between the two points for the straight Line and it also determines the n dimensional raster for the line. This is the algorithm which is the efficient algorithm for finding raster line and has the ability to convert the line to the form which can be adapted to display circles and other polygon or curves.

This algorithm is generally used for the scan conversion of the line. It only uses the addition, subtraction and multiplication operations as these operations are known as the fastest operation in the mathematics it can be used to generate the lines very quickly. It allows only the integer values.

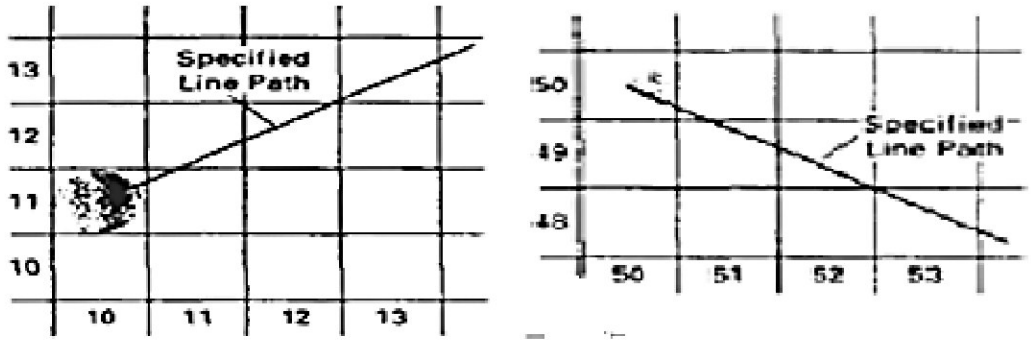


Figure 2.6 Section of a display

The accumulation of the different sampling of the x intervals involves the rounding off the error in successive addition of the floating-point increment. The scan line positions are defined by the vertical axes, and the pixel columns are identified by the horizontal axes. It allows the drift of pixel position.

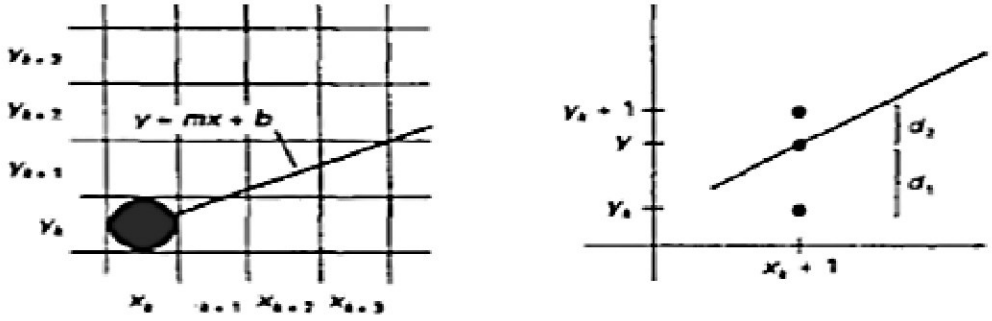


Figure 2.7 Showing pixel in a column

To define the Bresenham's approach, we- first consider the scan-conversion process for lines with positive slope less than 1. Pixel positions along a line path are then determined by sampling at unit x intervals. The y coordinate on the mathematical line at pixel column position x_k+l is calculated as-

$$y = m\,(x_k+1) + b \qquad (3.10)$$

Then

$$d_1 = y - y_k$$

$$= m\,(x_k+1) + b - y_k$$

and

$$d_2 = (y_k+1) - y$$

$$= y_k + 1 - m\,(x_k + 1) - b$$

The difference between these two separations is

$$d_1 - d_2 = 2m(x_k+1) - 2y_k + 2b - 1 \qquad (3.11)$$

$$m = \Delta y / \Delta x$$

where Δy and Δx are the vertical and horizontal separations of the endpoint positions, and defining:

$$p_k = \Delta x\,(d_1 - d_2)$$

$$= 2\Delta y \cdot x_k - 2\Delta x \cdot y_k + c \qquad (3.12)$$

The sign of p_k is the same as the sign of $d_1 - d_2$ since $\Delta x > 0$ for our example, Parameter c is constant and has the value $2\Delta y + \Delta x(2b - 1)$, which is independent of pixel position and will be eliminated in the recursive calculations for p_k. At step k + 1, the decision parameter is evaluated from Eq. 3.12 as

$$p_{k+1} = 2\Delta y \cdot x_{k+1} - 2\Delta x \cdot y_{k+1} + c$$

Subtracting Eq. 3-12 from the preceding equation, we have

$$p_{k+1} - p_k = 2\Delta y (x_{k+1} - x_k) - 2\Delta x (y_{k+1} - y_k)$$

But x_{k+1}, = $x_k + 1$ so that

$$p_{k+1} = p_k + 2\Delta y - 2\Delta x (y_{k+1} - y_k) \qquad (3.13)$$

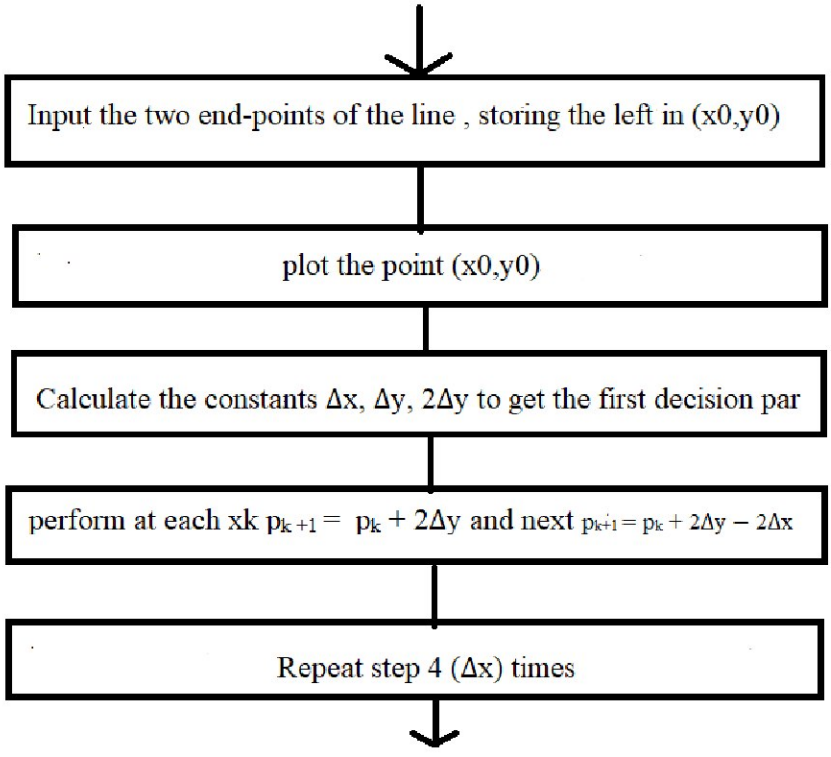
where the term $y_{k+1} - y_k$ is either 0 or 1, the first parameter is defined using the $\Delta y/\Delta x$:

$$p_0 = 2\Delta y - \Delta x \qquad (3.14)$$

Below is the procedure to draw the line using the Bresenham's line drawing algorihm with a positive slope less than 1.

For m > 1, find out whether you need to increment x while incrementing y each time. After solving, the equation for decision parameter p_k will be very similar, just the x and y in the equation gets interchanged.

There are the constants which are $2\Delta y - 2\Delta x$ and the $2\Delta x$ are define in the line and are calculated for each and every line to be converted as the arithmetic only allows addition and the subtraction.



2.5 CIRCLE DRAWING ALGORITHMS

Circle Generating Algorithm

We need to choose the nearby pixels from a printed pixel which can then form the arc and used to draw a circle.

2.5.1 PROPERTIES OF CIRCLES

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

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

We could use this equation to calculate the corresponding y values at each position as

$$y = y_c \pm \sqrt{r^2 + (x_c - x^2)} \tag{3.15}$$

Figure 3.9 shows the circle with the coordinates (xc, yc) and then the radius of the circle which defines the circle and in the figure 3.10 the spacing between the plotted pixel position is not uniform. We could even adjust the spacing by interchanging x and the y coordinate for the absolute value.

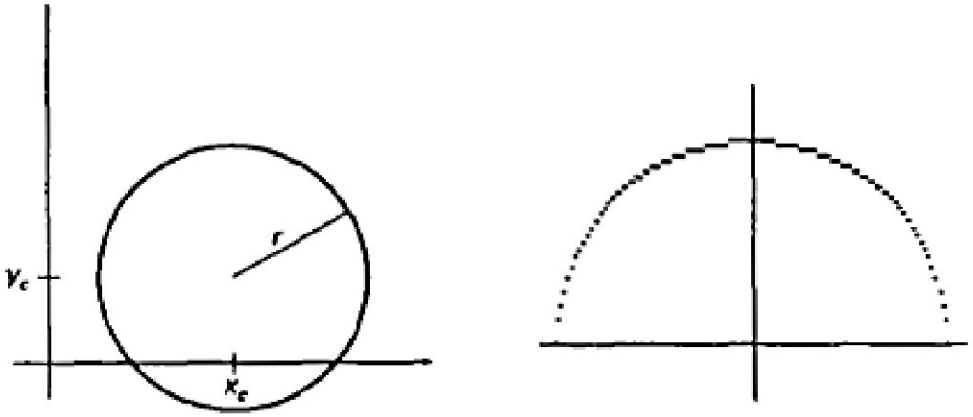


Figure 2.8 Circle with center coordinates (Xc, Yc) and radius r and the positive half of the circle plotted with Eq.3.15

$$x = x_c + r \ cos\, cos\, \theta$$

$$y = y_c + r \ sin\, sin\, \theta \tag{3.16}$$

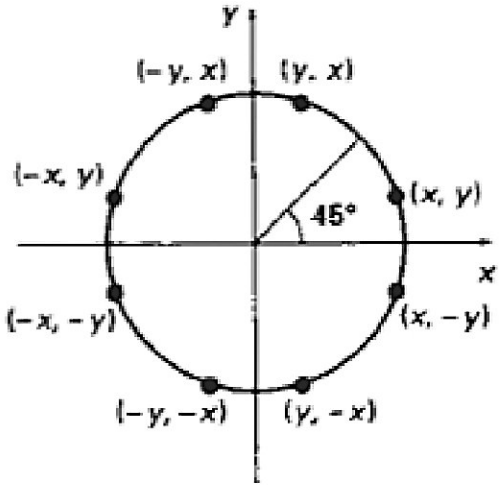


Fig. 2.9 Symmetry of a circle: calculation of circle points (x, y) in one octant yields the points to other seven octants.

2.5.2 MID-POINT CIRCLE DRAWING ALGORITHM

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

$$f(x,y) = x^2 + y^2 - r^2 \quad (3.17)$$

$$f(x,y) \begin{cases} < 0 & \text{if } (x, y) \text{ is inside the circle boundary} \\ = 0 & \text{of } (x, y) \text{ is on the circle boundary} \\ > 0 & \text{if } (x, y) \text{ is outside the circle boundary} \end{cases} \quad (3.18)$$

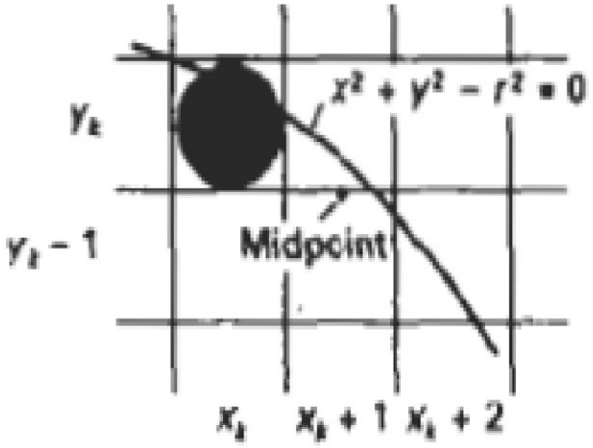
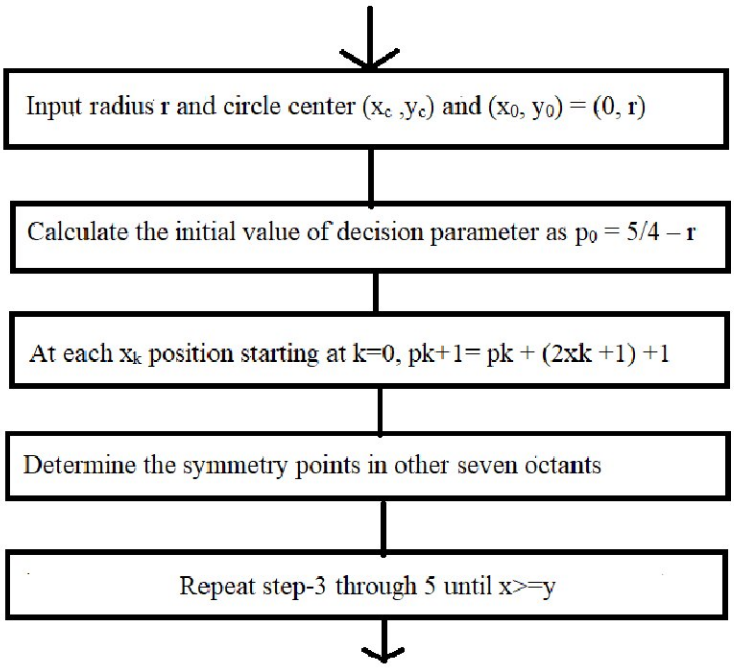


Fig.2.10 Midpoint between the two candidate pixels at sampling position (x_k +1)

We can summarize the steps in the midpoint circle algorithm as follows:

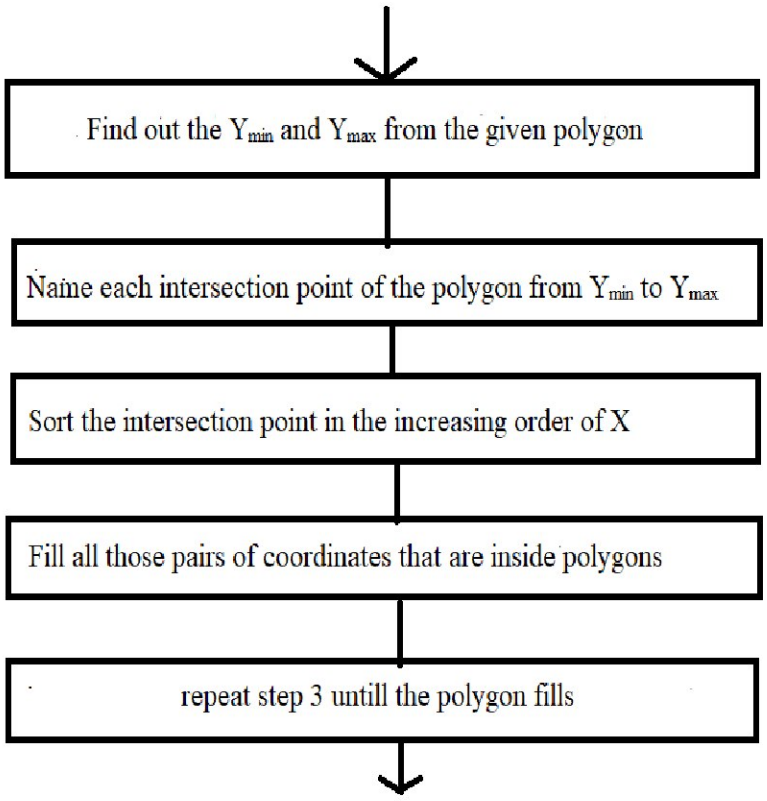


2.6 POLYGON FILLING ALGORITHM

2.6.1 SCAN LINE

It is the algorithm which allows the filling of the polygons using the horizontal lines or in technical term scan line. The concept is that the pixel which falls on the border of the polygon are determined in order to color.

This algorithm works by intersecting a scanline with polygon edges and fills the polygon between pairs of intersections. The following steps depict how this algorithm works.



2.7 SUMMARY

The graphic primitives discussed in this chapter provide the basic tools for constructing pictures with straight lines, curves, filled areas, patterns, and text. Examples of pictures generated with these primitives are given in Figs. 3-5 and 3-5.

Three methods that can be used to plot pixel positions along a straight-line path are the DDA algorithm, Bresenham's algorithm, and the midpoint method. For straight lines, Bresenham's algorithm and the midpoint method are identical and are the most efficient Frame-buffer.

The scan-line fill algorithm is an example of filling object interiors using the odd-even rule to locate the interior regions. Other methods for defining object interiors are also useful, particularly with unusual, self-intersecting objects.

Note: Kindly give a working example of DDA Algorithm, Bresenham's Algorithm and other algorithms for the better understanding of the student.

2.8 TECHNICAL QUESTIONS

1. Explain-scan-line-fill-algorithm-with-an-example?
2. Illustrate the loopholes of DDA line Algorithm and measures taken to overcame the problem?
3. Implement DDA to draw a line from (3, 4) to (7, 6)?
4. Draw a circle having radius as 9 and center of circle (90, 90) using the midpoint circle algo.
5. Implement the *set pixel* routine in Bresenham's line algorithm.
6. Which are line drawing algorithms?
   a) DDA Algorithm
   b) Bresenham's Algortithm
   c) All of the above
   d) None of the above
7. What are the special case of polygon vertices?
   a) If both lines intersecting at the vertex are on same side of scanline.
   b) If the lines intersecting at the vertex are on opposite side of scanline.
   c) All of the above.
   d) None of the above.
8. What are the properties of circle?
   a) Circle having equal radii are congruent.
   b) Circle having different radii are similar.
   c) Chords are equidistant from the centre are equal in length.
   d) All of the above.
9. Bresenham's Algorithm seeks to select the optimum raster locations that represent a:
   a) Straight line
   b) Curve line

c) Polygon

d) None of these

10. In Bresenham's circle algorithm, if points are generated from 900 to 450 and (x,y) are the Coordinate of last scan converted pixel then the next pixel coordinate is

a) (x+1,y+1)or(x-1,y-1)

b) (x+1,y)or(x,y+1)

c) (x,y+1)or(x+1,y-1)

d) (x+1,y)or(x+1,y-1)

[30]

UNIT-3 2-D VIEWING AND CLIPPING

Structure:

3.1 INTRODUCTION

Clipping is a computer graphics process which removes the objects, lines, or segments of line, which are outside of the viewing pane.

We use clipping in computer graphics primarily to remove lines, objects, or segments of line that are outside of the viewing pane. It is necessary to remove those points which are behind the viewer before getting the view because the viewing transformation is insensitive to the position of points with respect to the viewing volume.

When we want to show a large portion of a picture, then not only translation and scaling is necessary, but also the visible part of the picture is identified. The process of identifying is not easy. There are some parts of the image which are inside, while other parts are partially inside. The elements or lines that are partially visible will get omitted.

A process called clipping is used to decide the visible and invisible portion. Each element into the invisible and visible portion is determined by clipping. The visible portion gets selected and the invisible portion gets discarded.

There are three types of lines:

1. **Visible**: The lines which are entirely inside the window are considered as visible.
2. **Invisible**: The lines which are entirely outside the window are considered as invisible.
3. **Clipped**: The lines which are partially outside the window and partially inside is clipped. For clipping, we need to determine the intersection point of a line with the window.

We can apply the clipping through software as well as hardware. In some computers, hardware devices do clipping automatically. If hardware clipping is not available in the system then we use software clipping.

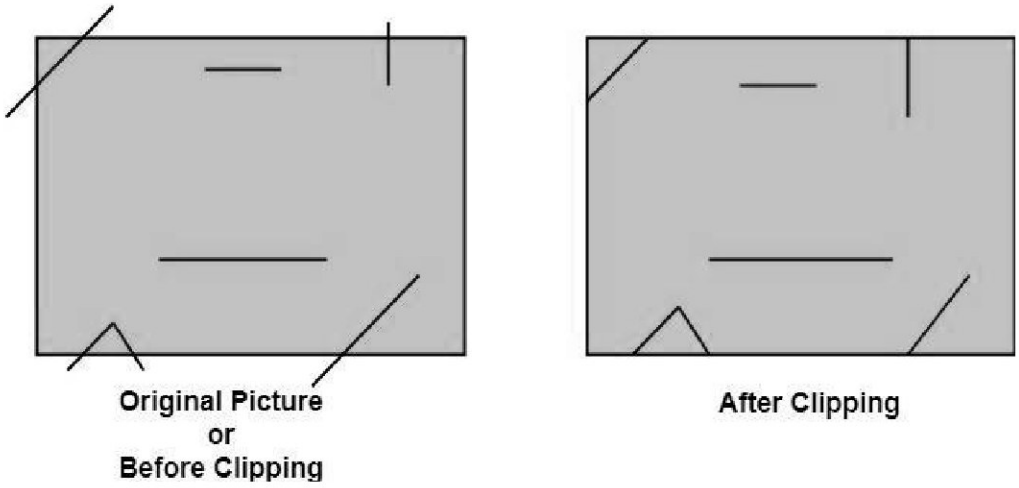


Fig.3.1

Three Kinds of Clipping are:

1. Point Clipping.
2. Line Clipping.
3. Polygon Clipping.

3.2 OBJECTIVES

By the end of this unit, we will understand:

- The basics of Line Clipping, point clipping and algorithms related to it.
- Polygon Clipping and algorithms related to it.
- Windowing algorithms.

3.3 POINT CLIPPING

- **|Point Clipping**

It is used to determining, whether the point is inside the window or not.

For this following conditions are checked.

1. $x \leq xmax$
2. $x \geq xmin$
3. $y \leq ymax$
4. $y \geq ymin$

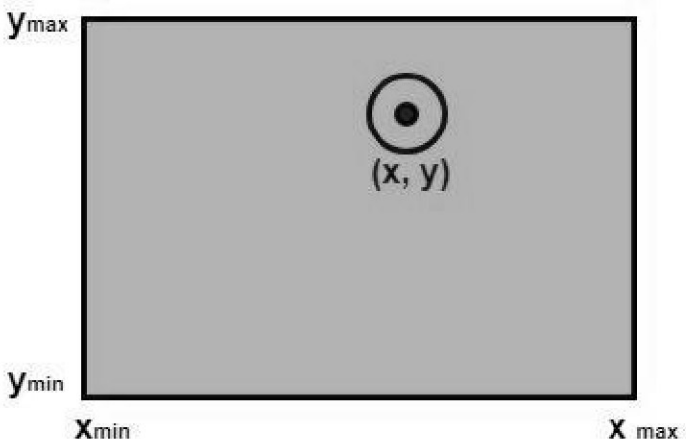


Fig. 3.2

1. Get the corner coordinates of both viewing plane.
2. Get the point coordinates.
3. Check whether the point lies between four corner coordinates of viewing pane.
4. If it lies inside the region, then display the point otherwise discard it.

3.4 LINE CLIPPING

➢ **Line Clipping**

It is same as point clipping. We cut the portion of the line that lies outside of the viewing window and keep the portion which lies inside the window.

➢ **Cohen-Sutherland Line Clippings**

Step1: Calculate positions of both endpoints of the line.

Step2: Perform OR operation on both of these end-points.

Step3: If the OR operation gives 0000

Line is considered to be visible

else

Perform AND operation on both endpoints

If And ≠ 0000

The line is invisible

else

And=0000

Line is considered the clipped case.

Step4: If a line is clipped case, find an intersection with boundaries of the window.

m=(y2-y1)(x2-x1)

1. If bit 1 is "1" line intersects with left boundary of rectangle window.

   y3=y1+m(x-X1)

   Where X = Xwmin where

   Xwmin is the minimum value of X co-ordinate of window

2. If bit 2 is "1" line intersects with right boundary.

   y3=y1+m(X-X1)

   Where X = Xwmax

   X more is maximum value of X co-ordinate of the window

3. If bit 3 is "1" line intersects with bottom boundary.

   X3=X1+(y-y1)/m

   Where y = ywmin

   ywmin is the minimum value of Y co-ordinate of the window

4. If bit 4 is "1" line intersects with the top boundary

   X3=X1+(y-y1)/m

   Where y = ywmax

   ywmax is the maximum value of Y co-ordinate of the window.

