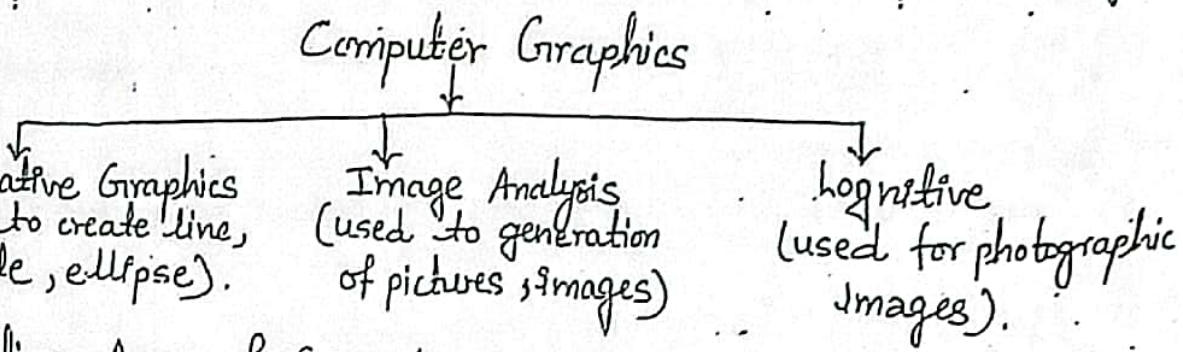
Unit-1

Introduction of Computer Graphics

Introduction: The process of transforming and presenting objects or information in visual form is called Computer Graphics. It is the creation of pictures with the help of a computer. Computer graphics maybe a business graph, drawings like line, circle or any other shape, animations etc.

In today life, computer graphics has now become a common element in user interfaces, T.V, ~~computer~~ commercial motion pictures etc. In computer graphics, two or three-dimensional pictures can be created that are used for research. It is also used used in various fields like engineering, mathematical model etc.



Application Areas of Computer Graphics:

Computer graphics deals with creation, manipulation and storage of different type of images and objects. Some of the applications of computer graphics are as follows:-

1) Computer Art → Using computer graphics we can create fine and commercial art which include animation packages, paint packages. These packages provide facilities for designing object shapes and specifying object motion. Cartoon drawing, paintings, logo design can also be done.

2) Computer Aided Drawing (CAD) → Designing of buildings, automobile, aircraft is done with the help of computer aided drawing. This helps in providing minute details to the drawing and producing more accurate and sharp drawings with better specification.

iii) Presentation Graphics → For the prepration of reports or summarising the financial, statistical, mathematical, scientific, economic data for research reports, managerial reports, moreover creation of bar graphs, pie charts, time chart, can be done using the tools present in computer graphics.

iv) Entertainment → Computer graphics finds a major part of its utility in the movie industry and game industry. Used for creating motion pictures, music video, television shows, cartoon animation films.

v) Education → Computer generated models are ~~extremly~~ useful for teaching huge number of concepts in an easy to understand and learn manner. Using computer graphics many educational models can be created through which more interest can be generated among the students regarding the subject.

vi) Training → Specialized system for traning like simulators can be used for training the candidates in a way that can be grasped in a short span of time with better understanding.

vii) Visualisation → Data visualisation helps in finding insights of the data, to check and study the behaviour of processes around us we need appropriate visualisation which can be achieved through proper usage of computer graphics.

viii) Image Processing → Various kinds of photographs or images require editing in order to be used in different places. Processing of existing images into refined ones for better interpretation is one of the many applications of computer graphics.

ix) Graphical User Interface (GUI) → The uses of pictures, images, icons, pop-up menus, graphical objects helps in creating a user friendly environment where working is easy and pleasent, using computer graphics we can create such an atmosphere where everything can be automated and anyone can get the desired action performed in an easy fashion.

⊛ Graphics Hardware ⊛

Classification of computer graphics :-

- **Interactive/active/online graphics** — Person involves himself in the graphics i.e. interacts with it e.g. games, chatting etc.)
- **Non interactive/passive/offline graphics** — (Person is not involved e.g. movie, TVs etc.)

⊛ What is pixel ?

Ans :- Smallest addressible screen element is known as pixel.

⊛ What is resolution?

Ans. The maximum number of points that can be displayed without overlap is refered to as resolution.

OR

The number of points per centimetre that can be plotted horizontally and vertically is called resolution.

for eg. 1024 × 640.

Types:

ⓐ Image resolution → Distance from one pixel to next pixel is called as image resolution. It is also called spacing.

ⓑ Screen resolution → Number of pixels in the horizontal and vertical directions.

Some important terms:

Aspect ratio → The ratio of vertical points to the horizontal points necessary to produce equal-length lines in both directions on screen is aspect ratio. For example: If 1024 × 640 is divided as $\frac{1024}{640}$ and reduced into lowest terms then we get 8:5.

Note: Aspect ratio (AR) = $\frac{\text{Width of image}}{\text{Height of image}}$

Frame buffer → The memory where the picture definition is stored is known as frame buffer. It is also called as refresh buffer. This area holds the set of intensity of values for all screen points.

Refresh rate → The refresh rate is the number of times a screen displays an image is repainted or refreshed per second. The refresh rate is expressed in hertz so a refresh rate of 75 means the image is refreshed 75 times in a second. The refresh rate for each display depends on the video card used.

Note - An interlaced display is a cathode-ray tube (CRT).