These are the possible cases for a given line:

1. **Inside the given rectangle completely:** If bitwise OR of region of two end points of line is 0 then both points are inside the rectangle.
2. **Outside the given rectangle completely:** If both the end points share at least one outside region, then the line does not cross the visible region. (bitwise AND of endpoints != 0).
3. **Inside the window partially**: Both the end points are in different regions. In this case, the algorithm finds only one of the two points is outside the rectangular region. The point of intersection of the line and the boundary of the window becomes new corner point and the algorithm repeats.

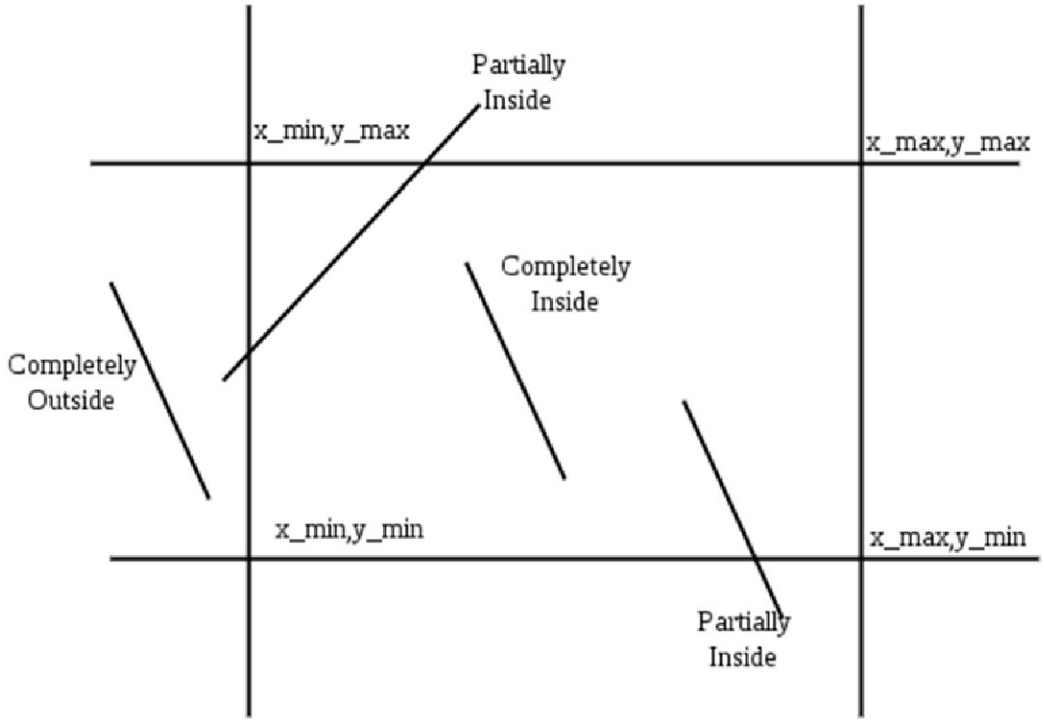


fig 3.3

- **Cyrus-Beck Line Clipping Algorithm**

It is a line clipping algorithm that is made for convex polygons.

It is also used for line clipping of non-rectangular windows, unlike Cohen Sutherland or Nicholl Le Nicholl. Repeated clipping is also removed that is needed in Cohen Sutherland.

Step1: Normal for each edge is calculated

Step2: Clipping line vector is calculated.

Step3: Dot product between the normal of the edge and the difference between one selected endpoint of one vertex per edge of the clipping line is calculated for all edges.

Step4: Dot product between the normal of edge for all edges and the vector of the clipping line is calculated.

Step5: Then we divide the former dot product by the latter dot product and multiplied by -1. This is 't'.

Step6: The values of 't' are classified as exiting or entering (from all the edges) by observing their denominators (latter dot product).

Step7: To calculate the coordinates, one value of 't' is chosen and put into the parametric form of a line, from each group.

Step8: If the entering 't' value is greater than the exiting 't' value, then the clipping line is rejected.

Cases:

1. **Case 1:** The line is partially inside the clipping window.

   0 <tE<tL< 1

   wheretE is 't' value for entering intersection point

   tL is 't' value for exiting intersection point

2. **Case 2:** The line has one point inside or both sides inside the window or the intersection points are on the end points of the line.

   $0 \leq tE \leq tL \leq 1$

3. **Case 3:** The line is completely outside the window.

   tL<tE

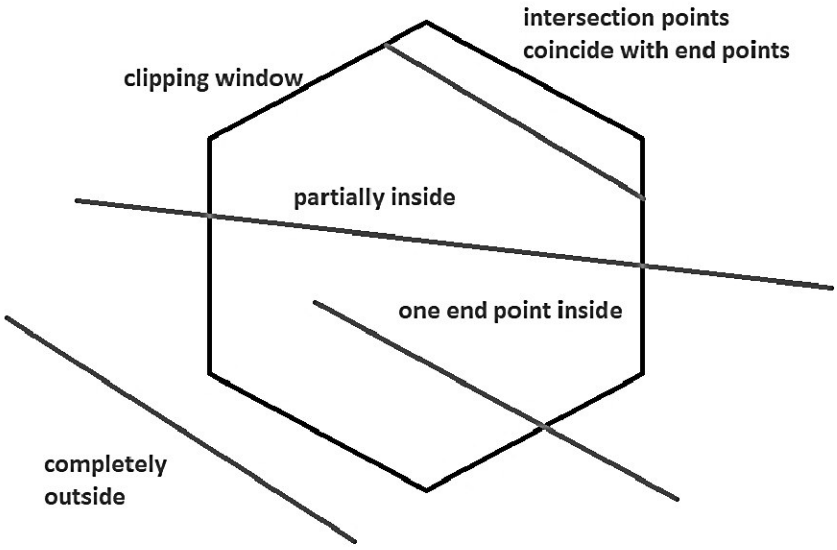


Fig 3.4

3.5 POLYGON CLIPPING

➢ **Polygon Clipping**

Removal of part of an object outside a polygon

Clipping a polygon can result in several disjoint polygons.

A clipping algorithm for polygon must deal with different cases. The case is particularly note worthy in that we separate the concave polygon by clipping it into two polygons. The task of clipping is very complex. We must test each edge of the polygon against each edge of the clip rectangle; we must add new edges, and discard, retain, or divide the existing edge. A single polygon can be clipped into multiple polygons. We need an organized way to deal with all these cases.

➢ **Sutherland-Hodgeman Polygon Clipping**

It is done by processing the boundary of polygon against each window edge or corner.

First step is that the entire polygons is clipped against first edge, then resulting polygon is considered against the second edge, so on for all four edges.

Four possible conditions while processing

1. If the first vertex is inside the window, the second vertex is outside the window then the first vertex is added to output list. The point of intersection of polygon side (edge) and the window boundary is also added to the output list.
2. If both vertices are inside the boundary of window then only second vertex is added to the output list.
3. If the one vertex is inside the boundary window and the other is outside the window then the edge which intersects with the window is added to output list.
4. If both vertices are the outside the boundary window, then both are discarded.

There are two sub-problems that need to be discussed before implementing the:

1. **To decide whether a point lies inside or outside the clipper polygon:**

   If the vertices of the clipper polygon are given in clockwise order then all the points lying on the right side of the clipper edges are inside that polygon. This can be calculated using:

   Given that the line starts from (x_1, y_1) and ends at (x_2, y_2)

   $$P = (x_2 - x_1)(y - y_1) - (y_2 - y_1)(x - x_1)$$

   if P<0, the point is on the right side of the line
   P=0, the point is on the line
   P>0, the point is on the left side of the line

2. **To find the point of intersection of an edge with the clip boundary:**

   If two points of each line(1,2 & 3,4) are known, then their point of intersection can be calculated using the formula:

$$(P_x, P_y) = \left(\frac{(x_1y_2 - y_1x_2)(x_3 - x_4) - (x_1 - x_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}, \frac{(x_1y_2 - y_1x_2)(y_3 - y_4) - (y_1 - y_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)} \right)$$

- **Windowing Transformation**

The process of selecting and enlarging a portion of a drawing is known as windowing. The mapping of a coordinate scene to device coordinates is known as a Windowing transformation etc.

Transformations are applied to objects to transform from one coordinate system to another coordinate system or within the same coordinate system. In Figure 3.5, a view of the different transformation phases in OpenGL, is shown. In modelling coordinates, an object is transformed to world coordinates, also called as object coordinates. In world coordinates, another system of coordinates is defined which is known as the viewing coordinates, and is based on the concept of synthetic camera. A camera position and orientation in the world is defined by these coordinates. Then we compute the coordinates of objects in the viewing coordinate system from world coordinate system. This can be done using the Model View transformation stage in OpenGL

Windowing is a process which transforms the co-ordinates from one space to another. It is used when we need to scale and transform the view of a program. For example: when an image is zoomed in, the data of the original image is transformed to fill the screen. This is done through the window.

In early computer graphics, the instructions were only based on screen. But now a days computer graphics instructions work in different domains such as world domains and modelling.

It is done in several steps:

Step1: The scene is constructed in world coordinate using the attributes and output primitives.

Step2: For some particular orientation, we need to set up a 2-dimensional system of viewing coordinate in the window coordinate plane and then we define a window in this viewing system.

Step3: When the frame of viewing is established, then we transform the description in world coordinates to viewing coordinates.

Step4: Viewport is defined in normalized coordinates (ranging from 0 to 1) so the description of the scene in viewing coordinate system is mapped to normalized coordinates.

Step5: As a final step all parts of the picture that are outside the viewport are cipped, and the content is transferred to the device coordinates.

Window to Viewport Transformation :

It is the process of transforming a 2-dimensional world-coordinate object to device coordinates. The objects that are inside the clipping window are mapped to the viewport. Viewport is the area on the screen where those world coordinates are mapped which we need to display.

A Viewport is a section of the screen where the image in the window of the world coordinate system will be drawn. We need a coordinate transformation to display the image in the viewport, which is in the window. In the viewport, the screen coordinate system is used so we need to transformation the world coordinate system to the screen coordinate system.

When we place a window on the world, we can see only some certain objects and the parts of objects. The points and lines that lie outside the selected

window are cut from the view. The process of cutting the outside points or lines is known as Clipping. In clipping, we have to examine each and every line to determine if it lies completely inside the selected window, completely outside the selected window, or crosses the boundary of the window. If the line is inside the window, it is displayed. If the line is outside the selected window, the lines and points are discarded. If the line crosses a boundary, then we have to determine the point of intersection and we only display the part that is inside the window.

Terminology:

- **World coordinate:** It is the Cartesian coordinate w.r.t which we define the diagram, like Xwmin, Xwmax, Ywmin, Ywmax.
- **Device Coordinate :**It is the screen coordinate where the objects is to be displayed, like Xvmin, Xvmax, Yvmin, Yvmax.
- **Window :**It is the area on world coordinates system selected to display.
- **ViewPort :**It is the area on the device coordinate where graphics is to be displayed.

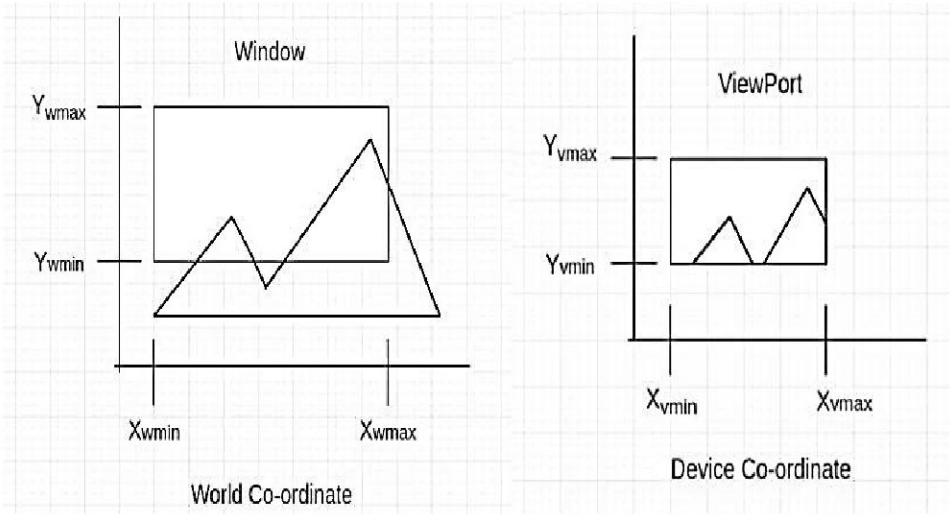


Fig 3.5

Basically, the window is an area in object space. It encloses the object. After the user selects this, space is mapped on the whole area of the viewport. Almost all 2D and 3D graphics packages provide means of defining viewport size on the screen. It is possible to determine many viewports on different areas of display and view the same object in a different angle in each viewport.

The size of the window is (0, 0) coordinate which is a bottom-left corner and toward right side until window encloses the desired area. Once the window is defined data outside the window is clipped before representing to screen coordinates. This process reduces the amount of data displaying signals.

Viewing transformation in several steps :

- We first construct the scene in the world coordinate system using the attributes and output primitives.

- In order to get a particular orientation, we need to set up a 2-dimensional viewing coordinates system in the window coordinate plane and define a window in that system.
- Once we established the viewing frame, then we transform description in world coordinates system to viewing coordinates system.
- Viewport is defined in normalized coordinates (ranging from 0 to 1) so the description of the scene in viewing coordinate system is mapped to normalized coordinates.
- As a final step all parts of the picture that are outside the viewport are cipped, and the content is transferred to the device coordinates.

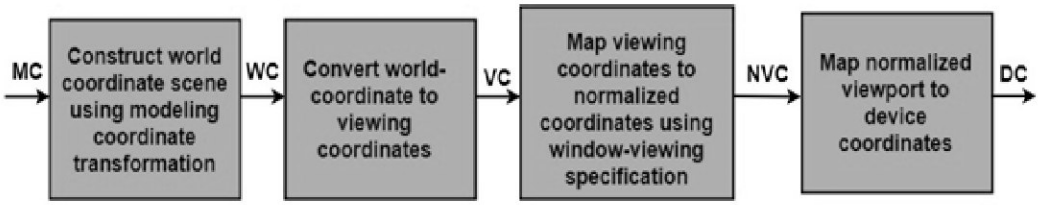


Fig: The two-dimensional viewing-transformation.

Fig 3.6

By changing viewport's position: We can view the objects at different locations on output device display area as shown in fig:3.7

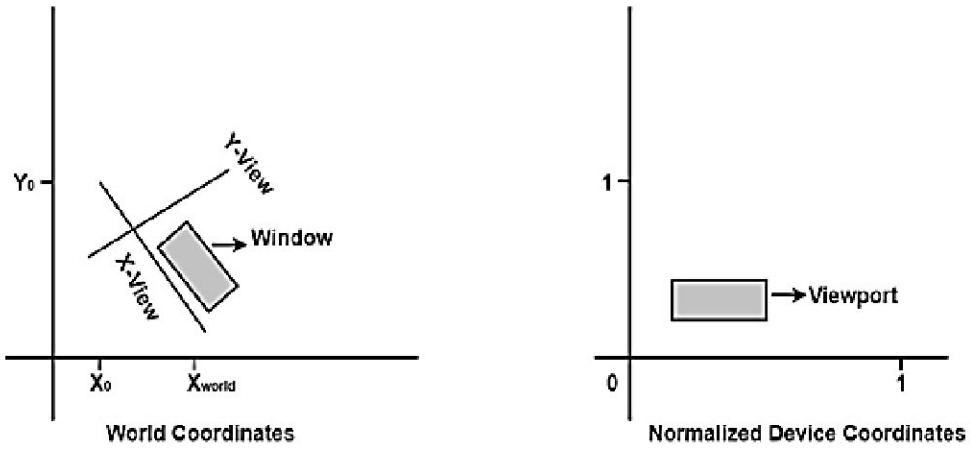


Fig:Setting up a rotated world window and corresponding normalized coordinate viewport.

Fig 3.7

3.6 SUMMARY

In this unit we had covered different topics like point clipping , line clipping , polygon clipping which contains different algorithms like Cyrus-Beck Line Clipping Algorithm, Cohen-Sutherland Line Clippings, for line clipping, Sutherland-Hodgeman Polygon Clipping for polygon clipping, then we had defined the various window to viewport transformation.

We discussed different steps for view transformation, and then one technique by changing the position of the viewport. This technique helps us to find the four major parameters for the transformation which are world coordinate, device coordinate, window and viewport.

3.7 TECHNICAL QUESTIONS

- Derive expression for windows to view port transformations.
- Explain the utility of Clipping algorithms with suitable example.
- What do you mean by polygon clipping? Explain Sutherland _ Hodgeman Polygon clipping with an example.
- Distinguish between panning and zooming.

1. What is the primary use of clipping in computer graphics?
   a) adding graphics
   b) removing objects and lines
   c) zooming
   d) copying
2. Which vertex of the polygon is clipped first in polygon clipping ?
   a) top right
   b) bottom right
   c) bottom left
   d) top left
3. In line clipping, the portion of line which is ____ of window is cut and the portion that is ____the window is kept :
   a) outside, inside
   b) inside, outside
   c) exact copy, different
   d) different, an exact copy
4. Cohen Sutherland clipping algorithm computes _____ number of intersections than NLN line clipping :
   a) More
   b) Less
   c) Same
   d) can't be predicted
5. The area around the clipping window is divided into a number of different :
   a) Pixels
   b) Squares
   c) Areas
   d) Lines

6. The region against which an object is to be clipped is called?
   a) Clipping Window
   b) Drawing Window
   c) Image Window
   d) Both b & c
7. A process of changing the position of an object in a straight line path from one coordinate location to another is called?
   a) Translation
   b) Rotation
   c) Motion
   d) Both b & c
8. The rectangle portion of the interface window that defines where the image will actually appear are called
   a) View port
   b) Transformation viewing
   c) Clipping window
   d) Screen coordinate system
9. Coordinates of window are knows as :
   a) Screen coordinates
   b) World coordinates
   c) Device coordinates
   d) Cartesian coordinates
10. Coordinates of viewport are known as :
   a) World coordinates
   b) Polar coordinates
   c) Screen coordinates
   d) Cartesian coordinates

Answers :

1. B
2. D
3. A
4. A
5. C
6. A
7. A
8. A
9. B
10. C

Uttar Pradesh Rajarshi Tandon
Open University

Master of Computer Science

MCS-116
Computer Graphics

BLOCK
2

TRANSFORMATIONS

UNIT-4

2-D and 3-D Transformations

UNIT-5

Viewing Transformation

Course Design Committee

Prof. Ashutosh Gupta
Director (In-charge)
School of Computer and Information Science, UPRTOU Prayagraj

Prof. Suneeta Agarwal
Department of CSE.
Motilal Nehru National Institute of Technology, Allahabad, Prayagraj

Dr. Upendra Nath Tripathi **Author**
Associate Professor
Department of Computer Science
Deen Dayal Upadhyaya Gorakhpur University, Gorakhpur

Dr. Ashish Khare
Associate Professor
Dept. of Computer Science, Allahabad University

Ms. Marisha
Assistant Professor (Computer Science)
School of Science, UPRTOU Prayagraj

Mr. Manoj Kumar Balwant
Assistant Professor (Computer Science)
School of Science, UPRTOU Prayagraj

Course Preparation Committee

Dr. Divya Kumar **Author**
Assistant Professor
Department of Computer Science & Engineering
Motilal Nehru National Institute of Technology Prayagraj

Prof. Abhay Saxena **Editor**
Dean, School of TCM
Dev Sanskriti Vishwavidyalaya, Haridwar -Uttarakhand

Prof. Ashutosh Gupta **Director (In-Charge)**
School of Computer & Information Sciences
UPRTOU Prayagraj

Mr. Manoj Kumar Balwant **Coordinator**
Assistant Professor (computer science)
School of Sciences, UPRTOU Prayagraj

©UPRTOU, Prayagraj - 2020
ISBN :

UNIT-4 2D AND 3D TRANSFORMATION

Structure :

4.1 INTRODUCTION

Pixel is a basic element of digital graphics. Pixels are combined to form a complete image, video, text or any visible thing on a computer display. 2D Transformation means a change in either position or orientation or size or shape of graphics objects like line, circle, arc, ellipse, rectangle, polygon, polylines etc. There are two specific types of transformations; modelling and viewing transformation. A modelling transformation is used to change the actual geometry of the object. Whereas viewing transformation is used to alter the displayed image.

The coordinates of an object can be represented in point matrix. There are two types of matrix format to represent points of an object: row matrix and column matrix.

For a 2D object with point (x, y)

Row matrix format is [x y], 1-row, 2-column.

Column matrix format is $\begin{bmatrix} x \\ y \end{bmatrix}$, 2-row, 1-column.

For a 3D object with point (x, y, z)

Row matrix format is [x y z], 1-row, 3-column

Column matric format is $\begin{bmatrix} x \\ y \\ z \end{bmatrix}$, 3-row, 1-column

For example, points (1, 2), (3, 4), (0,3) in row matrix format $\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 0 & 3 \end{bmatrix}$

and in column matrix format $\begin{bmatrix} 1 & 3 & 0 \\ 2 & 4 & 3 \end{bmatrix}$

Transformation can be expressed in matrix form as

$$[P^*]=[P][T]$$

where [P*] is new point matrix, [P] is old points matrix, and [T] is transformation matrix.

4.2 OBJECTIVE

After the end of this unit, you should be able to understand:

- How to translate an image.
- How to rotate an image.
- How to scale an image.
- Shearing of an image.
- Composite Transformation.
- Homogeneous Coordinate System.
- 3D Transformations.

4.3 BASIC TRANSFORMATION: TRANSLATION, ROTATION, SCALING, SHEAR

4.3.1 TRANSLATION

Translation transformation means an object moves to a different position on the screen such that every point of the object experiences the same displacement. Translation can be performed for points, lines and objects. In fig 4.1, an object A with coordinate (x, y) shifted right side in x-direction then new coordinate of object A' after translation is (x', y').

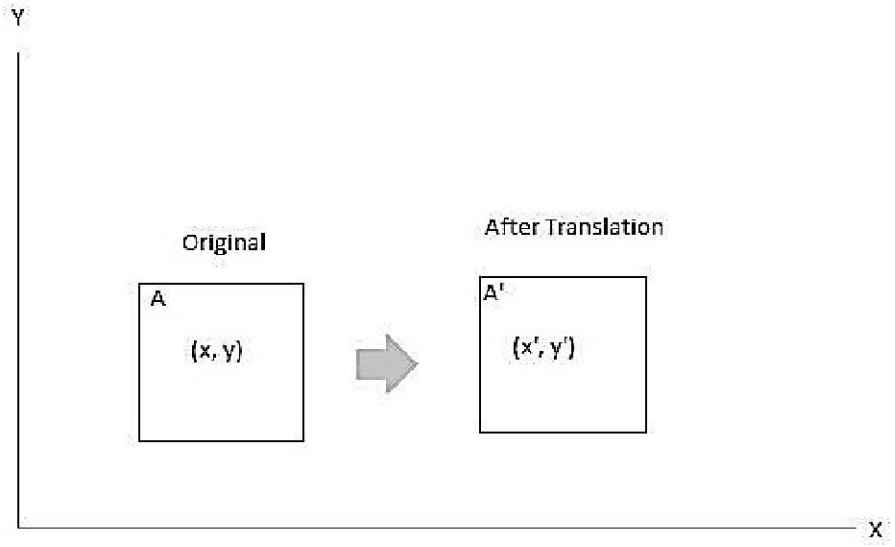


Fig. 4.1 Translation in x-direction

A is a square with 2D coordinates (x, y) = {(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)} after shifting A toward right direction, the new coordinates are (x', y') = {(x'$_1$, y'$_1$), (x'$_2$, y'$_2$), (x'$_3$, y'$_3$), (x'$_4$, y'$_4$)}.

Algebraically $x' = x + \Delta x$ (same as [P*]=[P]+[T]

$$y' = y + \Delta y$$

Δx and Δy is difference of coordinates in respective axis, also called translation vector or Shift vector.

For example-

Translation in x-direction-

In fig.4.1, an object A (x,y) moves in x-direction after translation new coordinates of object is A' (x', y'). In this type of translation, only x-coordinates changed whereas y-coordinates remain the same.

If coordinates of original square are A={(2,3), (2,6), (5,3), (5,6)}and square shifted right side by 5-unit on x-axis.

Point matrix for A is $\begin{bmatrix} 2 & 2 & 5 & 5 \\ 3 & 6 & 3 & 6 \end{bmatrix}$

Square shifted 5-unit right side, so translation vector matrix ΔA is $\begin{bmatrix} 5 & 5 & 5 & 5 \\ 0 & 0 & 0 & 0 \end{bmatrix}$

$$A' = A + \Delta A$$

$$\begin{bmatrix} 7 & 7 & 10 & 10 \\ 3 & 6 & 3 & 6 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 5 & 5 \\ 3 & 6 & 3 & 6 \end{bmatrix} + \begin{bmatrix} 5 & 5 & 5 & 5 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

So new coordinates of translated square are A'= {(7,3), (7,6), (10,3), (10,6)}.

Translation in y-direction

In fig.4.2, an object A (x,y) moves in y-direction after translation new coordinates of object is A' (x', y'). In this type of translation, only y-coordinates changed whereas x-coordinates remain the same.

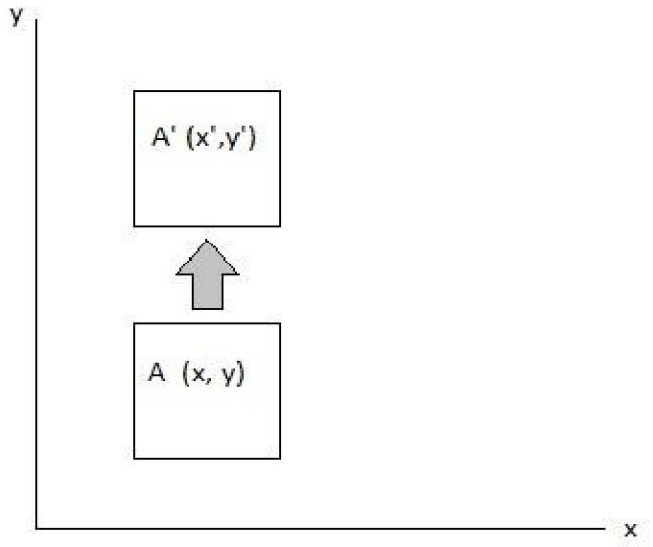


Fig. 4.2 Translation in y-direction

If coordinates of original square are A={(2,3), (2,6), (5,3), (5,6)}and square shifted upward side by 4-unit on y-axis.

Point matrix for A is $\begin{bmatrix} 2 & 2 & 5 & 5 \\ 3 & 6 & 3 & 6 \end{bmatrix}$

Square shifted 4-unit right side, so translation vector matrix ΔA is $\begin{bmatrix} 0 & 0 & 0 & 0 \\ 4 & 4 & 4 & 4 \end{bmatrix}$

$$A' = A + \Delta A$$

$$\begin{bmatrix} 2 & 2 & 5 & 5 \\ 7 & 10 & 7 & 10 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 5 & 5 \\ 3 & 6 & 3 & 6 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 4 & 4 & 4 & 4 \end{bmatrix}$$

So new coordinates of translated square are A'= {(2,7), (2,10), (5,7), (5,10)}.

Translation in both directions -

In fig.4.3, an object A (x,y) moves in both y-direction and x-direction after translation new coordinates of object is A' (x', y'). In this type of translation, both y-coordinates and x-coordinate changed. Δx shows the displacement in x-direction of a point and Δy shows the displacement in y-direction.

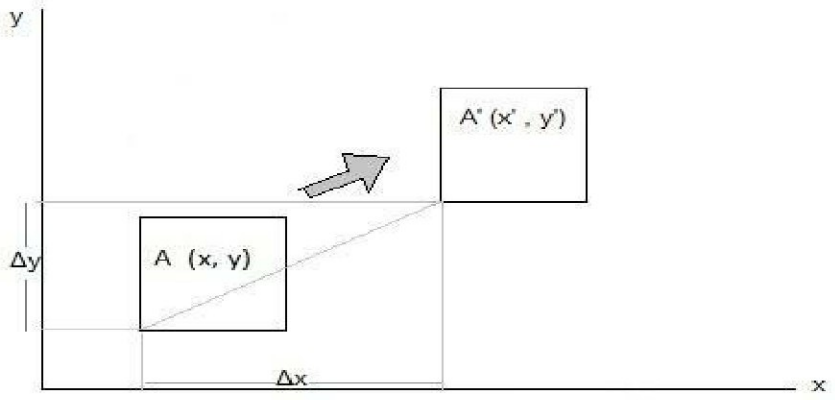


Fig. 4.3 Translation in both directions

If coordinates of original square are A={(2,3), (2,6), (5,3), (5,6)}and square shifted 3-unit right side on x-axis and upward side by 4-unit on y-axis.

Point matrix for A is $\begin{bmatrix} 2 & 2 & 5 & 5 \\ 3 & 6 & 3 & 6 \end{bmatrix}$

Square shifted 3-unit & 4-unit, so translation vector matrix ΔA is $\begin{bmatrix} 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 \end{bmatrix}$

$$A' = A + \Delta A$$

$$\begin{bmatrix} 5 & 5 & 8 & 8 \\ 7 & 10 & 7 & 10 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 5 & 5 \\ 3 & 6 & 3 & 6 \end{bmatrix} + \begin{bmatrix} 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 \end{bmatrix}$$

So new coordinates of translated square are A'= {(5,7), (5,10), (8,7), (8,10)}.

4.3.2 ROTATION

In rotation transformation, an object rotates about any point. Point of rotation can be origin of plane or any arbitrary pivot point.

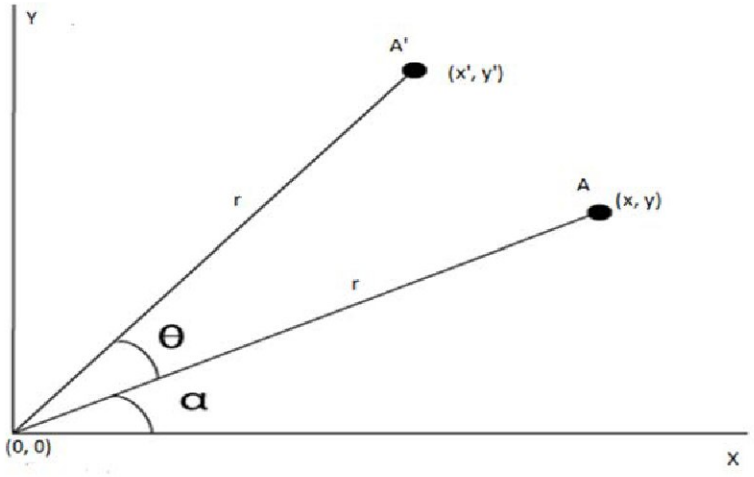


Fig. 4.4 Rotation

Rotation about Origin (0,0)-

In fig 4.4 point A rotates θ angle anti-clockwise about origin. Coordinate of point A is (x,y) and after rotation new coordinate of A' is (x',y').

$$A' = A * \Delta A$$

If the distance of point A from origin is r, then distance of point A' will also be r.

So,
$$x = r\cos\alpha$$
$$y = r\sin\alpha$$

and,
$$x' = r\cos(\alpha+\theta)$$
$$y' = r\sin(\alpha+\theta)$$

Now in matrix representation;

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

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

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

For example; if initially the coordinate of point is (x,y)=(4,4), and point rotates about origin by θ=30°.

So, new coordinate of point

$$x' = 4*\cos30° - 4*\sin30°$$

$$y' = 4*\sin30° + 4*\cos30°$$

Hence (x', y')= (1.46, 5.46).

Rotation about an arbitrary point (x_c, y_c)-

If we want to rotate an object or point about an arbitrary point, first of all, we translate the point about which we want to rotate to the origin. Then rotate point or object about the origin, and at the end, we again translate it to the original place. We get rotation about an arbitrary point.

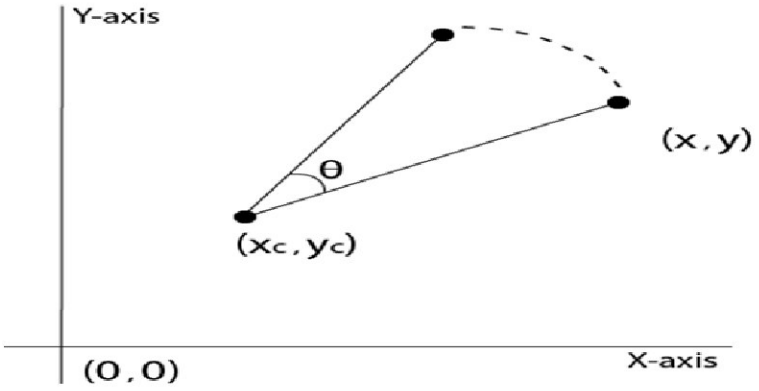


Fig. 4.5 Rotation about (x_c, y_c)

In Fig. 4.5 point (x,y) rotates about point (x_c, y_c) by θ° angle in 2D space.

For example-

If a point (4,5) want to rotate about (2,2) by 45° anti-clockwise .

$(x_c,y_c) = (2,2)$

$(x,y) = (4,5)$

$\theta = 45°$

Step 1: Translate (x_c,y_c) to the origin (0,0)

So the translation vector is $\Delta x = -2$ and $\Delta y = -2$

Translate (x,y) by $\Delta x = -2$ and $\Delta y = -2$

So, x'=4-2=2 and y'=5-2=3, (x',y')=(2,3)

Step 2: Now rotate new translated point about the origin by 45°, new coordinate after

rotation about origin

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

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

$$x'' = \frac{-1}{\sqrt{2}} \; and \; y'' = \frac{5}{\sqrt{2}}$$

Step 3: Finally, rotated coordinate is translated back by translation vector by $\Delta x = 2$

and $\Delta y = 2$.

After translation back new coordinates of rotated point about (2,2) is

$x_f = x''+2$ and $y_f = y''+2$

So the final coordinate of point (4,5) after rotation about (2,2) by 45° anti-clockwise , $(x_f, y_f) = \left(\frac{-1}{\sqrt{2}} + 2 \, , \, \frac{5}{\sqrt{2}} + 2\right)$

4.3.3 SCALING

Scaling is a transformation that changes the size or shape of an object. Scaling can be with respect to the origin or with respect to the any arbitrary point.

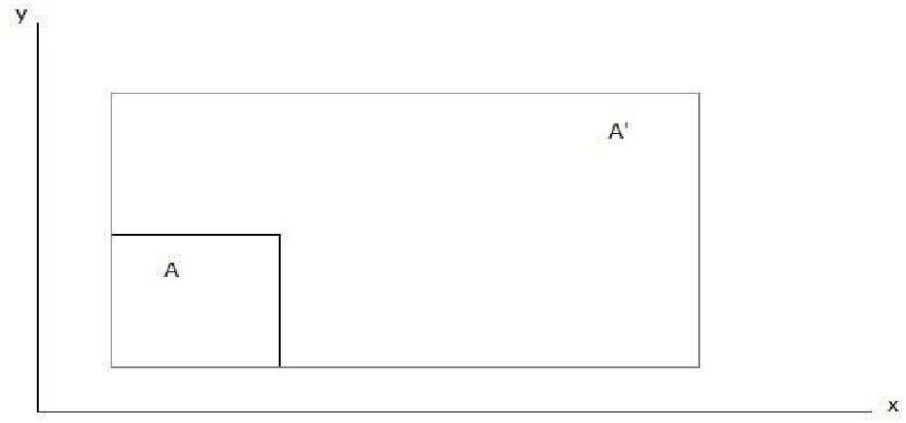


Fig. 4.6 Scaling of an object

In fig.4.6 after scaling object A, it become object A' with increased shape. If scaling factor > 1, then the object size is increased. If scaling factor < 1, then the object size is reduced.

If the initial coordinate of object is (x, y)

Scaling factor for x-axis and y-axis are S_x and S_y respectively.

Coordinate of object after scaling is (x',y')

So, $x' = x * S_x$

$y' = y * S_y$

Matrix representation is: [P*] = [P][T]

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} \begin{bmatrix} Sx & 0 \\ 0 & Sy \end{bmatrix}$$

For example; if initial coordinates of an square are (x,y) = {(0,0), (0,4), (4,0), (4,4)}. Scaling factor for x-axis is S_x=3 and for y-axis is S_y=2 about origin.

Calculate scaling of each coordinate individually;

For (0,0) x'= 0*3 = 0 and y'= 0*2=0

For (0,4) x'= 0*3 = 0 and y'= 4*2=8

For (4,0) x'= 4*3 = 12 and y'= 0*2=0

For (4,4) x'= 4*3 = 12 and y'= 4*2=8

So the new coordinates of scaled object are (x',y') = {(0,0), (0,8), (12,0), (12,8)}.

4.3.4 SHEARING

Shearing is an ideal technique to change the shape of an object in a two dimensional plane. In this transformation, the sliding of layers of object occurs.

Shearing can be done in one direction at a time or in both directions simultaneously.

X-shear:

In fig.4.7, the change is happening in x-direction, y-coordinate remains same. Means upper layer of object A is sliding against lower layer of object A. After shearing, shape of object changed from square to parallelogram.

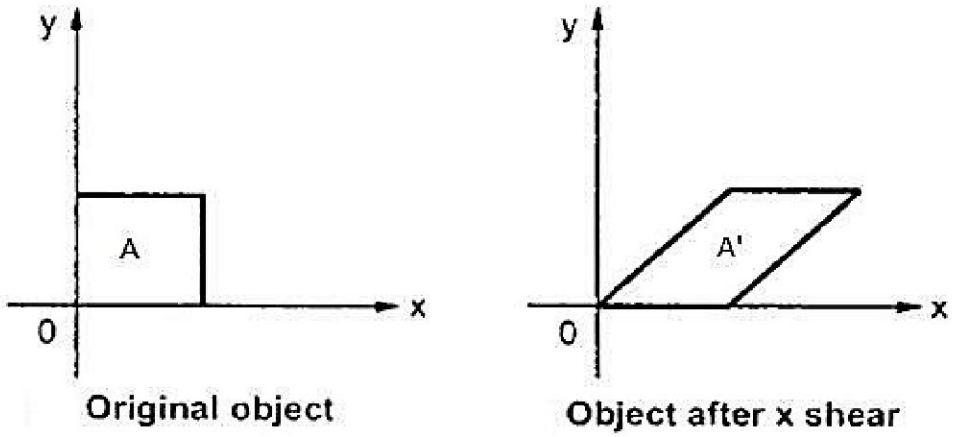


Fig.4.7 shearing of an object in x-direction

Changed coordinate is as follow after shearing

$x' = x + sh_x * y$ where sh_x is shearing factor in x-direction

$y' = y$

in point matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 1 & sh_x \\ 0 & 1 \end{bmatrix}$$

For example; if initial coordinates of an square are (x,y) = {(0, 0), (0, 3), (3, 0), (3, 3)} and x-shear factor is $sh_x=2$.

For (0,0)	$x' = 0 + 2*0 = 0$	and	$y' = 0$
For (0,3)	$x' = 0 + 2*3 = 6$	and	$y' = 3$
For (3,0)	$x' = 3 + 2*0 = 3$	and	$y' = 0$
For (3,3)	$x' = 3 + 2*3 = 9$	and	$y' = 3$

So, after x-shearing new coordinates of sheared square are (x',y')= {(0,0), (6,3), (3,0), (9,3)}.

Y-shear:

In fig.4.8, the change is happening in y-direction, x-coordinate remains same. Means right layer of object A is sliding against left layer of object A. After shearing, shape of object changed from square to parallelogram.

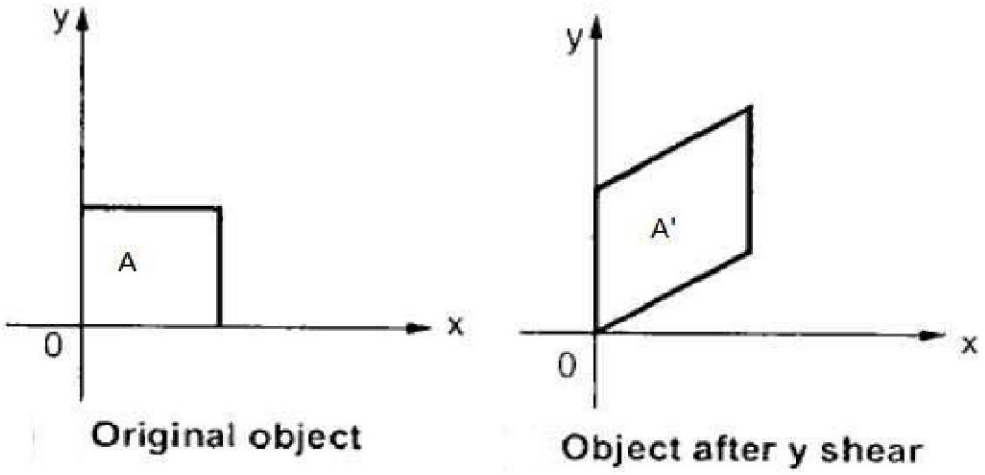


Fig.4.8 shearing of an object in y-direction

Changed coordinate is as follow after shearing

$x' = x$

$y' = y + sh_y * x$ where sh_y is shearing factor in y-direction

in point matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ sh_y & 1 \end{bmatrix}$$

For example; if initial coordinates of an square are (x,y) = {(0, 0), (0, 3), (3, 0), (3, 3)} and y-shear factor is sh_y=2.

For (0,0)	x'= 0	and	y'= 0 + 2*0=0
For (0,3)	x'= 0	and	y'= 3 + 2*0=3
For (3,0)	x'= 3	and	y'= 0 + 2*3=6
For (3,3)	x'= 3	and	y'= 3 + 2*3=9

So, after y-shearing new coordinates of sheared square are (x',y')= {(0,0), (0,3), (3,6), (3,9)}.

4.4 COMPOSITE TRANSFORMATION

The transformations we have studied above are performed individually. If more than one transformation performed sequentially as a single job then it is called composite transformation. More clearly if a change in object needs more than one transformation means need sequence of transformations then these sequences of transformation is called composite transformation.

Case 1: Scaling then Translation

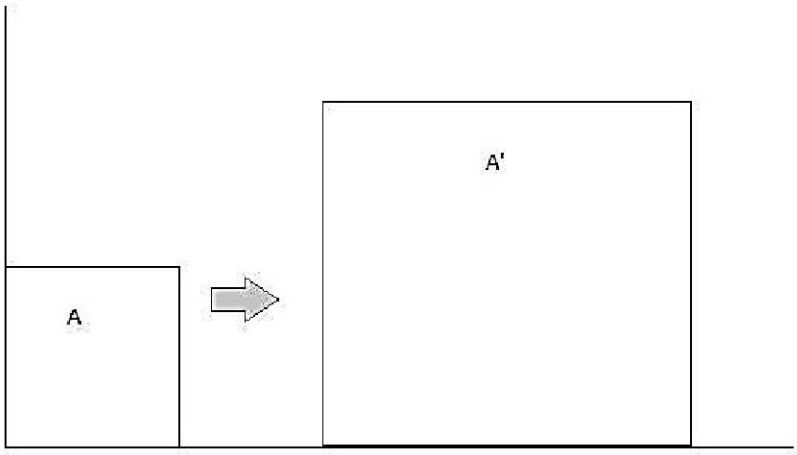


Fig. 4.5 Composite: Scaling then Translation

In fig.4.5, if A want to change in A', it is not possible with a single transformation technique. There is need of two transformation sequences, first scaling the object and then translate it to new coordinate.

Case 2: Scaling then Translation then Rotation

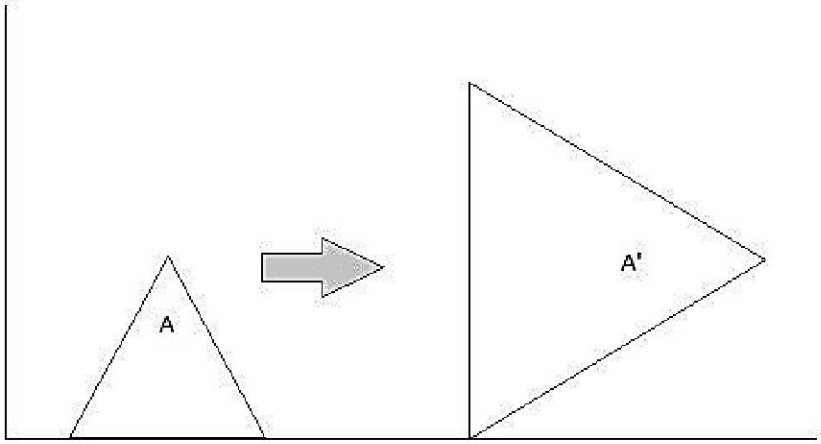


Fig. 4.6 Composite: Scaling then Translation then Rotation

In this case, if A want to change in A', it is not possible with a single transformation technique. There is need of three transformation sequences, first scaling the object then translate it to new coordinate and then rotate it 90° clockwise.

4.5 HOMOGENOUS COORDINATE SYSTEM

To combine these three transformations into a single transformation, homogeneous coordinates are used. In homogeneous coordinate system, two-dimensional coordinate positions (x, y) are represented by triple-coordinates. Homogeneous coordinates are generally used in design and construction applications. Here we perform translations, rotations, scaling to fit the picture into proper position. In homogeneous coordinates, a third coordinate is added to a point. Instead of being represented by an ordered pair of two numbers (x,y), each

point is represented by a triple (x,y,W). Two sets of homogeneous coordinates (x,y,W) and (x',y',W') represent the same point if one is a multiple of the other. Homogeneous coordinate for:

Translation: $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \Delta x & \Delta y & 1 \end{bmatrix}$ or $\begin{bmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{bmatrix}$

Scaling: $\begin{bmatrix} S_x & 0 & 0 \\ 0 & S_x & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Rotation: $\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$ or $\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Shearing: x-shear $\begin{bmatrix} 1 & 0 & 0 \\ sh_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

y-shear $\begin{bmatrix} 1 & sh_y & 0 \\ sh_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

4.6 3D TRANSFORMATIONS

In this, object or a point is considered in three-dimensional space. Transformation of a object can be about any of three dimensions. The location of objects relative to others can be easily expressed in three-dimensional space. 3D transformations that deal with scaling, translation and rotation are a simple extension of 2D transformation.

4.6.1 3D TRANSLATION

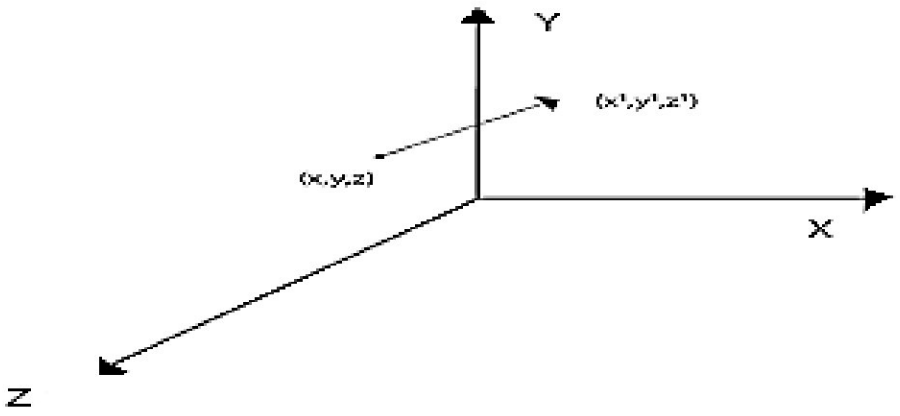


Fig. 4.7 3D Translation

Initial coordinate of object is (x,y,z)

Translation factor in each three directions is t_x, t_y, t_z.

New coordinate of object after translation is (x', y', z')

So, new coordinate for each point of object

$$x' = x + t_x$$

$$y' = y + t_y$$

$$z' = z + t_z$$

In matrix form,

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_y \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4.6.2 3D ROTATION

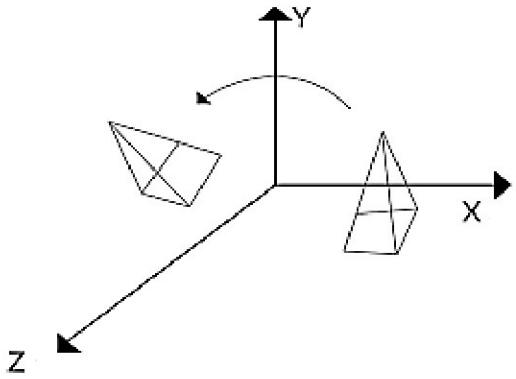


Fig. 4.8 3D Rotation

Initial coordinate of object is (x,y,z)

Initial angle of object with respect to origin is α

Object is rotated by angle θ

After rotation new coordinate of object (x',y',z')

There are possibly three types of rotations about x-axis, about y-axis and, about z-axis.

Rotation about x-axis-

So, new coordinate for each point of object

$$x' = x$$

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

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

In matrix form (in homogenous coordinates)

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

Rotation about y-axis-

So, new coordinate for each point of object

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

$$y' = y$$

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

In matrix form (in homogenous coordinates)

$$[x'\ y'\ z'\ 1\ ] = [x\ y\ z\ 1\ ] [cos\theta\ 0\ 0\ 1\ sin\theta\ 0\ 0\ 0\ -sin\theta\ 0\ 0\ 0\ cos\theta\ 0\ 0\ 1\ ]$$

Rotation about z-axis-

So, new coordinate for each point of object

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

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

$$z' = z$$

In matrix form (in homogenous coordinates)

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

4.6.3 3D SCALING

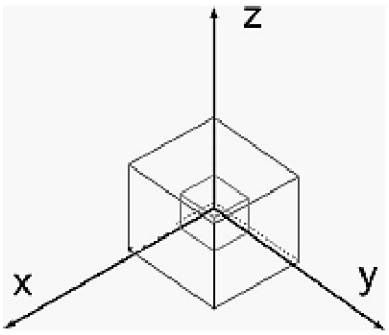


Fig. 4.9 3D Scaling

Initial coordinate of object is (x,y,z)

Scaling factors about x-axis, y-axis, z-axis are S_x, S_y, S_z respectively.

New coordinate of object after rotation (x', y',z')

So, new coordinate for each point of object

$$x' = x * S_x$$

$$y' = y * S_y$$

$$z' = z * S_z$$

In matrix form (in homogenous coordinates)

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4.7 SUMMARY

In this unit 2D transformations are shown; translation, scaling, rotation, and shearing. 2D transformation is done in two-dimensional plane generally XY-plane. When 3rd direction added to 2D-plane as Z-direction, XY-plane becomes 3D space. Real life objects are present in 3D space, so 3D transformation is more practical than 2D transformation. In 3D transformation, there are also translation, scaling, rotation, and shearing. Translation is to move an object in 2D/3D space w.r.t. origin or any arbitrary point. Rotation is to rotate an object in 2D/3D space w.r.t. origin or and arbitrary point by angle θ. Scaling is used to change the size of an object without changing its structure. In shearing, layer slides against each other and change the structure of object.

Homogeneous coordinates provide a method to perform certain standard operations on points in Euclidean space by means of matrix multiplications. As we shall see, they provide a great deal more, but let's first review what we know up to this point.

4.8 TECHNICAL QUESTION

1. Why do we need transformation in computer graphics?
2. Write down the difference between 2D and 3D transformation.
3. Design a problem for composite transformation and explain each transformation steps in detail.
4. What is the need to use Homogenous coordinates system. Point out problems if homogeneous coordinates are not used.
5. A square with given coordinates {(1,2),(4,2),(1,5),(4,5)} compute the new coordinates after composite transformation a) Translation in +x direction by 4 unit and then b) Rotation of 45° anti-clockwise about origin.

6. If initial coordinates of an square are (x,y) = {(0,0), (0,4), (4,0), (4,4)}. Scaling factor for x-axis is Sx=3 and for y-axis is Sy=2 about origin. Calculate the new coordinates of given square.

7. A pyramid is in 3D space with coordinates {(2,2,0),(2,4,0),(4,2,0),(4,4,0),(3,3,2)}. Perform 3D rotation of 30° about x-axis, y-axis, and z-axis individually and find new coordinates.

8. A cube is in 3D space with coordinates {(2,2,2), (2,4,2), (4,2,2), (4,4,2), (2,2,4), (2,4,4), (4,2,4), (4,4,4)}. Perform 3D scaling; 2 unit in x-axis, 2 units in y-axis, and 2 units in z-axis. Write down new coordinates of cube and represent in homogenous coordinates.

9. Derive the homogenous coordinates for 2D translation and 2D shearing.

10. Write down differences between scaling and shearing transformation with proper example.

UNIT-5 VIEWING TRANSFORMATION

Structure :

5.1 INTRODUCTION

Viewing transformation also called viewport transformation. Viewpoint transformation is a convenient method for generating a pictorial view of a 3D object. It is based on a vector defines by the line of sight. This vector extends from the view site to the viewpoint. The new view is produced by transforming the line of sight vector so that it is outwardly normal to the viewing surface. In other words if the line of sight vector is part of the object, it would appear as a point after the transformation.

The basic steps of the viewpoint transformation are:

a. Translate the view site to the origin.

b. Rotate the line of sight vector so that it corresponds to the z-axis.

c. Translate the view site back to its original position.

3D objects which are defined and manipulated using actual physical units of measurement in a 3D space, has to be transformed at one stage from a 3D representation to a 2D representation. Because finally the image is viewed on a 2D plane of the display device. Such 3D-to-2D transformation is called projection. 2D projected images are formed by the intersection of lines called projectors with a plane called the projection plane. Projectors are lines from an arbitrary point called the centre of projection through each point in an object.

The major two categories of projection are:

a. Parallel projection

b. Perspective projection

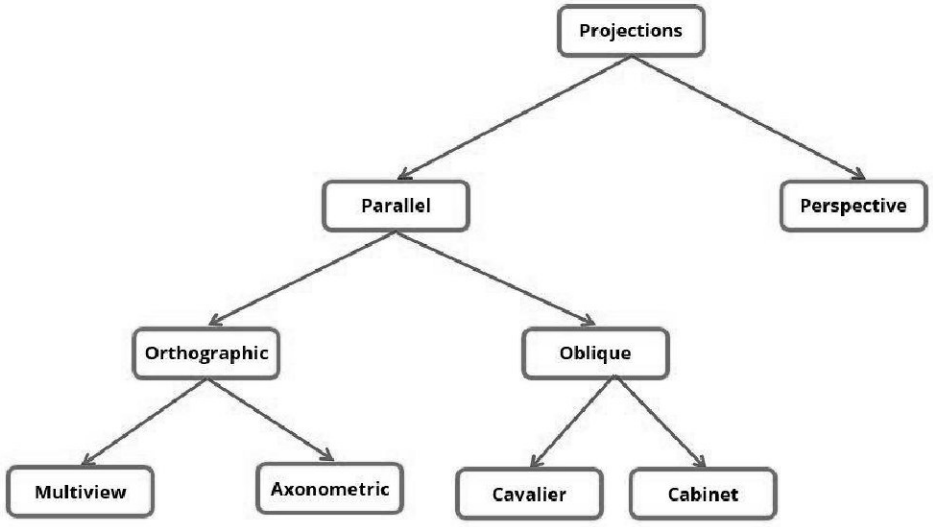


Fig. 5.1 Projection

5.2 OBJECTIVE

After the end of this unit, you should be able to understand:

- Different types of projection like :
- parallel projection,
- orthographic projections, etc.

5.3 PARALLEL PROJECTION

When the centre of projection is situated at infinite distance such that the projectors are parallel to each other, this type of projection is called parallel projection. There are two types of parallel projection:

a. Orthographic Projection

b. Oblique Projection

Characteristics of Parallel Projection:-

1. Parallel projection can give the accurate view of object.
2. Parallel projection represents the object in a different way like telescope.
3. Parallel projection does not form realistic view of object.
4. In parallel projection, the distance of the object from the centre of projection is infinite.
5. Projector in parallel projection is parallel.
6. Parallel projection can preserve the relative proportion of an object.
7. The lines of parallel projection are parallel.

5.4 ORTHOGRAPHIC PROJECTION

The direction of projection is perpendicular to the plane of projection. In fig.5.2, when the direction of projection is parallel to any of the principal axes this produces front, top and side views of the object, also called multi-view drawing. If projection lines are parallel with x-axis then object is projected on z-y plane in 2D format with x=0. In the similar way, If projection lines are parallel with y-axis then object is projected on z-x plane in 2D format with y=0 and If projection lines are parallel with z-axis then object is projected on x-y plane in 2D format with z=0.

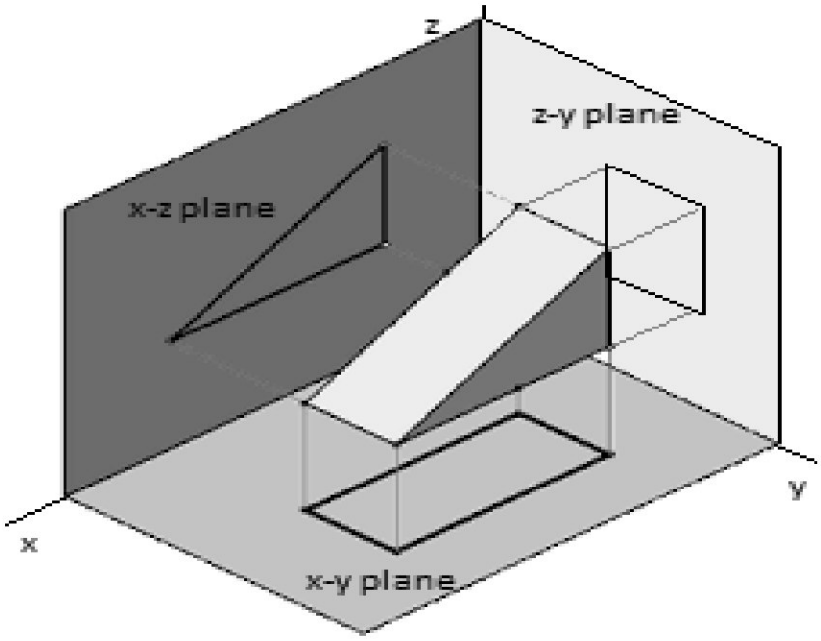


Fig. 5.2 Orthographic projection

If (x', y', z') are assumed to be the coordinates of the projected point. The transformation matrices are:

For projection onto the x-y plane $\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$

For projection onto the x-z plane $\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$

For projection onto the x-y plane $\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$

A single orthographic projection does not provide sufficient information to visually and practically reconstruct the shape of an object. Thus multiple orthographic projections are needed; in fig.5.3 there are 6-views of an object: top view, bottom view, front view, rear view, right view and, left view.

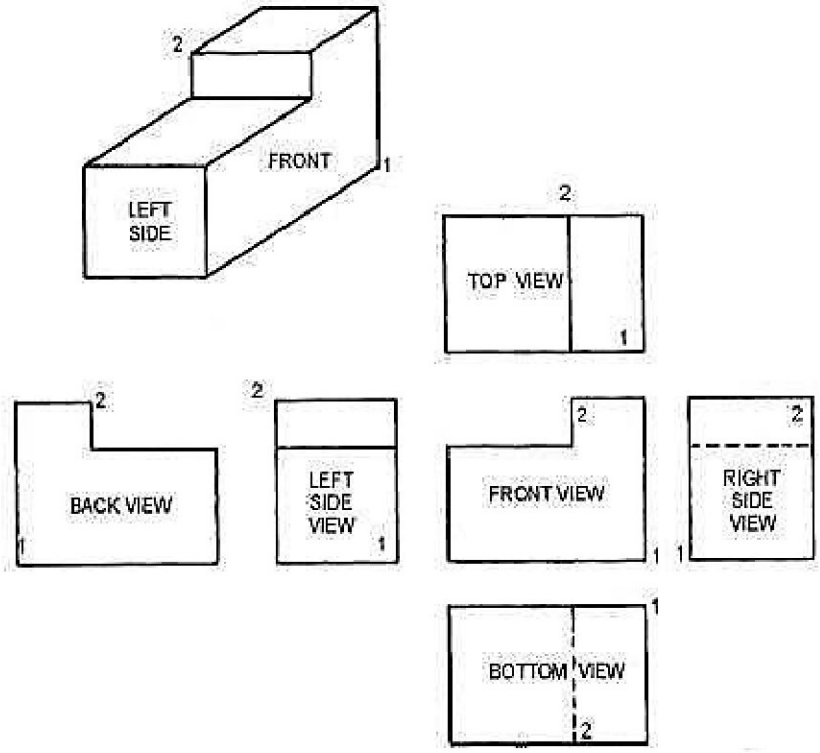


Fig 5.3 6-views orthographic projection

Example: Show all the six views of a given object shown in following Fig.5.4. The vertices of the object are A(4,0,0), B(4,4,0), C(4,4,8), D(4, 0, 4), E (0,0,0), F(0,4,0), G(0,4,8), H(0,0,4).

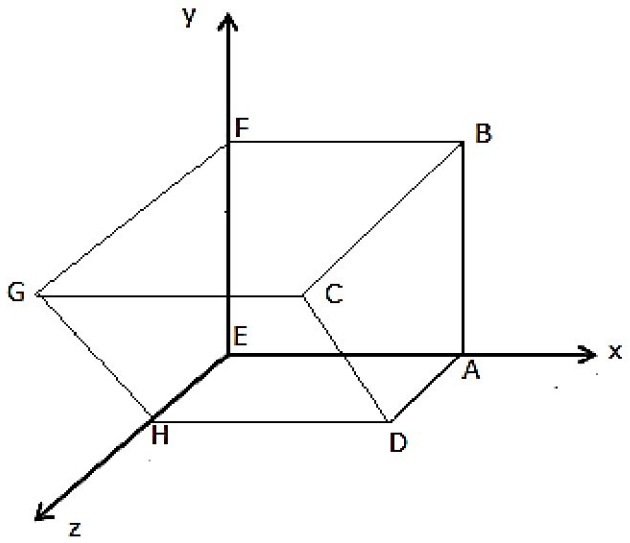


Fig. 5.4 Given object in 3D space

We can represent the given object in terms of Homogeneous-coordinates of its vertices as:

$$V = [ABCDEFGH] = \begin{matrix} A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \end{matrix} \begin{pmatrix} 4 & 0 & 0 & 1 \\ 4 & 4 & 0 & 1 \\ 4 & 4 & 8 & 1 \\ 4 & 0 & 4 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 4 & 0 & 1 \\ 0 & 4 & 8 & 1 \\ 0 & 0 & 4 & 1 \end{pmatrix}$$

If projected coordinates assumed as A', B', C', D', E', F', G', H'.

Front view: In fig.5.5, projection on x-y plane and z=0. So the new coordinate of a given object after projection can be as;

$$\begin{matrix} A' \\ B' \\ C' \\ D' \\ E' \\ F' \\ G' \\ H' \end{matrix} \begin{pmatrix} x'_1 & y'_1 & z'_1 & 1 \\ x'_2 & y'_2 & z'_2 & 1 \\ x'_3 & y'_3 & z'_3 & 1 \\ x'_4 & y'_4 & z'_4 & 1 \\ x'_5 & y'_5 & z'_5 & 1 \\ x'_6 & y'_6 & z'_6 & 1 \\ x'_7 & y'_7 & z'_7 & 1 \\ x'_8 & y'_8 & z'_8 & 1 \end{pmatrix} = \begin{bmatrix} 4 & 0 & 0 & 1 \\ 4 & 4 & 0 & 1 \\ 4 & 4 & 8 & 1 \\ 4 & 0 & 4 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 4 & 0 & 1 \\ 0 & 4 & 8 & 1 \\ 0 & 0 & 4 & 1 \end{bmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{matrix} A' \\ B' \\ C' \\ D' \\ E' \\ F' \\ G' \\ H' \end{matrix} \begin{pmatrix} x'_1 & y'_1 & z'_1 & 1 \\ x'_2 & y'_2 & z'_2 & 1 \\ x'_3 & y'_3 & z'_3 & 1 \\ x'_4 & y'_4 & z'_4 & 1 \\ x'_5 & y'_5 & z'_5 & 1 \\ x'_6 & y'_6 & z'_6 & 1 \\ x'_7 & y'_7 & z'_7 & 1 \\ x'_8 & y'_8 & z'_8 & 1 \end{pmatrix} = \begin{bmatrix} 4 & 0 & 0 & 1 \\ 4 & 4 & 0 & 1 \\ 4 & 4 & 0 & 1 \\ 4 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 4 & 0 & 1 \\ 0 & 4 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

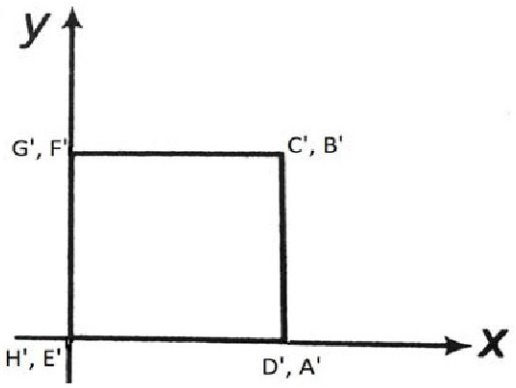


Fig. 5.5 Front view projection of object in x-y plane

Right view: In fig.5.6 projection on y-z plane and x=0. So the new coordinate of a given object after projection can be as;

$$\begin{matrix} A' \\ B' \\ C' \\ D' \\ E' \\ F' \\ G' \\ H' \end{matrix}\begin{pmatrix} x'_1 & y'_1 & z'_1 & 1 \\ x'_2 & y'_2 & z'_2 & 1 \\ x'_3 & y'_3 & z'_3 & 1 \\ x'_4 & y'_4 & z'_4 & 1 \\ x'_5 & y'_5 & z'_5 & 1 \\ x'_6 & y'_6 & z'_6 & 1 \\ x'_7 & y'_7 & z'_7 & 1 \\ x'_8 & y'_8 & z'_8 & 1 \end{pmatrix} = \begin{bmatrix} 4 & 0 & 0 & 1 \\ 4 & 4 & 0 & 1 \\ 4 & 4 & 8 & 1 \\ 4 & 0 & 4 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 4 & 0 & 1 \\ 0 & 4 & 8 & 1 \\ 0 & 0 & 4 & 1 \end{bmatrix} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{matrix} A' \\ B' \\ C' \\ D' \\ E' \\ F' \\ G' \\ H' \end{matrix}\begin{pmatrix} x'_1 & y'_1 & z'_1 & 1 \\ x'_2 & y'_2 & z'_2 & 1 \\ x'_3 & y'_3 & z'_3 & 1 \\ x'_4 & y'_4 & z'_4 & 1 \\ x'_5 & y'_5 & z'_5 & 1 \\ x'_6 & y'_6 & z'_6 & 1 \\ x'_7 & y'_7 & z'_7 & 1 \\ x'_8 & y'_8 & z'_8 & 1 \end{pmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 4 & 0 & 1 \\ 0 & 4 & 8 & 1 \\ 0 & 0 & 4 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 4 & 0 & 1 \\ 0 & 4 & 8 & 1 \\ 0 & 0 & 4 & 1 \end{bmatrix} \backslash$$

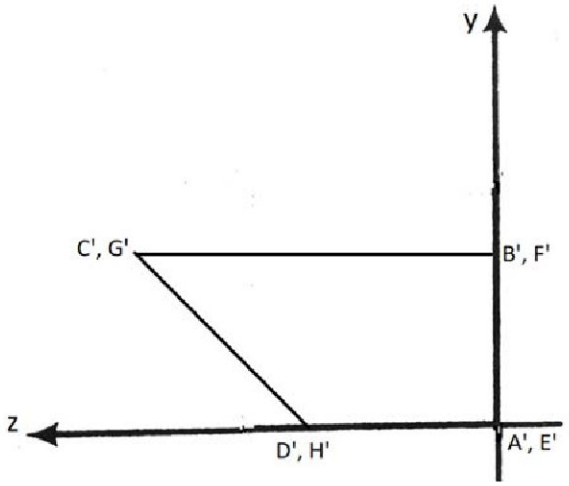


Fig. 5.6 Right side view projection in y-z plane

Top view: In fig.5.7, projection on x-z plane and y=0. So the new coordinate of a given object after projection can be as;

$$\begin{matrix} A' \\ B' \\ C' \\ D' \\ E' \\ F' \\ G' \\ H' \end{matrix}\begin{pmatrix} x'_1 & y'_1 & z'_1 & 1 \\ x'_2 & y'_2 & z'_2 & 1 \\ x'_3 & y'_3 & z'_3 & 1 \\ x'_4 & y'_4 & z'_4 & 1 \\ x'_5 & y'_5 & z'_5 & 1 \\ x'_6 & y'_6 & z'_6 & 1 \\ x'_7 & y'_7 & z'_7 & 1 \\ x'_8 & y'_8 & z'_8 & 1 \end{pmatrix} = \begin{bmatrix} 4 & 0 & 0 & 1 \\ 4 & 4 & 0 & 1 \\ 4 & 4 & 8 & 1 \\ 4 & 0 & 4 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 4 & 0 & 1 \\ 0 & 4 & 8 & 1 \\ 0 & 0 & 4 & 1 \end{bmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{matrix} A' \\ B' \\ C' \\ D' \\ E' \\ F' \\ G' \\ H' \end{matrix} \begin{pmatrix} x'_1 & y'_1 & z'_1 & 1 \\ x'_2 & y'_2 & z'_2 & 1 \\ x'_3 & y'_3 & z'_3 & 1 \\ x'_4 & y'_4 & z'_4 & 1 \\ x'_5 & y'_5 & z'_5 & 1 \\ x'_6 & y'_6 & z'_6 & 1 \\ x'_7 & y'_7 & z'_7 & 1 \\ x'_8 & y'_8 & z'_8 & 1 \end{pmatrix} = \begin{bmatrix} 4 & 0 & 0 & 1 \\ 4 & 0 & 0 & 1 \\ 4 & 0 & 8 & 1 \\ 4 & 0 & 4 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 8 & 1 \\ 0 & 0 & 4 & 1 \end{bmatrix}$$

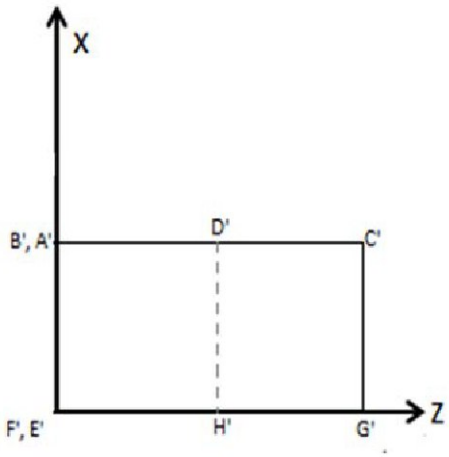


Fig.5.7 Top view projection in x-z plane

Axonometric Projections are orthographic projection in which the direction of projection is not parallel to any of the three principal axes. They are defined according to the angles made between the coordinate axes in screen projection. They can be classified into three standard types known as isometric, diametric, and trimetric.

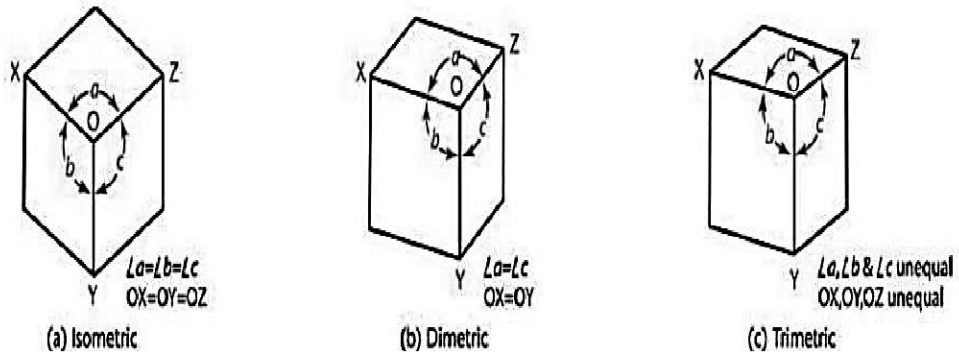


Fig. 5.3 Axonometric projection

In **Isometric projection,** the direction of projection makes equal angles with all three principal axes. In fig.5.3.(a), angles ∠a, ∠b and ∠c are same in this projection view.

In Diametric projection, the direction of projection makes equal angles with exactly two of the principle axes. In fig.5.3.(b), angle ∠a and ∠c are same but ∠b is not equal angle in this projection view.

In Trimetric projection, the direction of projection makes unequal angles with the three principal axes. In fig.5.3.(c), angles ∠a, ∠b and ∠c are not same in this projection view.

5.3.1 OBLIQUE PROJECTION

When projector lines are parallel but the angle between the projectors and the plane of projection is not equal to 90°. In oblique projection, we can view the object better than orthographic projection. Two sub-categories of oblique projections are: Cavalier projection and Cabinet projection.

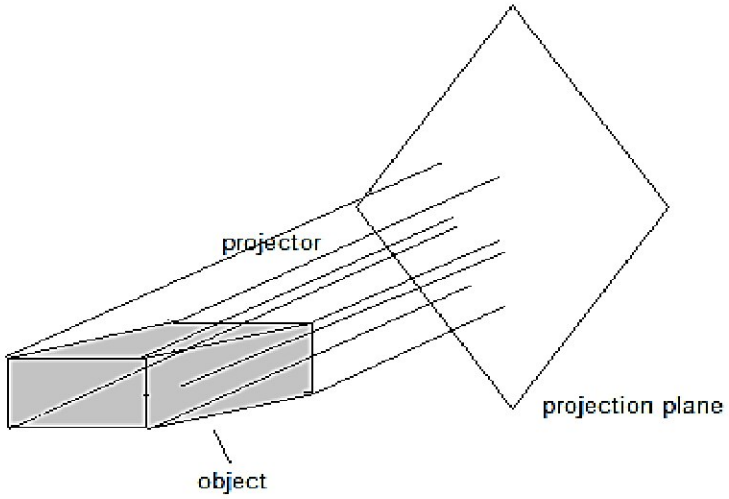


Fig. 5.3 Oblique projection

Some common sub-categories of oblique projections are: Cavalier projection and Cabinet projection.

Cavalier projection- The direction of projection is so chosen that there is no foreshortening of lines perpendicular to the plane of projection. In fig 5.4.

Cabinet projection- The direction of projection is s chosen that lines perpendicular to the plane of projection are foreshortened by half their length. In fig 5.4.

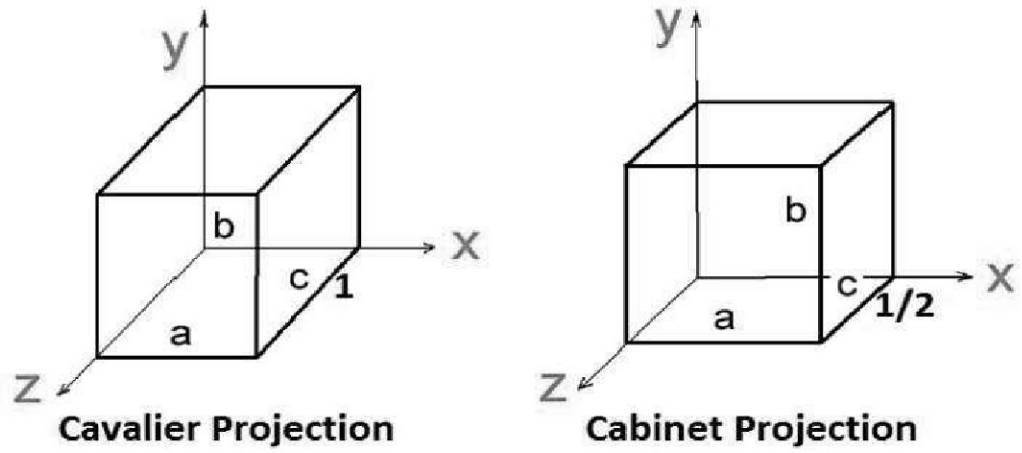


Fig. 5.4 Cavalier and Cabinet projection

Oblique parallel projection onto plane. The projectors make an angle β with the plane of projection in fig.5.5

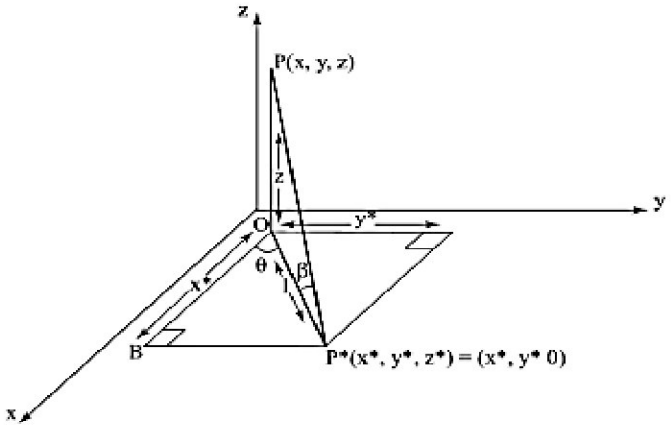


Fig 5.5 Oblique parallel projection onto plane

The line of length l, which joins O and P* (in the projection plane) is making an angle θ with the x-axis. From the triangle ΔOBP*, we have

$$x^* = x + l\,cos\theta$$

$$y^* = y + l\,sin\theta$$

Also, from ΔPOP*, we get,

$$tan\,tan\,\beta = \frac{z}{l} \quad or \quad l = \frac{z}{tantan\,\beta}$$

Substituting the value of l yields

$$x^* = x + \frac{z}{tantan\,\beta}cos\theta \quad and \quad y^* = y + \frac{z}{tantan\,\beta}sin\theta$$

In oblique projection the lines perpendicular to the plane of projection are foreshortened by the direction of projection. The change in length of the projected line is measured in terms of the foreshortening factor f with respect to a given direction of projection.

Θ, is the angle which the projected line OP* makes with the positive x-axis.

Thus the foreshortening in the z-direction is

$$= \frac{|OP|}{|OP^*|} +$$

$$= \frac{z}{tantan\,\beta}$$

$= zf, \quad where\ f = \frac{1}{tantan\,\beta}$ is the foreshortening factor

In matrix form the oblique transformation can be expressed as

$$[T_{Ob}] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ f\cos\theta & f\sin\theta & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where f the foreshortening factor and θ is the arbitrary angle made with the x-direction.

5.5 ISOMETRIC PROJECTION

It is a type of Orthographic projection. In this special case the direction of projection makes equal angles with all three principal axes.

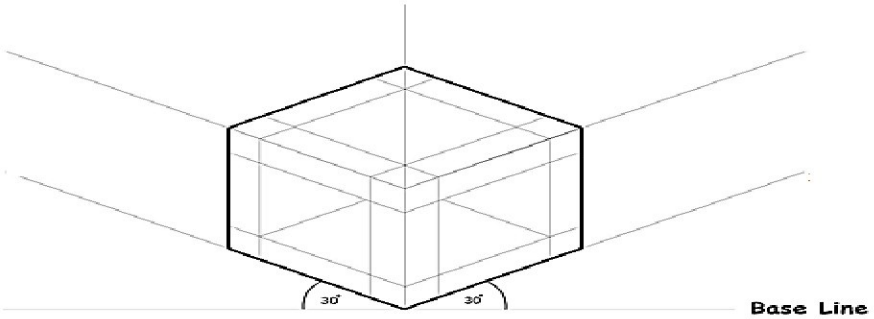


Fig. 5.4 Isometric projection

5.6 PERSPECTIVE PROJECTION

In this type of projection, the centre of projection is a unique finite point, so projection lines are not parallel to each other. If the position of projection point and the projection plane is modified, then the result of perspective projection will also change. In computer graphics, perspective projection is a technique employed to generate images or photographs that look so natural. When a person sees scenes in day-to-day life, the far-away objects look smaller relative to closer objects. This projection's property can provide knowledge about depth. Thus, artists often employ perspective projection for drawing three-dimensional sceneries. An important aspect of the perspective projection is that this concept can preserve straight lines and facilitates to project the end points of three-dimensional lines alone, and then draw a two-dimensional line between the projected end points.

In fig.5.5 Perspective projection depends on the relative position of the eye and the view plane. In the usual arrangement the eye lies on the z-axis and the view plane is the xy-plane. To determine the projection of 3D point connects the point and the eye by a straight line, where the line intersects the view plane. This intersection point is the projected point.

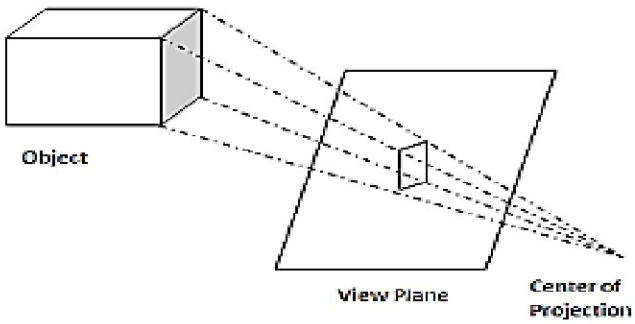


Fig. 5.5 Perspective projection

Two important characteristics of this concept are listed below:

1. Perspective foreshortening
2. Vanishing point

Perspective foreshortening: Due to foreshortening, lengths and objects seem small from the centre of projection. The more an artist increases the distance from the projection's centre, the small will be the appearance of the object.

Vanishing point: Vanishing point is considered as a side effect of the perspective projection. Parallel lines typically tend to meet on a "vanishing point".

Based on the number of vanishing points, the perspective projection is of three types, and they are listed below:

1. Single-point perspective projection
2. Double-point perspective projection
3. Triple-point perspective projection

Single-point perspective projection: In single-point perspective projection, there will be only one vanishing point, in Fig.5.6(a).

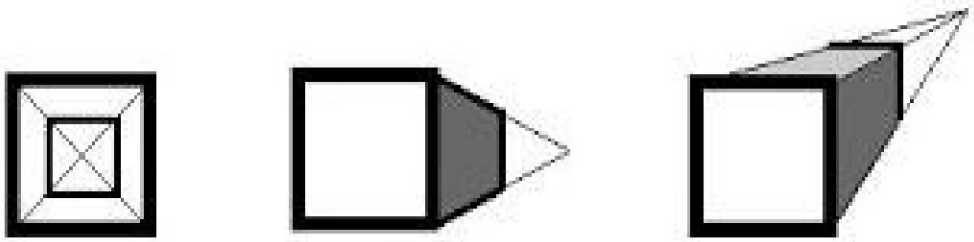


Fig. 5.6 (a) single point perspective projection

Double-point perspective projection: In double-point perspective projection, there will be a couple of vanishing points. The first vanishing point will be in the direction of "x". The second vanishing point will be in the direction of "y" Fig.5.6 (b).

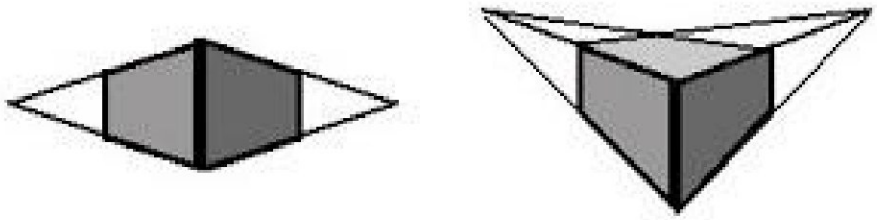


Fig. 5.6 (b) double point perspective projection

Triple-point perspective projection: In triple-point perspective projection, there will be triple vanishing points. The first vanishing point will be in the direction of "x". The second vanishing point will be in the direction of "y". The third vanishing point will be in two directions Fig.5.6(c).

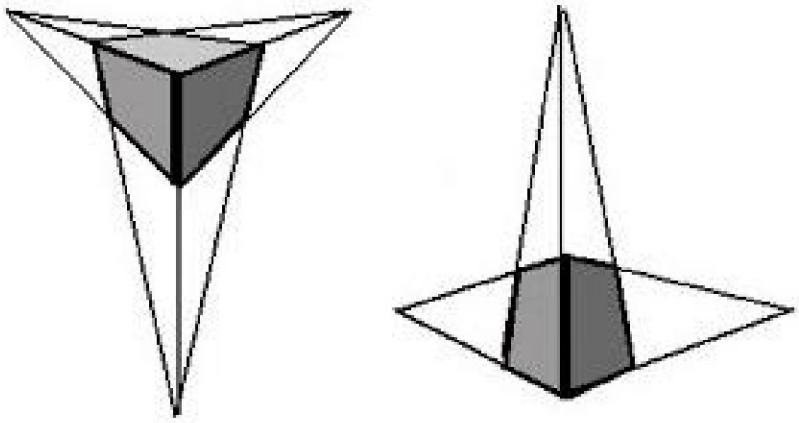


Fig. 5.6 (c) Triple point perspective projection

5.7 SUMMARY

The viewing transformation converts objects from their 3-dimensional camera-space coordinates into the appropriate 2-dimensional raster-space coordinates. The camera coordinate system is a coordinate system with the camera at the origin, looking out over the positive z axis. It is, essentially, the scene from the camera's point of view. The raster coordinate system is the space of the pixels on the monitor. Connecting these two coordinate systems there is a special coordinate system known as the screen coordinate system. The screen coordinate system is, conceptually, the same as the film plane of a camera. It is usually best to consider both the screen coordinate system and the raster coordinate system to be two-dimensional.

There are basically two types of projections parallel projection and perspective projection. Centre of projection is located at infinity in parallel projection whereas centre of projection at finite point in perspective projection. Parallel projection further classified in orthogonal and oblique projection. In

orthogonal projection, projection lines are perpendicular to the plane of projection. If projection lines are parallel to axis, 6 possible view of object can be seen. There are some cases in which projection lines are not parallel to the any principle axis; this is called axonometric projection a special case of orthogonal projection. Isometric projection is a type of axonometric orthogonal projection. In oblique projection, projector lines are parallel but the angle between the projectors and the plane of projection is not equal to 90°. In oblique projection, we can view the object better than orthographic projection. Two sub-categories of oblique projections are: Cavalier projection and Cabinet projection.

5.8 TECHNICAL QUESTIONS

1. What do you mean by viewing transformation? Also explain view port.
2. Why we need such 3D-to-2D transformation? Draw the hierarchy of this transformation.
3. Write down the differences between parallel and perspective projection.
4. Explain the mechanism of orthogonal projection in detail. Draw 6-view orthogonal projection of a 3D object assumed by you.
5. What is axonometric projection? Explain each case with proper diagram. Also explain isometric projection in brief.
6. Show all the six views orthogonal projection of a given object with vertices A(0,0,0), B(0,2,0), C(2,0,0), D(2, 2, 0), E (0,1,2), F(0,3,2), G(2,1,2), H(2,3,2). Also represent the given object in terms of Homogeneous-coordinates.
7. What is oblique projection? Explain the differences between oblique and orthogonal projection.
8. Explain both cavalier and cabinet projection with proper diagrammatical example. Also brief about foreshortening factor.
9. What is vanishing point? Explain the types of perspective projection based on number of vanishing points.
10. Explain the terms: Projection line, projection plan, centre of projection, and angle of projection.

Uttar Pradesh Rajarshi Tandon
Open University

Master of Computer Science

MCS-116
Computer Graphics

BLOCK
3
MODELING & RENDERING

UNIT-6

Curves and Surfaces

UNIT-7

Visible – Surface Detection

UNIT-8

Polygon Rendering and Ray Tracing Methods

Course Design Committee

Prof. Ashutosh Gupta
Director (In-charge)
School of Computer and Information Science, UPRTOU Prayagraj

Prof. Suneeta Agarwal
Department of CSE.
Motilal Nehru National Institute of Technology, Allahabad, Prayagraj

Dr. Upendra Nath Tripathi **Author**
Associate Professor
Department of Computer Science
Deen Dayal Upadhyaya Gorakhpur University, Gorakhpur

Dr. Ashish Khare
Associate Professor
Dept. of Computer Science, Allahabad University

Ms. Marisha
Assistant Professor (Computer Science)
School of Science, UPRTOU Prayagraj

Mr. Manoj Kumar Balwant
Assistant Professor (Computer Science)
School of Science, UPRTOU Prayagraj

Course Preparation Committee

Dr. Divya Kumar **Author**
Assistant Professor
Department of Computer Science & Engineering
Motilal Nehru National Institute of Technology Prayagraj

Prof. Abhay Saxena **Editor**
Dean, School of TCM
Dev Sanskriti Vishwavidyalaya, Haridwar -Uttarakhand

Prof. Ashutosh Gupta **Director (In-Charge)**
School of Computer & Information Sciences
UPRTOU Prayagraj

Mr. Manoj Kumar Balwant **Coordinator**
Assistant Professor (computer science)
School of Sciences, UPRTOU Prayagraj

©UPRTOU, Prayagraj - 2020
ISBN :

UNIT-6 CURVES AND SURFACES

Structure:

6.1 INTRODUCTION

A curve is an infinitely comprehensive series of points. Every point beyond endpoints has two neighbours. Curves can be divided into three broad categories: explicit, implicit, and parametric curves. Objects are a set of surfaces. 3D representation of objects is divided into two categories: representation of boundaries and representation of the space-partitioning.

Polygon is a representation of the surface. It is primitive which is closed in nature. It is formed using a collection of lines. There are two types of polygon: concave and convex.

6.2 OBJECTIVES

After the end of this unit, you should be able to understand the basics of curves and surfaces including.

- Polygon Representation Methods like (Face/vertex/winged/dynamic)
- Bezier curves, surfaces and plane equations.
- Surface of Revolution

6.3 POLYGON REPRESENTATION METHODS

6.3.1 POLYGON SURFACES

The polygon surfaces are popular in design, and solid-modelling applications as their wireframe display can be made easily to provide a general

indication of surface structure. The realistic scenes are produced for illumination through interpolation of shading patterns across the polygon surface. Polygons are easy to process, thus speeding up rendering and display of objects. Many systems allow the definition of objects in other ways (such as splines) but reduce all objects to polygons for processing. Thus, some systems allow objects to be described in other ways (such as splines) but reduce all objects to polygons for processing.

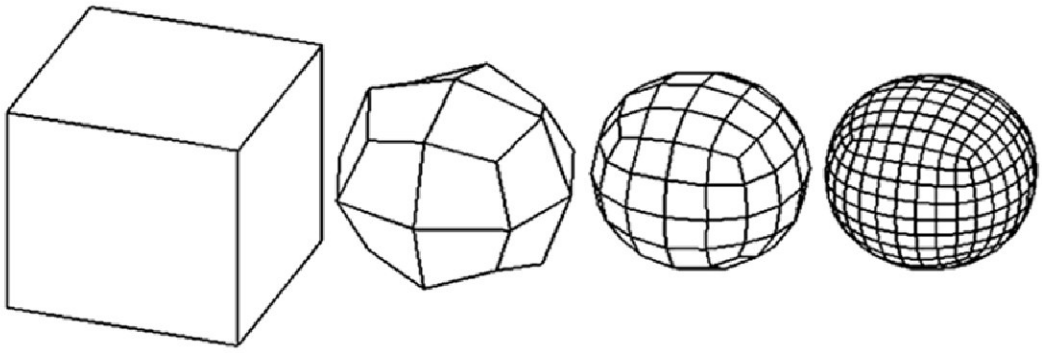


Fig. 6.1 Polygon Surface

6.3.2 POLYGON MESH

A set of line segments and polygon approximate 3D surfaces and solids. Those surfaces are called polygonal meshes. For such meshes at most, two polygons share each of the edges. The project's skin in fig.6.2 is a variation of a set of polygons or faces. Popular polygon mesh types include a triangle strip and quadrilateral mesh.

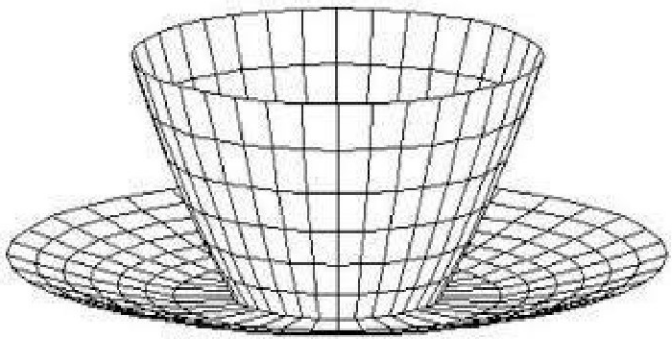


Fig. 6.2 Quadrilateral mesh surface

A polygon mesh is a surface that is constructed out of a set of polygons that are joined together by common edges. In 3d computer graphics and solid modelling, it is a set of vertices, edges, and faces that form the shape of a polyhedral structure. Vertices, edges, faces, polygons, and surfaces are basic elements of mesh to construct an object; in fig.6.3, these elements are shown.

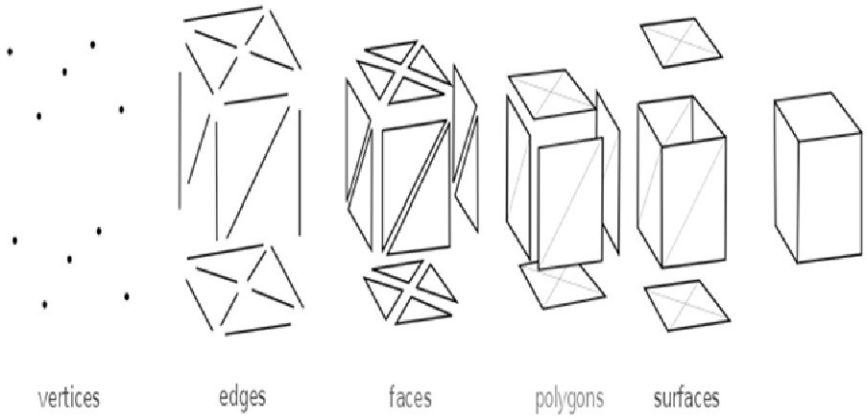


Fig. 6.3 Elements of polygon mesh

One can represent polygon meshes in a variety of ways, using different methods to store the vertex, edge and face data. These include:

1. Vertex-vertex meshes
2. Face-vertex meshes
3. Winged-edge meshes
4. Render dynamic meshes

Vertex-vertex meshes- Represent an object as a set of vertices connected to other vertices. This is the simplest representation, but not commonly used since the face, and is implicit in the edge detail. Thus, to generate a list of faces for rendering, one need to traverse the data. Additionally, edge and face operations are not quickly accomplished. Fig 6.4, a polygon with ten vertices v0, v1, v2, v3, v4, v5, v6, v7, v8, and v9. Vo at the origin.

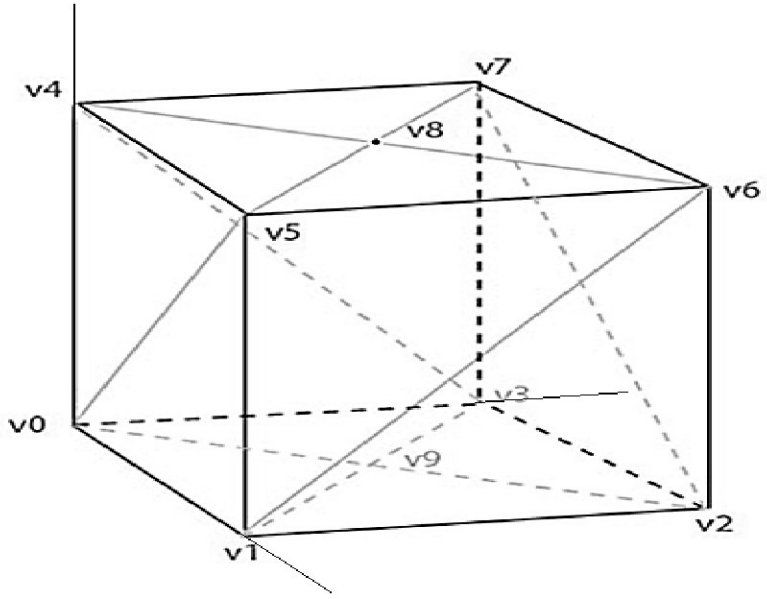


Fig. 6.4 vertex-vertex meshes

Vertex List		
v0	0,0,0	v1 v5 v4 v3 v9
v1	1,0,0	v2 v6 v5 v0 v9
v2	1,1,0	v3 v7 v6 v1 v9
v3	0,1,0	v2 v6 v7 v4 v9
v4	0,0,1	v5 v0 v3 v7 v8
v5	1,0,1	v6 v1 v0 v4 v8
v6	1,1,1	v7 v2 v1 v5 v8
v7	0,1,1	v4 v3 v2 v6 v8
v8	.5, .5, 1	v4 v5 v6 v7
v9	.5, .5, 0	v0 v1 v2 v3

Vertex list contains three columns; first column is for vertex, second column is for coordinate of that vertex, and third column shows the connected vertices from specified vertex.

Face-vertex meshes- One can represent an object as a set of faces and a set of vertices. It is the most commonly used representation of mesh, being the input usually embraced by modern hardware graphics.

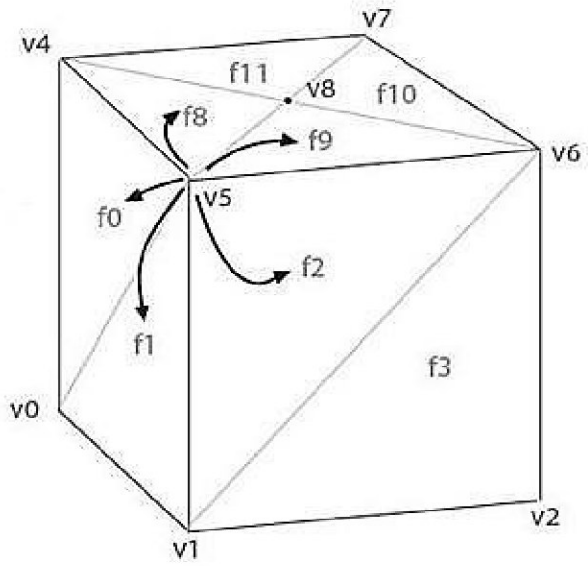


Fig. 6.5 Face-vertex meshes

In fig.6.5, there are 15 faces (7 faces are in back side) and 10 vertices. Every 15 faces are triangle. We can see, v5 attached to 5 faces (f0 f1 f2 f9 f8). First table is showing vertex and associated faces, second table showing faces and their associated vertices.

Face List	
f0	v0 v4 v5
f1	v0 v5 v1
f2	v1 v5 v6
f3	v1 v6 v2
f4	v2 v6 v7
f5	v2 v7 v3
f6	v3 v7 v4
f7	v3 v4 v0
f8	v8 v5 v4
f9	v8 v6 v5
f10	v8 v7 v6

f11	v8 v4 v7
f12	v9 v5 v4
f13	v8 v6 v5
f14	v9 v7 v6
f15	v9 v4 v7

Vertex List		
v0	0,0,0	f0 f1 f12 f15 f7
v1	1,0,0	f2 f3 f13 f12 f1
v2	1,1,0	f4 f5 f14 f13 f3
v3	0,1,0	f6 f7 f14 f15 f5
v4	0,0,1	f6 f7 f0 f8 f11
v5	1,0,1	f0 f1 f2 f9 f8
v6	1,1,1	f2 f3 f4 f10 f9
v7	0,1,1	f4 f5 f6 f11 f10
v8	.5, .5, 1	f8 f9 f10 f11
v9	.5, .5, 0	f12 f13 f14 f15

Winged-edge meshes- Winged-edge meshes, invented by Baumgart in 1975, specifically represent the vertices, faces, and edges of a network. This representation is commonly used in modelling programs to provide full versatility in modifying the mesh configuration dynamically, so operations of splitting and merging can be performed easily. In fig.6.6, object shown with vertices, face and edges of each side. Wing structure show an edge between two faces and associated edges at the vertex. Edge v1-v2, v2 is front and v1 is back vertex, CCW edge is nothing but just an edge starting from counter clockwise direction and CW edge for clockwise direction.

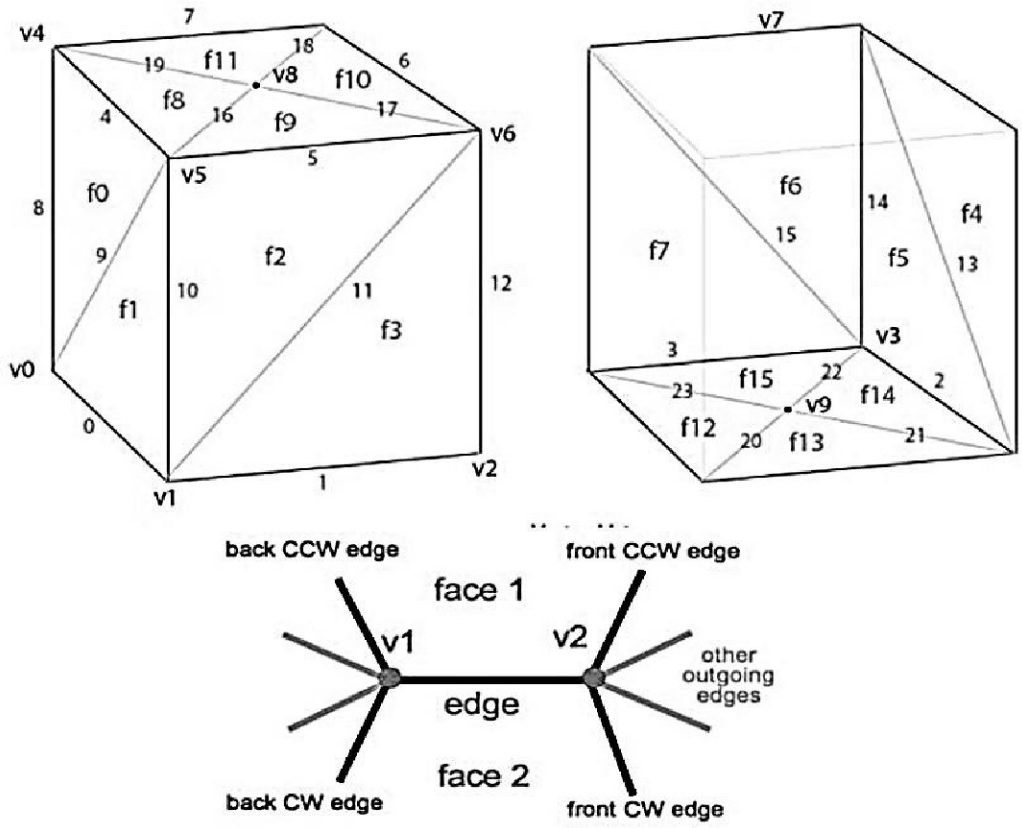


Fig.6.6 Winged-edge meshes structure of cube

Vertex List		
V0	0,0,0	8 9 0 23 3
V1	1,0,0	10 11 1 20 0
V2	1,1,0	12 13 2 21 1
V3	0,1,0	14 15 3 22 2
V4	0,0,1	8 15 7 19 4
V5	1,0,1	10 9 4 16 5
V6	1,1,1	12 11 5 17 6
V7	0,1,1	14 13 6 18 7
V8	.5, .5, 1	16 17 18 19

V9	.5, .5, 0	20 21 22 23
Face List		
F0	4 8 9	
F1	0 10 9	
F2	5 10 11	
F3	1 12 11	
F4	6 12 13	
F5	2 14 13	
F6	7 14 15	
F7	3 8 15	
F8	4 16 19	
F9	5 17 16	
F10	6 18 17	
F11	7 19 18	
F12	0 23 20	
F13	1 20 21	
F14	2 21 22	

F15	3 22 23

Edge List			
			CCW/CW edges
E0	V0 v1	F1 f12	9 23 10 20
E1	V1 v2	F3 f13	11 20 12 21
E2	V2 v3	F5 f14	13 21 14 22
E3	V3 v0	F7 f15	15 22 8 23
E4	V4 v5	F0 f8	19 8 16 9
E5	V5 v6	F2 f9	16 10 17 11
E6	V6 v7	F4 f10	17 12 18 13
E7	V7 v4	F6 f11	18 14 19 15
E8	V0 v4	F7 f0	3 9 7 4
E9	V0 v5	F0 f1	8 0 4 10
E10	V1 v5	F1 f2	0 11 9 5
E11	V1 v6	F2 f3	10 1 5 12
E12	V2 v6	F3 f4	1 13 11 6
E13	V2 v7	F4 f5	12 2 6 14
E14	V3 v7	F5 f6	3 15 13 7
E15	V3 v4	F6 f7	14 3 7 15
E16	V5 v8	F8 f9	4 5 19 17
E17	V6 v8	F9 f10	5 6 16 18
E18	V7 v8	F10 f11	6 7 17 19

E19	V4 v8	F11 f8	7 4 18 16
E20	V1 v9	F12 f13	0 1 23 21
E21	V2 v9	F13 f14	1 2 20 22
E22	V3 v9	F14 f15	2 3 21 23
E23	V0 v9	F15 f12	3 0 22 20

Render dynamic meshes- Winged-edge meshes are not the only type that allows for dynamic geometry adjustments. A new representation that combines winged-edge meshes with face-vertex meshes is rendered dynamic mesh that explicitly stores both the vertices of a vertex face and faces and the faces and vertices of an edge.

6.3.3 POLYGON TABLE

This is the polygon surface description, using vertex coordinates and other attributes:

1. Geometric data table: vertices, edges, and polygon surfaces.
2. Attribute table: eg. Degree of transparency and surface reflectivity etc.

Some consistency checks of the geometric data table:

- At least 2 edges of each vertex are classified as endpoints
- Each edge is part of a polygon or more
- Every single polygon is closed

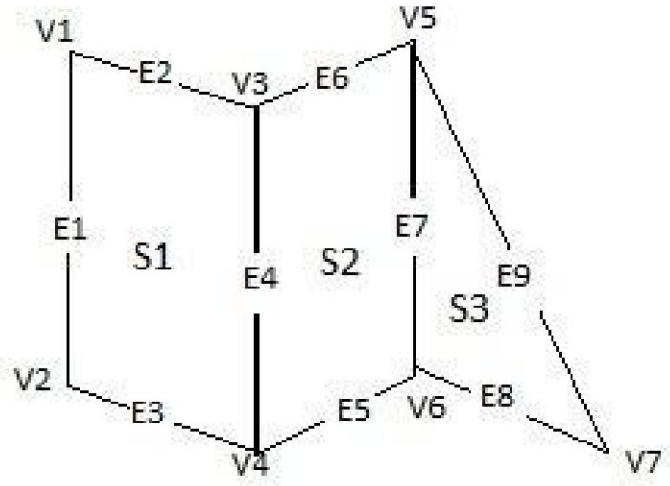


Fig. 6.3 Three surface polygon

Vertex Table
v1: x1, y1, z1
v2: x2, y2, z2
v3: x3, y3, z3
v4: x4, y4, z4
v5: x5, y5, z5
v6: x6, y6, z6
v7: x7, y7, z7

Polygon- Surface Table
S1 – E1, E2, E3, E4
S2 - E4, E5, E6, E7
S3 - E7, E8, E9
EDGE Table
E1 – v1, v2
E2 – v1, v3
E3 – v2, v4
E4 – v3, v4
E5 – v4, v6
E6 – v3, v5
E7 – v5, v6
E8 – v6, v7
E9 – v5, v7

6.3.4 PLANE EQUATION

The plane surface equation is expressed as:

$$Ax + By + Cz + D = 0$$

On a plane let (x,y,z) be any point, and where the A, B, C, and D coefficients are constants representing the plane's spatial properties. Using the coordinate values of the plane's non-collinear points, the values of A, B, C, and D can be obtained

by solving the set of three equations. The three vertices of the plane are assumed to be

(x1, y1, z1), (x2, y2, z2), and (x3, y3, z3).

The equation for the ratios A/D, B/D, and C/D is solved for obtaining the values of A, B, C, and D.

$$(A/D)\, x_1 + (B/D)\, y_1 + (C/D)\, z_1 = -1$$

$$(A/D)\, x_2 + (B/D)\, y_2 + (C/D)\, z_2 = -1$$

$$(A/D)\, x_3 + (B/D)\, y_3 + (C/D)\, z_3 = -1$$

To obtain the above equations in determinant form, apply Cramer's rule to the above equations.

$A = [1\ y_1\ z_1\ 1\ y_2\ z_2\ 1\ y_3\ z_3\ ]$ $B = [x_1\ 1\ z_1\ x_2\ 1\ z_2\ x_3\ 1\ z_3\ ]$ $C = [x_1\ y_1\ 1\ x_2\ y_2\ 1\ x_3\ y_3\ 1\ ]$ $D = [x_1\ y_1\ z_1\ x_2\ y_2\ z_2\ x_3\ y_3\ z_3\ ]$

It is said that, for any point (x, y, z) with parameters A, B, C, and D,

- $Ax + By + Cz + D \neq 0$ implies that the point is not on the plane.
- $Ax + By + Cz + D < 0$ implies that the point is inside the surface.
- $Ax + By + Cz + D > 0$ implies that the point is outside the surface.

6.4 BEZIER CURVE

In computer graphics, Bezier curves are used to generate curves that appear relatively smooth on any scale. Lines of polygons are not as smooth as those of Bezier curves. Linear interpolations arise from Bezier curves. It just picks a point in between two points. Control points describe a Bezier Curve. Control points can be two, three, four or more. In Fig.6.4 a set of control points b0, b1, b2 and b3 determines this Bezier curve. Points b0 and b3 are corner ends. Points b1 and b2 describe curve form.

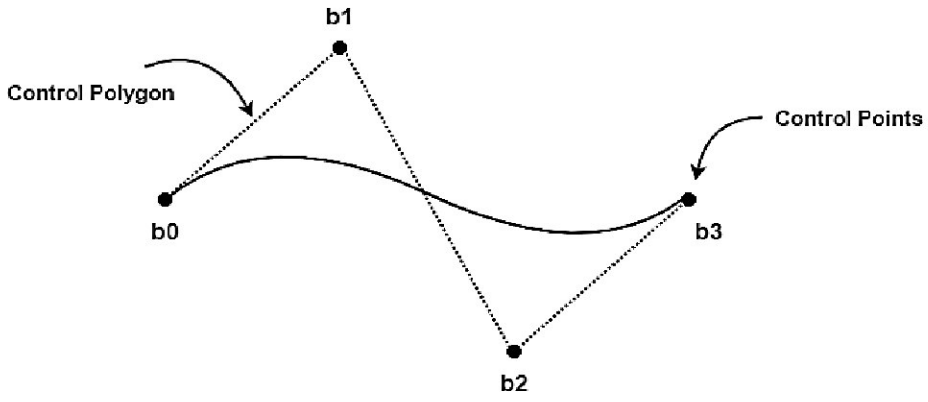


Fig. 6.4 Bezier curves

Bezier curve Equation $P(t) = \sum_{i=0}^{n} B_i J_{n,i}(t)$

Here

t is any parameter where 0 <= t <= 1

P(t) = Any point lying on the bezier curve

B_i = i^{th} control point of the bezier curve

n = degree of the curve= control points - 1

$J_{n,i}(t)$ = Blending function = $C(n,i)t^i(1-t)^{n-i}$ where C(n,i) = n! / i!(n-i)!

Cubic Bezier curve- Cubic Bezier curve is a Bezier curve with degree 3. The total number of control points in a cubic Bezier curve is 4. Fig 6.4 is a Cubic Bezier curve.

$$P(t) = \sum_{i=0}^{n} B_i J_{n,i}(t)$$

Substitute n=3 for a cubic Bezier curve

$P(t) = B_0J_{3,0}(t) + B_1J_{3,1}(t) + B_2J_{3,2}(t) + B_3J_{3,3}(t)$

So, now equation is $\mathbf{P(t) = B_0(1-t)^3 + B_13t(1-t)^2 + B_23t^2(1-t) + B_3t^3}$

Applications of Bezier Curves-

Bezier curves have their applications in the following fields-

1. Computer Graphics-
   - Bezier curves are commonly used for forming smooth curves in computer graphics.
   - The curve is fully embedded in its control points convex hull.
   - So, the points can be displayed graphically and used to intuitively manipulate the curve.
2. Animation-
   - In animation applications such as Adobe Flash and synfig, bezier curves are used to trace the movement.
   - In Bezier curves the users outline the desired path.
   - The application generates the frames needed to move the object along the path.
   - Bezier curves are also used for 3D animation to describe 3D paths, as well as 2D curves.
3. Fonts-
   - True style fonts use composite Bezier curves consisting of Bezier quadratic curves.
   - Modern imaging systems such as postscript, asymptote etc. use Bezier composite curves consisting of cubic Bezier curves to draw curved forms.

Example: Given a bezier curve with 4 control points, B_0[1 0] , B_1[3 3] , B_2[6 3] , B_3[8 1]. Determine any five points on the curve. Draw a rough curve diagram, too.

Solution- We have-

- The given curve is defined by 4 control points.
- So, the given curve is a cubic Bezier curve.

The parametric equation for a cubic Bezier curve is-

$\mathbf{P(t) = B_0(1-t)^3 + B_1 3t(1-t)^2 + B_2 3t^2(1-t) + B_3 t^3}$

Substituting the control points B_0, B_1, B_2 and B_3, we get-

$P(t) = [1\ 0](1-t)^3 + [3\ 3]3t(1-t)^2 + [6\ 3]3t^2(1-t) + [8\ 1]t^3$ .....(1)

Now,

To get 5 points lying on the curve, assume any 5 values of t lying in the range

$0 <= t <= 1.$

Let 5 values of t are 0, 0.2, 0.5, 0.7, 1.

For t = 0 :

Substituting t=0 in (1), we get-

$P(0) = [1\ 0](1-0)^3 + [3\ 3]3(0)(1-t)^2 + [6\ 3]3(0)^2(1-0) + [8\ 1](0)^3$

$P(0) = [1\ 0] + 0 + 0 + 0$

$P(0) = [1\ 0]$

For t = 0.2

Substituting t=0.2 in (1), we get-

$P(0.2) = [1\ 0](1-0.2)^3 + [3\ 3]3(0.2)(1-0.2)^2 + [6\ 3]3(0.2)^2(1-0.2) + [8\ 1](0.2)^3$

$P(0.2) = [1\ 0](0.8)^3 + [3\ 3]3(0.2)(0.8)^2 + [6\ 3]3(0.2)^2(0.8) + [8\ 1](0.2)^3$

$P(0.2) = [2.304\ 1.448]$

For t = 0.5 :

Substituting t=0.5 in (1), we get-

$P(0.5) = [1\ 0](1-0.5)^3 + [3\ 3]3(0.5)(1-0.5)^2 + [6\ 3]3(0.5)^2(1-0.5) + [8\ 1](0.5)^3$

$P(0.5) = [1\ 0](0.5)^3 + [3\ 3]3(0.5)(0.5)^2 + [6\ 3]3(0.5)^2(0.5) + [8\ 1](0.5)^3$

$P(0.5) = [4.5\ 2.375]$

For t = 0.7:

Substituting t=0.7 in (1), we get-

$P(t) = [1\ 0](1-t)^3 + [3\ 3]3t(1-t)^2 + [6\ 3]3t^2(1-t) + [8\ 1]t^3$

$P(0.7) = [1\ 0](1-0.7)^3 + [3\ 3]3(0.7)(1-0.7)^2 + [6\ 3]3(0.7)^2(1-0.7) + [8\ 1](0.7)^3$

$P(0.7) = [5.984\ 2.233]$

For t = 1:

Substituting t=1 in (1), we get-

$P(1) = [1\ 0](1-1)^3 + [3\ 3]3(1)(1-1)^2 + [6\ 3]3(1)^2(1-1) + [8\ 1](1)^3$

$P(1) = [1\ 0] \times 0 + [3\ 3] \times 3 \times 1 \times 0 + [6\ 3] \times 3 \times 1 \times 0 + [8\ 1] \times 1$

$P(1) = [8\ 1]$

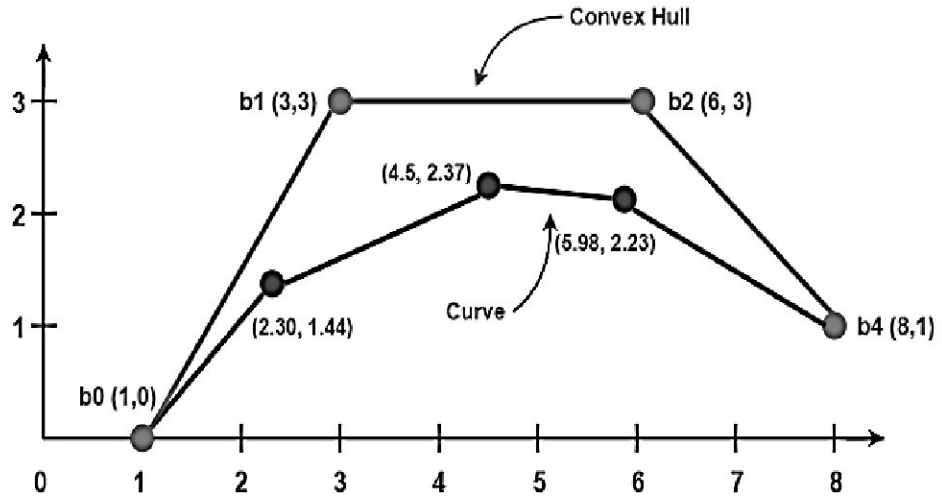


Fig.6.5 Solution curve for this question

Bezier curve properties:-

a. They always passes through control points first and last.

b. Their distinguishing control points are convexly embedded in the hull.

c. Not all points are on curve. That's perfectly natural, we'll see later on how the curve is formed.

d. The convex hull of control points always has a curve inside.

e. At a point t = t0 a given Bezier curve can be subdivided into two Bezier segments which at the point corresponding to the parameter value t = t0 join together.

6.5 BEZIER SURFACE

The Bezier surface is created by blending functions of two orthogonal Bezier curves as the Cartesian product.

$$P(u,v) = \sum_{j=0}^{m} \quad \sum_{k=0}^{n} \quad P_{j,k}\, BEZ_{j,m}(v).BEZ_{k,n}(u)$$

Where j and k are parametric spatial points and represent the position of the knots in real space. The Bezier functions define a given knot's weighting. They are coefficients for Bernstein.

The definition of the Bezier functions is:

$$BEZ_{k,n}(u) = C(n,k)u^k.(1-u)^{u-k}$$

Where C(n, k) stands for binary coefficients. When u=0, for all other points, the function is one for k=0, and zero. When we combine two orthogonal parameters, a Bézier curve is found along each edge of the surface, as defined by the points along that edge. Bezier surfaces are useful for interactive design and the design of the car body was first applied.

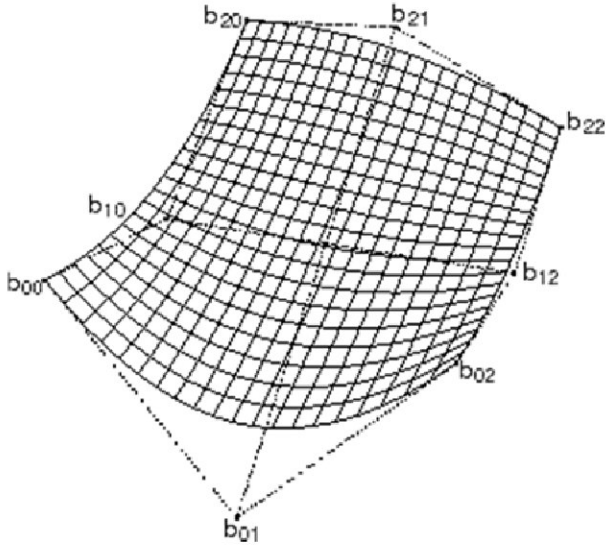


Fig. 6.6 Bezier Surface

6.6 SURFACE OF REVOLUTION

A surface of revolution is a surface in three-dimensional space created by rotating a curve, known as the generatrix, about a straight line in the same plane, known as the axis. In many cases, this axis is the x-axis or the y-axis. Therefore, the resulting surface still exhibits azimuthal symmetry. Examples of surfaces of revolution include the apple, cone (excluding the base), conical frustum (excluding the ends), cylinder (excluding the ends), Darwin-de Sitter spheroid, Gabriel's horn, hyperboloid, lemon, oblate spheroid, parabolic, prolate spheroid, pseudo sphere, sphere, spheroid, and torus (and its generalization, the toroid).

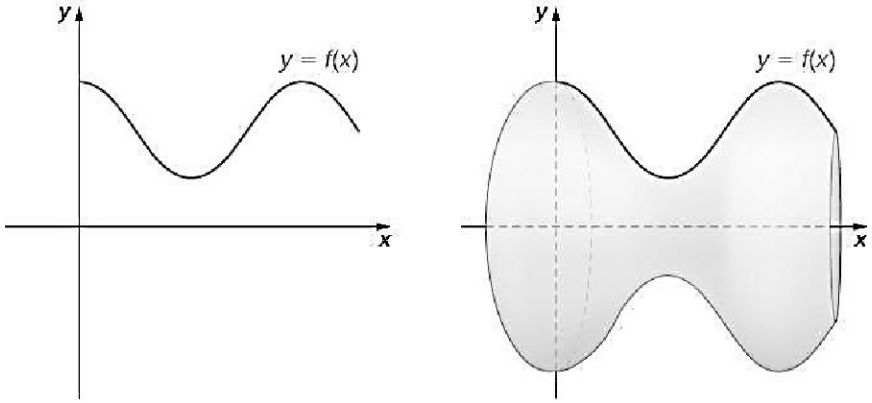


Fig. 6.7 Surface of Revolution about x-axis

Revolution about x-axis-

For a curve defined by $y = f(x) > 0$ on the interval$a \leq x \leq b$, the formula for the surface area is given by

$$S_x = 2\pi \int_a^b f(x)\sqrt{1 + [f'(x)]^2}dx$$

Revolution about y-axis-

For a curve defined by $x = g(y) > 0$ on the interval$c \leq y \leq d$, the formula for the surface area is given by

$$S_y = 2\pi \int_c^d g(y)\sqrt{1 + [g'(y)]^2}dy$$

Example: Let f(x) = √x over the interval [1,4]. Find the surface area of the surface generated by revolving the graph of f(x) around the x–axis. Round off the answer to three decimal places.

Solution- The graph of f(x) and the surface of rotation are shown in Fig.6.8

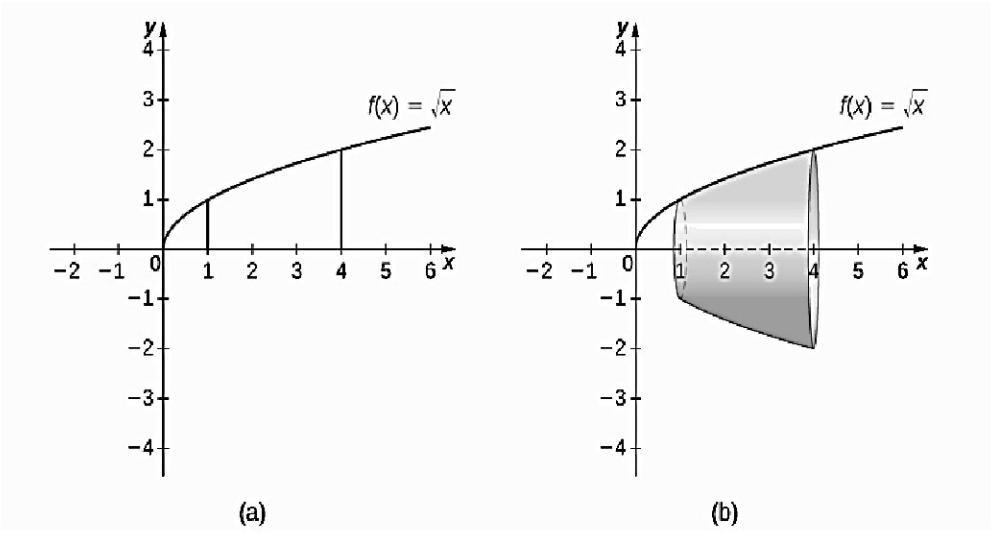


Fig. 6.8 show (a) f(x) graph and (b) surface of revolution

We have $f(x) = \sqrt{x}$

Then $f'(x) = \frac{1}{2\sqrt{x}}$ and $(f'(x))^2 = \frac{1}{4x}$

Surface Area $= \int_a^b (2\pi f(x))\sqrt{1 + (f'(x))^2}dx$

$= \int_1^4 2\pi\sqrt{x}\sqrt{1 + \frac{1}{4x}}dx$

$= \int_1^4 2\pi\sqrt{x + \frac{x}{4x}}dx$

$= \int_1^4 2\pi\sqrt{x + \frac{1}{4}}dx$

Let $u = x + \frac{1}{4}$. Then $du = dx$.

When x=1,then u=5/4, and when x=4, then u=17/4.

So, $$\int_1^4 2\pi\sqrt{x+1/4}dx = \int_{5/4}^{17/4} 2\pi\sqrt{u}du$$

$$= 2\pi\left[\frac{2}{3}u^{3/2}\right]_{5/4}^{17/4}$$

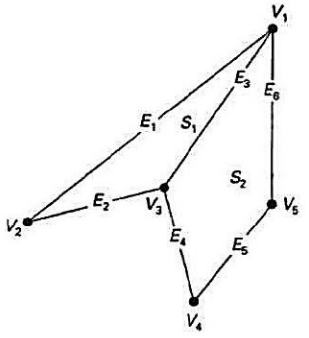
$$= \frac{\pi}{6}[17\sqrt{17} - 5\sqrt{5}] \approx 30.85$$

6.7 SUMMARY

In computer graphics, polygons are used to compose images which appear three-dimensional. Typically, (but not always) triangular, polygons arise when the surface of an object is modelled, vertices are chosen and the object is created in a wire frame model. A polygon can be represented by its polygon surfaces (basic 2D structure), polygon table (consisting of coordinates of edges and vertices), plane equation (equation lines and curves), and polygon mesh (collection of vertices, edges and faces). In computer graphics, bezier curves are used to draw forms, for CSS animation and in many other places. Bezier surfaces are a type of mathematical spline used in computer graphics, computer-assisted design and modeling of finite elements. Bezier process has different conditions and properties to use. We can draw a 3D model of any specific axis by rotating and 2D curve, curve generates a surface which is called a revolution surface.

6.8 TECHNICAL QUESTIONS

1. What is polygon representation? Write down the basic terminologies used in polygon representation.
2. Explain various methods to design polygon mesh.
3. Create polygon tables for polygon given below



4. Explain the properties of Bezier curves. What is Bezier surface?
5. A line in XY plane passes from origin and make 45° from x-axis rotates about y-axis. Calculate the surface area of generated 3D object by the revolution of line. Assume the next end point coordinate of line accordingly.

UNIT-7 VISIBLE - SURFACE DETECTION

Structure :

7.1 INTRODUCTION

When we view a picture containing opaque objects and surfaces, then we can't see those objects from views that are behind from objects closer to the eye. We must remove these hidden surfaces to get a realistic screen image. The identification and removal of these surfaces is called Visible-surface detection or Hidden surface problem.

There are two approaches for removing hidden surface problems

1. Object-Space method
2. Image-space method

The Object-space method is implemented in a physical coordinate system.

The image-space method is implemented in the screen coordinate system.

In the real world, when we see an object we won't be able to see the part of the object that is hidden by an opaque material as it obstructs the light rays from the hidden part. In the computer generation, when objects are projected onto the screen coordinate system, no such automatic elimination takes place. Instead, all parts of every object are visible.

We must apply a hidden line algorithm or hidden surface algorithm to the set of objects to remove these parts and create a more realistic image. These algorithms uses some form of geometric sorting to differentiate between visible parts of objects and those parts that are hidden.

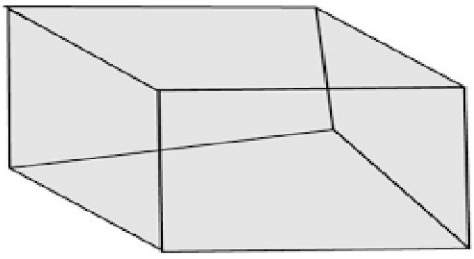
For displaying the 3D objects on a 2D screen, firstly we need to choose the viewing position and then identify only those part of the screen that is visible from the viewing position. Surfaces that are hidden by other opaque surfaces along the line of sight (projection) are invisible to the viewer.

Characteristics of approaches: -

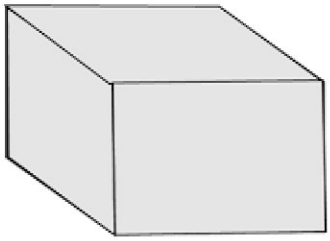
- Require large memory size?
- Require long processing time?
- Applicable to which types of objects?

Considerations:-

- The complexity of the scene
- Type of objects in the scene
- Available equipment
- Static or animated?



Object with hidden line



Object when hidden lines removed

Classification of Visible-Surface Detection Algorithms:

7.1.1 OBJECT-SPACE METHODS

Compare objects as well as parts of objects to each other within the scene definition to find out which surfaces, as a whole, we should label it as visible:

For each object in the scene do

Begin

1. Discover those parts of the object whose view is not hidden by other parts of it or any other object with respect to the viewing specification.
2. Draw those parts in the object color.

End

- Each object is compared with all other objects to find the visibility of the object parts.
- For n objects in the scene, complexity will be O(n2).

- Perform the calculations at the resolution in which the objects are defined (only limited by the computation hardware).
- Computationally it is more expensive when compared with image space methods because the first step is more complex but on the positive side, the display is more accurate, eg. Due to the possibility of intersection between surfaces.
- It is suitable for scenes with a small number of objects and objects with simple relationships with each other.
- It is implemented on the physical coordinate system in which the object is defined.
- In this, calculations are not based on the resolution of the display device so a change of object can be easily adjusted.
- These were developed for a vector graphics system
- Object-based algorithms operate on continuous object data.
- The Vector display used for object method has a large address space
- It requires a lot of calculations if the image is to enlarge.
- If the number of objects in the scene increases, computation time also increases.

7.1.2 IMAGE-SPACE METHODS

- The screen coordinate system is used to implement the image-space method and it is concerned with the final image.
- For each pixel in the image do

  Begin

  1. Determine the object nearest to the viewer that is penetrated by the projector through the pixel.
  2. Draw the pixel in the object color.

End

- Examine all n objects for each pixel to determine the one that is closest to the viewer.
- If there are p pixels in the image, the complexity depends on n and p (O(np).
- The accuracy of the calculation is bounded by the display resolution.
- A change of display resolution requires re-calculation.
- It uses the resolution of the display device and performs the calculation on the basis of that. So, the change is difficult to adjust.

- These are developed for raster devices. The raster display is very flexible as they keep on refreshing the screen by taking the values stored in the frame buffer.
- These operate on object data.
- They are appropriate for applications where accuracy is important. The image can be enlarged without losing accuracy.
- In this method, complexity increases with the complexity of visible parts.

In both method sorting is used a depth comparison of individual lines, surfaces are objected to their distances from the view plane.

7.2 OBJECTIVES

After the end of this unit, you should be able to understand:

- The visible surface detection problem.
- The visible surface detection methods.

7.3 DEPTH BUFFER METHOD

- It is also called Z-Buffer Algorithm.
- The depth buffer algorithm is the simplest image-space algorithm.
- For each pixel on the display screen, we keep a record of the depth of an object within the pixel that lies closest to the observer.
- In addition to depth, we also record the intensity that should be displayed to show the object. Depth buffer algorithm requires 2 arrays, color, and depth each of which is indexed by pixel coordinates (x, y).

  This algorithm compares surface depths at each pixel position on the projection plane. Object depth is usually measured from the view plane along the z-axis of a viewing system.

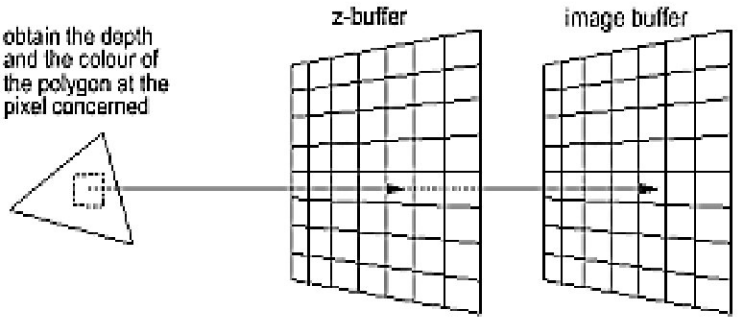


Fig. 7.1 Z-buffer Algorithm

Algorithm:

Step1: Initialize the depth of each pixel.

i.e, d(i, j) = max length (infinite)

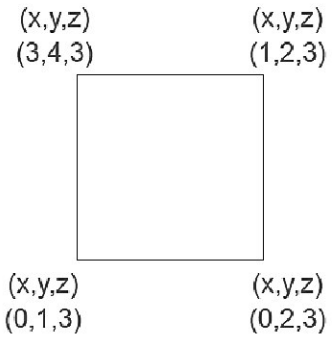
Step 2: Initialize the color value for each pixel

as c(i, j) = background-color

Step 3: For each polygon, do the following steps :

```
for (each pixel in polygon's projection)  {
        find depth i.e, z of the polygon at(x, y) corresponding to a pixel(i, j)
         if (z < d(i, j))
         {
          d(i, j) = z;
         c(i, j) = color;
          }
        }
```

Let's illustrate with an example. Suppose a polygon is given as below :



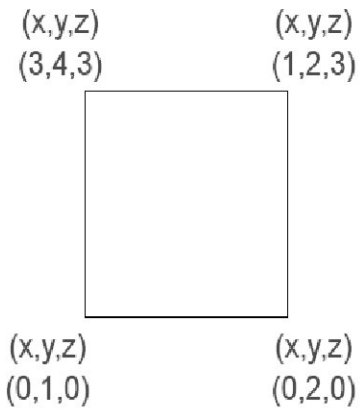
In starting, the depth of each pixel is taken infinite.

∞	∞	∞
∞	∞	∞
∞	∞	∞
∞	∞	∞

As the z value at every position in the given polygon is 3 which is smaller than infinite, so we get d(i,j):

3	3	3
3	3	3
3	3	3
3	3	3

Now, let's change the z values. In the given figure, the z value goes from 0 to 3.



In starting, the depth of each pixel will be infinite as :

∞	∞	∞
∞	∞	∞
∞	∞	∞
∞	∞	∞

Now, the z values generated on the pixel will be different which are as shown below :

3	3	3
2	2	2
1	1	1
0	0	0

Points to remember:

- This method is not good for the cases where only a few objects are to be rendered as the method requires an additional buffer (compared with the Depth-Sort Method) and the overheads involved in updating the buffer.
- This method is not complex and does not require any additional data structures.
- The calculation of the z-value of a polygon can be done incrementally.
- Pre-sorting of polygons is not needed.

- Object-object comparison is not required.
- It can also be applied to non-polygonal objects.
- Hardware implementations of the algorithm are available in some graphics workstations.
- The algorithm could be applied for large images too, eg., apply the algorithm to the 4 quadrants of the image separately to reduce the requirement of a large additional buffer.

Limitations of Depth-Buffer Algorithm:

- The depth buffer Algorithm is not always practical due to the enormous size of depth and intensity arrays,

  Ex- for generating an image with 500 x 500 pixels, it requires 2, 50,000 storage locations for every array.

- To overcome this problem, we divide the image into many smaller images and then apply the depth buffer algorithm.

  Ex- Divide the original 500 x 500 raster into 100 rasters each of 50 x 50 pixels. For processing each small raster an array of only 2500 elements is required, but execution time grows because each polygon is processed many times.

- Precision issues (scintillating, worse with perspective projection.)
- Hard to do analytic Antialiasing
- Hard to simulate translucent polygons.

7.3.1 APPLICATION OF COHERENCE IN VISIBLE SURFACE DETECTION METHODS

- Coherence is the result of local similarity.
- Object properties vary smoothly within a small local region as objects have a continuous spatial extent in the scene. Calculations can then be made incremental.

Types of coherence:

A. Object Coherence:

Do not compare if one object is entirely separate from another. Object visibility can often be decided by examining a circumscribing solid (eg. A sphere or a polyhedron.)

B. Face Coherence:

Compute surface properties only for one part of a face and apply it to adjacent parts after small incremental modification.

C. Edge Coherence:

The Visibility of an edge changes only when it crosses another edge, so if one segment of a nonintersecting edge is visible, the entire edge is also visible.

D. Scan line Coherence:

Line or surface segments visible that are visible in one scan line are also likely to be visible in adjacent scan lines. As a result, the image of a scan line is similar to the image of adjacent scan lines.

E. Area and Span Coherence:

The Span of adjacent groups of pixels in an image is often covered by the same visible face. This coherence is predicated on the assumption that little enough region of pixels will presumably lie within one polygon. Due to this computation effort in searching for those polygons which contain a given screen area (region of pixels) as in some subdivision algorithms will be reduced.

F. Depth Coherence:

The depths of adjacent parts of the same surface are similar.

G. Frame Coherence:

Images of the same scene at consecutive points in time are likely to be similar, despite small changes in objects and viewpoint, except near the edges of moving objects.

7.4 SCAN-LINE METHOD

It is an image space algorithm to identify visible surfaces. Instead of one pixel at a time, it processes one line at a time. It uses the concept area of coherence to speed up the process which is discussed later. Before process the next line we have to group the polygons which are overlapping on intersecting at a given scan line. Z- Value is calculated during scanning a line for each pixel to determine the nearest value. The tables that are used in this algorithm are edge table (ET), Polygon Table (PT), Active Edge table (AET), and flag value to on or off.

Edge Table (ET): It contains:

- coordinate endpoints of each line in the scene
- the inverse slope of each line
- pointers into the polygon table to connect edges to surfaces.

Polygons Table (PT): It contains:

- Surface material property.
- Coefficients of the plane.
- Other surface doter, points of edge table, etc

Active Edge Table (AET): Information of those edges which are crossing the scan line is stored on an active edge table (AET) to keep the track of active edge. Values are stored in ascending order of X.

Flag: Flag value is used to on or off when line move left to right or right to left respectively.

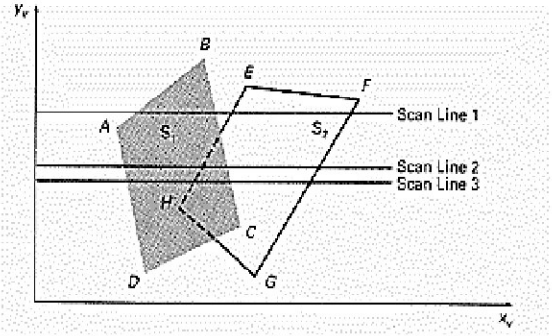


Fig. 7.2 Scan line Algorithm

Algorithm

Step1: Start algorithm

Step2: Initialize the desired data structure

1. Create a polygon table having color, edge pointers, coefficients
2. Establish an edge table contains information regarding, the endpoint of edges, pointer to polygon, inverse slope.
3. Create an Active edge list. This will be sorted in increasing order of x.
4. Create a flag F. It will have two values either on or off.

Step3: Perform the following steps for all scan lines

1. Enter values in Active edge list (AEL) in sorted order using y as value
2. Scan until the flag, i.e. F is on using a background-color
3. When one polygon flag is on, and this is for surface S1enter color intensity as I1into refresh buffer
4. When two or image surface flags are on, sort the surfaces according to depth and use intensity value Sn for the nth surface. This surface will have least z depth value
5. Use the concept of coherence for remaining planes.

Step4: Stop Algorithm

To speed up the process:

Recall the basic idea of polygon filling:

For each scan line crossing a polygon, this method determines the meeting points of the scan line with the polygon edges. These meeting points are sorted from left to right and then the pixels between each intersection pair are filled.

A similar idea is applied here, first, we fill every scan line span by span. When polygons are overlapping on the scan line, we perform depth calculations at their edges to find which polygon should be visible at which span.

We can process any number of overlapping polygon surfaces with this method. Perform the Depth calculation method only when there are polygons overlapping.

If there is no change within the pattern of the intersection of polygon edges with the successive scan lines, it is not necessary to do depth calculations.

This works only if surfaces do not cut through or otherwise cyclically overlap each other. If cyclic overlap happens, we can divide the surfaces to eliminate the overlaps.

The algorithm is applicable to non-polygonal surfaces.

Memory requirement is less than that for depth-buffer method.

Lot of sortings are done on x-y coordinates and on depths.

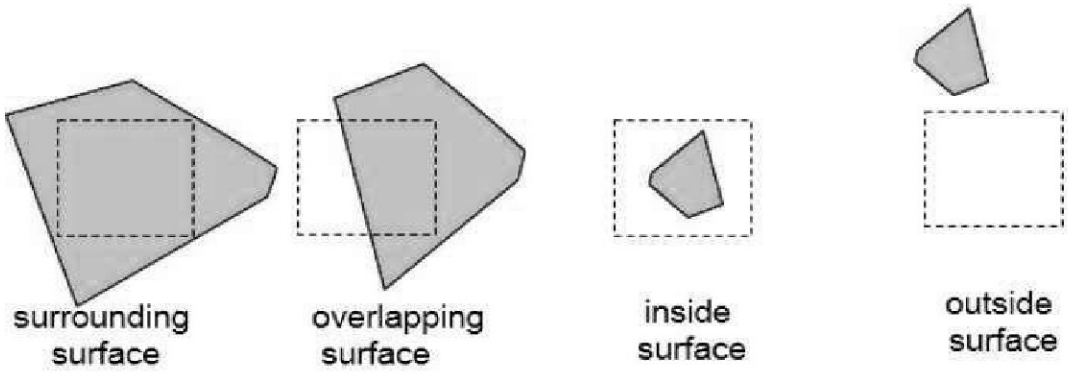
7.5 AREA SUBDIVISION METHOD

Area Subdivision Method was proposed by John Warnock. It is a divide and conquer hidden surface algorithm

This algorithm classifies polygons into trivial or nontrivial cases with respect to the current viewing window.

It is easy to handle trivial cases but nontrivial cases, the current viewing window is recursively divided into four equal subwindows, each of which is then used for reclassifying remaining polygons. Continue this process until all polygons are trivially classified or until the current window reaches the pixel resolution of the screen. At that point, the algorithm changes to a simple z-depth sort of the intersected polygons, and the pixel color becomes that of the polygon closest to the viewing screen.

All polygons are readily classified with respect to the current window into the four categories illustrated in the following Figure:



Surrounding surface – Surface that encloses the area completely.

Overlapping surface – Some part of surface is inside and some part is outside the area.

Inside surface– Surface that is completely inside the area.

Outside surface– Surface that is completely outside the area.

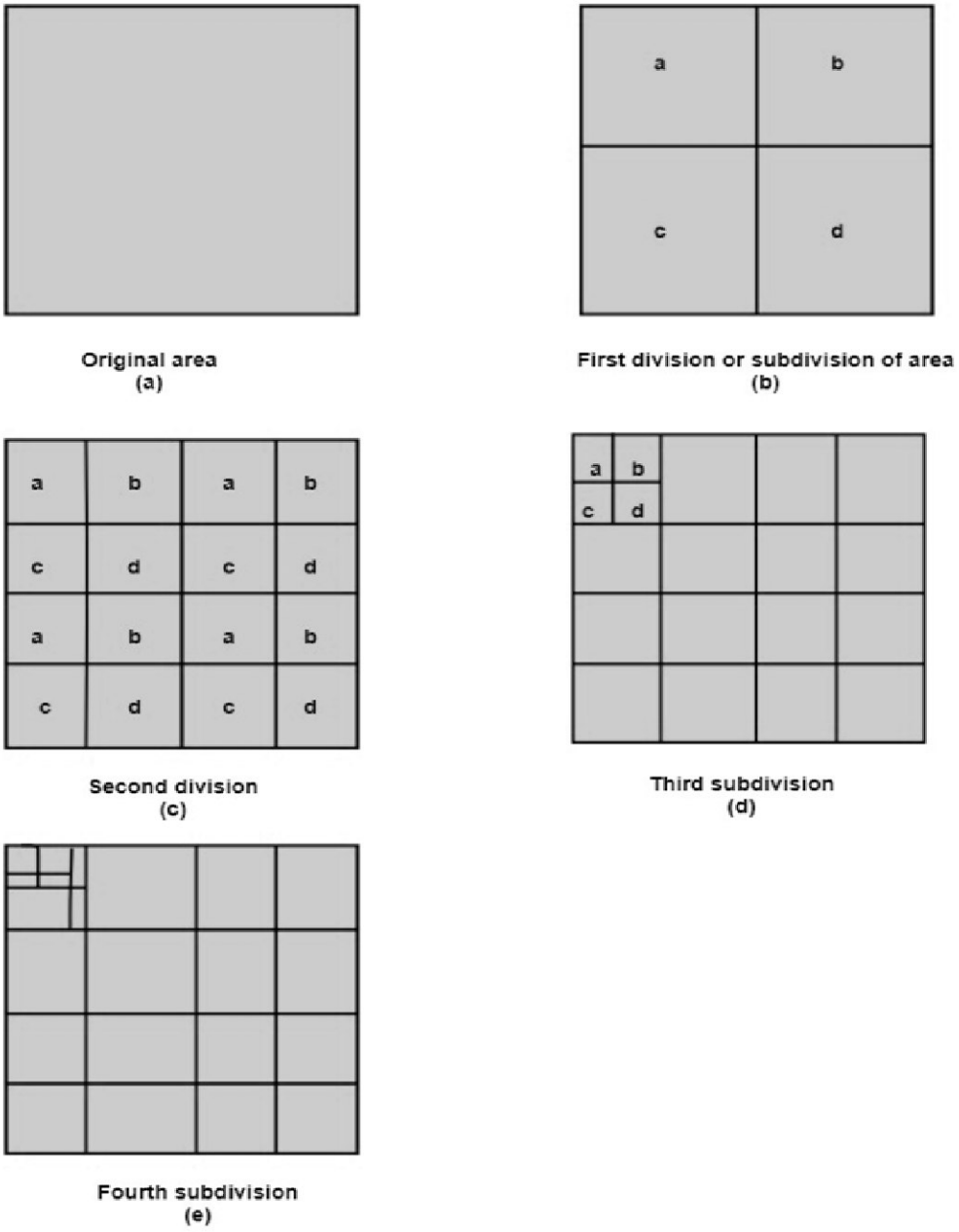


Fig. 7.3 Area Subdivision Algorithm

7.6 SUMMARY

In this unit, we discussed about those parts of the image that are covered by the non-transparent part of the image. In reality, we can not see these hidden

parts but when objects are projected onto the screen coordinate system, these hidden parts are visible.

We discussed three algorithms to remove this hidden part of the image

1. Depth Buffer (Image-Space Method)
2. Scale Line (Image Space Method)
3. Area Subdivision (Object Space Method)

Area subdivision algorithms and scan-line are best for those images where objects are expanding horizontally and/or the scenes with a fewer number of objects (about several thousand surfaces).

Z-buffer and subdivision algorithms are suitable for the scene with fewer than a few thousand surfaces.

7.7 TECHNICAL QUESTIONS

1. The surfaces that is blocked or hidden from view in a 3D scene are known as:
   a) Hidden Surface
   b) Frame Buffer
   c) Quad Tree
   d) None of these
2. The method which is based on the principle of checking the visibility point at each pixel position on the projection plane are called:
   a) Object – space method.
   b) Image space method.
   c) Both a & b
   d) None of these
3. The name of a visible surface detection algorithm are:
   a) Back face detection
   b) Back face removal
   c) Ray tracing
   d) None of these
4. For which purpose , one needs to apply natural light effects to visible surface:
   a) Fractals
   b) Quad Tree

c) Rendering

d) None of these

5. Which of them is a flexible strip that is used to produce smooth curve using a set of point:

a) Sp line

b) Scan line method

c) Depth sorting method

d) None of these

6. The problem of hidden surface are

a) Removal of hidden surface

b) Identification of hidden surface

c) Both a & b

d) None of these

7. The algorithm of hidden surface are

a) Object-space method

b) image-space method

c) Both a & b

d) None of these

8. The method which is based on the principle of comparing objects and parts of objects to each other to find which are visible and which are hidden are called

a) Object-space method

b) Image space method.

c) Both a & b

d) None of these

9. Which surface algorithm is based on perspective depth

a) Depth comparison

b) Z-buffer or depth-buffer algorithm

c) subdivision method

d) back-face removal

10. The array are used with scan line coherence algorithm are

a) For intensity value

b) For depth value

c) Both a & b

d) None of these

Answers:

1. A
2. B
3. A
4. C
5. A
6. C
7. C
8. A
9. B
10. C

UNIT-8 POLYGON RENDERING AND RAY TRACING METHODS

Structure :

8.1 INTRODUCTION

In computer graphics, Rendering means giving proper intensity at each point in a graphical object to make it look like real world object and in this unit we are covering polygon rendering. For a realistic display of a scene the lighting effects should appear naturally. An illumination model, which is sometimes called as a lighting model, is used to compute the color of an illuminated position on the surface of an object.

The shader model then determines when this color information is computed by applying the illumination model to identify the pixel colors for all positions on the scene. Different approaches are available that calculate the color for each pixel individually or interpolate between pixels in the vicinity. This chapter will introduce the basic concepts that are necessary for achieving different lighting effects. It is a method wherein the object's surfaces visible on camera are determined by incident rays or say casting rays of light from the viewer in the scene. The concept behind this casting is to incident rays from the eye, per pixel and gets an object that blocks the path of that ray. The shading of the surface is calculated by 3-D computer graphics shading models used traditionally. This type of casting is not a same as ray tracing, although it can be considered as an abridged, and considerably a faster, version of the ray tracing algorithms. The rest of the unit is organized as follows. Section 1.1 lists the objectives of the unit. Section 1.2 discusses the basics of Illumination Model. Sections 1.3 and 1.4 explain the Shading and Ray Tracing respectively. Section 1.5 captures the summary of the unit and section 1.6 ends the unit with some Technical Questions for students to work out.

8.2 OBJECTIVES

After end of this unit, you should be able to understand:

- The basics of Polygon rendering and Ray tracing methods
- Ambient Reflection, Diffuse Reflection and Specular Reflection.
- Gouraud Shading, Phong Shading.

8.3 ILLUMINATION MODEL

For determining the intensity (color) of a pixel, which results from the projection of an object (for example a polygon), illumination models are used. An illumination model describes how to compute an intensity (color) of a point within the scene depending on the incoming light from different light sources. The computation is usually done in object space.

It is used to calculate the intensity of light that we should see at a given point on the surface of an object. Lighting model is used to replicate lighting effect in rendered environment where light is approximated. Without lighting models, replicating the effects of lightning as they occur in the natural world would require more processing power than is practical for computer graphics. The purpose of lighting, or illumination model is to compute the colour of each pixel or the amount of light reflected for different surfaces on the scene. Object oriented lighting and global illumination are two main illumination models. They are different in that object oriented lighting consider every object individually, whereas global illumination shows how light interact between objects. Currently, new researchers are carried on the techniques of global illumination that replicate more accurately how light interacts with its environment.

Object oriented lighting:

Object oriented lighting, also called local illumination, is defined by mapping a single source of light to a single object. This is a fast computation technique, but often gives an incomplete approximation of how light behave in the scene in reality. It is generally approximated by summing a combination of specular, diffuse, and ambient light of some specific object. The two local illumination models are the Phong and the Blinn-Phong illumination model.

Light Sources:

It allows for different ways to introduce light into graphics scene.

Point Source (a):

It emits light from a single point in all the directions, with decreasing intensity of the light with respect to distance. Light bulb is an example of a point source.

Directional (b):

A directional source uniformly lights a scene from one direction. The intensity of light produced by it does not decrease with distance, unlike the point source, as the source is treated very far away from the scene. Sunlight is an example of a directional source.

Spotlight (c):

A spotlight produces a directed cone of light. The light becomes more intense closer to the spotlight source and to the center of the light cone. An example of a spotlight is a flashlight.

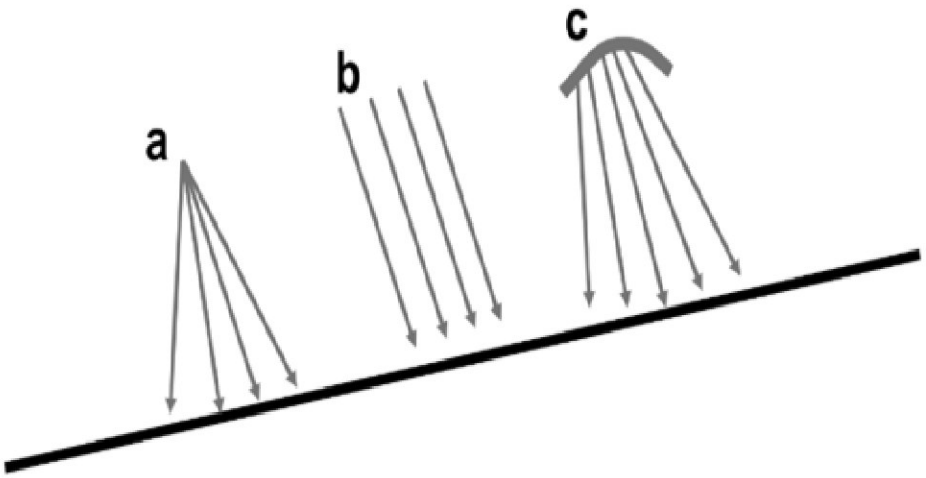


Figure 8.1

Illumination model has three major methods:

Ambient Reflection:

This illumination model only considers the reflection that is coming directly from the light source; it does not consider the secondary reflections and that's why to account a secondary reflection, light is added that distributes homogeneously through the entire 3D scene. Let's assume that there is some non-directional light in the environment (background light). Then the amount of ambient light that is incident on each object will be constant for all surfaces in the all directions. It is very simple model but not very realistic. The intensity of light of the ambient light source is distributed constantly throughout the scene. As ambient lighting is not directed, uni-colored and uni-material objects still appear in one colour. Ambient light models undirected light from an infinitely large light source, the color of an object depends on reflection coefficient that is defined by the amount of reflected light.

Diffuse Reflection:

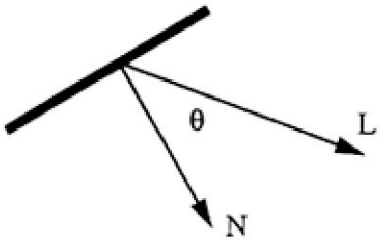
Diffuse surfaces are rough or grainy, like soil, fabric and it appears equally bright from all viewing directions. Diffuse reflection is the direct illumination of an object by an even amount of light interacting with a light-scattering surface. After light strikes an object, it is reflected as a function of the surface properties of the object as well as the angle of incoming light. This interaction is the primary contributor to the object's brightness and forms the basis for its color.

Lowest common denominator for all geometry.

$I = Idkd\cos\theta$

– Id = intensity of light

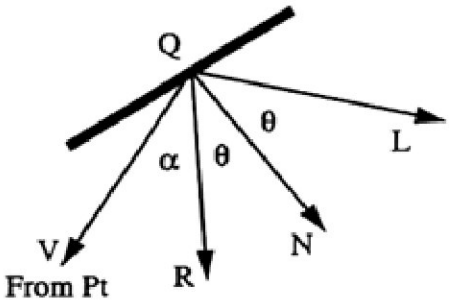
– kd = coefficient of diffuse reflection



Specular Reflection:

The specular reflection component gives objects shine and highlights it. This is distinct from mirror effects because other environment objects are not visible in this reflection. Instead, it creates bright spots on the objects based upon the intensity of the specular lighting component and the surface's specular reflection coefficient.

It models shiny and glossy surfaces (like metal, plastic) with highlights. The intensity of reflectance with reflected angle. An ideal specular surface (mirror) reflects light in one direction exclusively. The glossy objects are not ideal mirrors and reflect in the immediate vicinity of R.



$I = Iaka + Ip \quad kd \cdot \cos(\theta) + ks \cdot \cosn(\alpha)$

8.4 SHADING

Shading in general or in drawing is used to depict the range of darkness by applying dark shades for dark areas and light shades for light areas. Light patterns and shaded areas helps in creating the illusion of depth on paper.

In Computer Graphics, Shading is a process of alteration of colour of an object or a surface or a polygon in 3D scene. Alteration can be done based on certain things like angle of the surface to a light source or light sources, distance of surface from light, angle of source to the camera and material properties to create a photorealistic effect.

Shading is performed by a program called '*shader*' during the rendering process.

Shading model can be used to calculate the intensities and colors to display on the surface. This model has two main ingredients: surface's properties and

properties the of illumination falling on it. Reflectance is a property of surface which determines amount of incident light reflected and if the surface have different reflectance for different wavelengths of light, it will appear coloured.

Shading model can be divided into three parts, transparency effect, contribution from diffuse illumination (uniform illumination from all directions) and contribution from light sources.

E is a Shading term to find total energy coming from a point of an object and all three parts contribute in it. It is the energy that should generate by a display to show realistic image of an object.

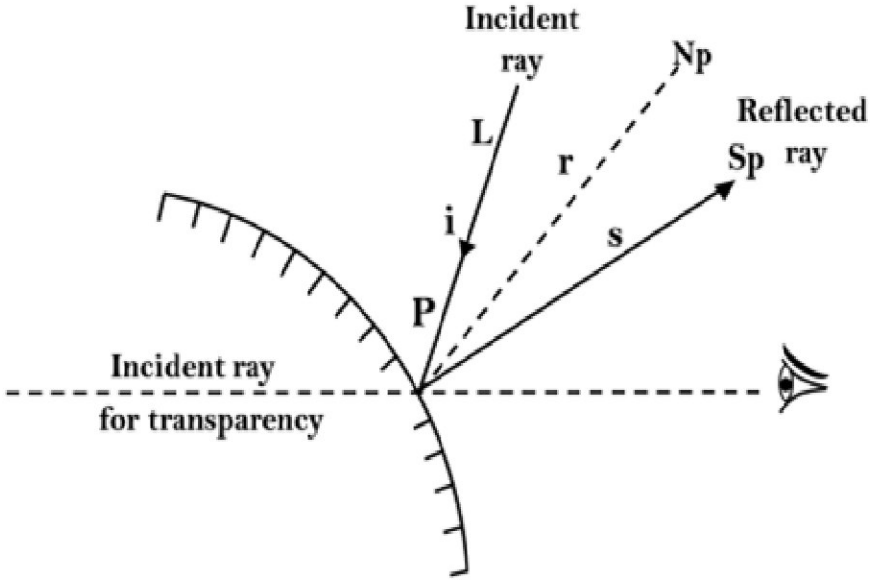


Fig.8.2

The simplest form of shading is diffuse illumination and is given as:-

$$E_{pd}=R_p\, I_d$$

Where E_{pd} is the energy coming from point P due to diffuse illumination. I_d is the diffuse illumination falling on the entire scene, and R_p is the reflectance coefficient at P.

Shading Techniques:

A *Surface Normal* is often needed for lightning computation during Shading. There are 2 main techniques of shading namely, Flat Shading and Smooth Shading.

Flat Shading:

It is one lightning calculation per polygon. Lightning is evaluated only once for each polygon that too usually for the first vertex of the polygon, assuming that all polygons are flat and surface of polygon is normal. The computed colour is used for whole polygon and makes the corners sharper. Here the colours change discontinuously at polygon borders and this technique is usually used in advanced shading techniques.

Smooth Shading:

In contrast to flat shading, here colours change pixel by pixel, resulting in smooth colour transition between two adjacent polygons. Values are usually calculated at the vertex and *bilinear interpolation* is used to calculate pixel values among the vertices of polygon.

Smooth shading is further of two types *Gouraud shading* and *Phong shading*.

Gouraud Shading:

It is one lightning calculation per vertex. It was developed by Gouraud so named after him as Gouraud Shading. This type of shading renders a polygon surface by linear interpolating intensity value across the surface. It eliminates the intensity discontinuities that occur in flat shading by coordinating each intensity values of polygon with value of adjacent polygons along the common edges.

Following steps are applied to render each polygon surface:-

1. Determine the average unit normal vector at each polygon vertex.
2. To determine vertex intensity apply illumination model to each vertex.
3. Linear interpolate the vertex intensities over the surface of the polygon.

We obtain the normal vector at each polygon vertex by averaging the surface normal of all the polygons staring that vertex.

PROBLEMS

1. Gouraud shading interpolates linearly so it can make the highlight much bigger.
2. Gouraud shading can miss highlights that occur in the middle of a polygon.
3. Inaccuracies can become too apparent due to lightning being conducted only at vertices.

Phong Shading:

It is one lightning calculation per pixel. It was developed by Phong Bui Tuong and named after him as Phong Shading. It is also known as Normal vector Interpolation shading because for rendering a polygon surface it interpolates normal vector and then apply the illumination model. It reduces the Match-band effect and also displays more realistic highlights on a surface.

Following steps are applied to render each polygon surface:-

1. Determine the average unit normal vector at each polygon vertex.
2. Interpolate the vertex normal over the polygon surface.
3. Apply an illumination model along each scan line to calculate projected pixel intensities for the surface points.

Phong shading is able to produce highlights that occur in the middle of a polygon.

Problem with Phong Shading is that the vertex normal can be incorrect when calculated as average of face normal but it can be solved by adding more polygons or by testing for angles and using different vertex normal for adjacent polygons.

8.5 RAY TRACING

Ray Tracing is one of the most successful image synthesizing technique that has a great capability to produce greatest degree of realism virtually which is much better than other rendering methods. It is based on the idea that reflection and refraction can be modelled by recursively following the path that the light takes as it bounces through the real world. Ray Tracing is one of the rendering techniques used for generating an image by tracing the path of light as pixels in a plane of the image and simulating its effects when encountered with virtual objects. When Ray Tracing was invented, the computers at that time around 1960's were slow and couldn't pace up with the highly advanced technique. But now, as computers are more powerful, and one can say computation power has improved exponentially, the ray tracing has become one of the most popular technique.

Where is it used?

It is generally used in the areas where cost to implement is not an issue and time taken to render is tolerated. It needs a very high cost and time to render the image. It is used in architecture, it helps the architects to present design proposals and design which have more realistic rendering. Ray tracing is also used on a large scale in the theatre, television lighting design and also in animation.

How Ray Tracing works?

Ray tracing works in two phases namely modelling and rendering. First the ray tracer models the scene using geometric primitives such as polygons. In this phase, the algorithm creates objects, defines the eye and the lights, and determines how each object will look like. This is the first phase which is known as modelling. In the second phase, image is created out of the resulting geometric description by applying surface characteristics to the objects. This second phase is called as rendering.

Types of Ray Tracing:

- **Forward Ray Tracing :**

  In Forward ray tracing, we follow the light particles also called photons from the light source to the object. Although forward ray tracing is accurate, it is the most inefficient because of the reason that many rays from the light source never come through the view plane and into the eye. Tracking each and every light ray from the source is a waste because very few of them will contribute to the final image as seen from the eye.

Another name of Forward Ray Tracing is Photon Tracing.

- **Backward Ray Tracing :**

  It is more efficient method of Ray Tracing. In this method, a ray is used which is created at the eye and passes through the view plane into the world. The first object this ray will hit, will be the object visible from that point of the view plane. It also allows the light ray to bounce around and using this it can figure out the shading and colouring of that viewpoint in the real world the display it appropriately.

The drawback of this ray tracing technique is that it only considers the light rays coming through the view plane in the eye and is contributing in final image.

Hybrid Ray Tracing:

After seeing advantages and drawbacks of both forward ray tracing and backward ray tracing methods, the recent researchers have come up with a hybrid solution. In this hybrid Ray Tracing technique, forward ray tracing is performed only at a certain level. Then the algorithm will record the data and will perform backward ray tracing.

The final result will be a contribution of both the Forward Ray Tracing and the Backward Ray Tracing.

Hybrid Ray Tracing

Shadowing:

The areas where the light propagates is easy to see but the regions where there is ray of light that has no direct path is difficult to model, for instance area behind a surface or under an object. The ray tracing model incorporates these realistic shadows into the final resultant image.

Reflection:

When there is a very highly reflective surface into a scene, we need to make it look different than the normal object on the scene and it also adds more illumination into the scene. The illumination effects on such highly reflective object must be different from the other normal objects onto the scene.

Transparent objects produce both reflected and transmitted light. The requirement here is to model the transparent objects such as glass.

When we view the glass from one side we should be able to see the light transmitted from the objects behind the glass and should be able to incorporate this in our resultant image. This can be done by using the Ray Tracing.

Refraction:

Refraction has to be incorporated in the model to give realistic transparency in our final image. When we have an object that can refract the light ray or a part of light is reflected and the other is refracted, in such cases the light is bend and has to be modelled to give a realistic effect

8.6 SUMMARY

In this unit we covered several topics related to the Polygon rendering and ray tracing algorithms which are used to represent the image on the 3-D surface and use to perform complex operations in a less time using the advance Polygon rendering then at last we had learn about the ray tracing algorithms which are predominantly used to simulate the reflection and shadows on the surface.

8.7 TECHNICAL QUESTIONS

This section consists of some Technical Questions related to the unit:

1. What is Polygon Rendering how it is used for real world 3-D objects?
2. List the difference between ambient reflection and diffuse reflection?
3. Why specular reflection is termed as mirror-like reflection?
4. What is Gouraud shading? How it eliminates the intensity discontinuity that occurs in flat shading?
5. What is Phong Shading? What is the advantage of applying the illumination model at each surface point?
6. List the advantages and disadvantages of the Ray tracing.

Multiple Choice Questions:

1. For which purpose, one needs to apply natural light effects to visible surface.
    a) Fractals
    b) Quad-tree
    c) Rendering
    d) None of these
2. The basic ray tracing algorithm provides:
    a) Transparency
    b) Visible-surface detection
    c) Shadow effect, multiple light source illumination
    d) All of these
3. Ray-tracing is an extension of
    a) Ray calling
    b) Ray casting
    c) Ray sampling

d) None of these

4. A fast and simple method for rendering an object with polygon surface is

   a) Constant-intensity shading

   b) Flat Shading

   c) Both a & b

   d) None of these

5. A surface rendering algorithm:

   a) Is used to calculate the intensity of light that we should see at a given point on the surface of an object.

   b) Uses the intensity calculations to determine the light intensity

   c) Scattered light from a rough surface.

   d) Light source creating highlights.

6. Diffuse reflection is :

   a) Is used to calculate the intensity of light that we should see at a given point on the surface of an object

   b) Uses the intensity calculations to determine the light intensity

   c) Scattered light from a rough surface.

   d) Light source creating highlights.

7. A shading model is:

   a) Is used to calculate the intensity of light that we should see at a given point on the surface of an object

   b) Uses the intensity calculations to determine the light intensity

   c) Scattered light from a rough surface

   d) Light source creating highlights

8. In the equation j= (1-kt) Irefl + ktItrans :: Itrans refers to:

   a) Intensity

   b) Transmitted intensity

   c) Reflected intensity

   d) Transparency coefficient

9. In the equation j= (1-kt) Irefl + ktItrans :: kt refers to:

   a) Intensity

   b) Transmitted intensity

c) Reflected intensity

d) Transsparency coefficient

10. In the equation j= (1-kt) Irefl + ktItrans :: Irefl refers to:

a) Intensity

b) Transmitted intensity

c) Reflected intensity

d) Transparency coefficient

Answers:

1. C
2. D
3. C
4. C
5. B
6. C
7. A
8. B
9. D
10. C

ROUGH WORK