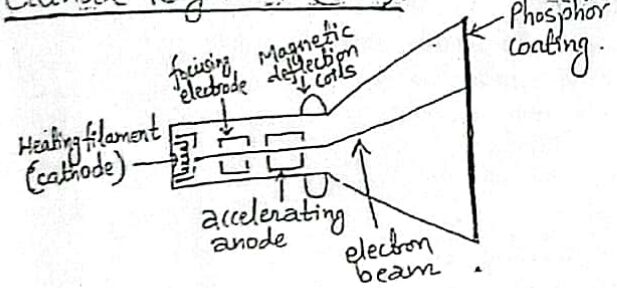
Display Technology:

Graphical displays are generally of following two types:

(a) **Vector displays** → Vector displays generally display lines, specified by their endpoints. Vector display systems operate by direct control of the electron beam of a cathode ray tube (CRT). Vector displays have certain advantages such as the absence of aliasing. Aliasing is the jagged appearance of primitives as displayed on a raster device.

(b) **Raster displays** → Raster display typically have an array of addressable dots, which can be individually set to a particular color or intensity. Raster displays can be implemented by several technologies. Currently the most popular is the cathode ray tube, which can implement both raster and vector displays.

Cathode Ray Tube (CRT):



- The electron gun emits a beam of electrons (cathode rays).
- The electron beam passes through focusing and deflection systems that direct it towards specified positions on the phosphor-coated screen.
- When the beam hits the screen, the phosphor emits a small spot of light at each position contacted by the electron beam.
- It redraws the picture by directing the electron beam back over the same screen points quickly.

The two basic techniques for producing color displays with a CRT are the beam-penetration method and the shadow-mask method. These use ~~mar~~ three colours Red, Green and Blue (RCB) and their combinations.

Types of CRTs:

(a) **Raster-Scan Display:** - In a raster scan system, the electron beam is swept across the screen, one row at a time from top to bottom. As the electron beam moves across each row, the beam intensity is turned on and off to create a pattern of illuminated spots. Picture definition is stored in memory area called Refresh Buffer or Frame Buffer. Stored intensity values are painted on the screen one row at a time as shown below:

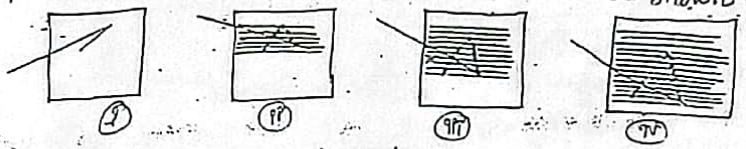


fig. A raster-scan system displaying A.

Refreshing on Raster-Scan display is carried out at the rate of 60 or higher frames per second. Most of display devices are based on this technology. For example, CRT, color CRT, LCD and LED etc.

Horizontal retrace / vertical retrace → Returning of electron beam from right end to left end after refreshing each scan line

is horizontal retrace. At the end of each frame, the electron beam returns to the top left corner to begin next frame called vertical retrace.

Architecture of Raster-Scan System:

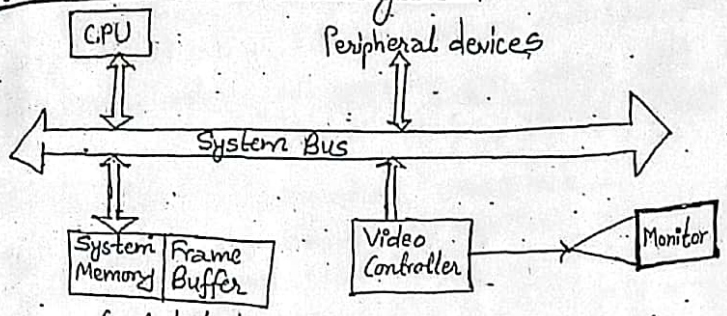


fig. Architecture of simple Raster-scan system.

Raster-scan system consists of several processing units. CPU is the main processing ~~system~~ unit of computer system. Architecture of Raster-scan consit of system bus that helps to access peripheral devices. It consist of video controller which is used to control the operation of display device. A fixed area of system memory is reserved for the frame buffer. The contents of frame buffer are used to control the CRT beam's intensity or colour.

Advantages:

- Show Realistic pictures
- Million Unique hues can be performed
- Shadow scenes are conceivable (i.e capable of being imagined).

(means colour)

Disadvantages:

- Low Resolution
- Electron beam coordinated to whole screen not exclusively to those parts of screen where picture is drawn so long when drawn picture is longer than whole screen.
- Expensive
  high memory is required

(b) Random-Scan (Vector) display System:

In this technique, the electron beam is directed only to the part of the screen where the picture is to be drawn rather than scanning from left to right and top to bottom as in raster scan. It is also called vector display or calligraphic display. Picture definition is stored as a set of line-drawing commands in an area of memory referred to as the refresh display file. Random-scan displays are designed to draw all the componend lines of a picture 30 to 60 times each second.

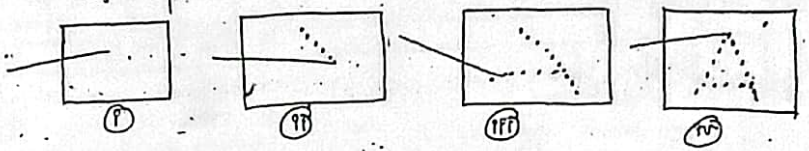


fig. Random-scan displaying A

The refresh rate of vector display depends upon the no. of lines to be displayed for any image. It is designed for drawing all component lines 30 to 60 times per second. Plotter is the best example of this system.

Architecture of Random Scan System:

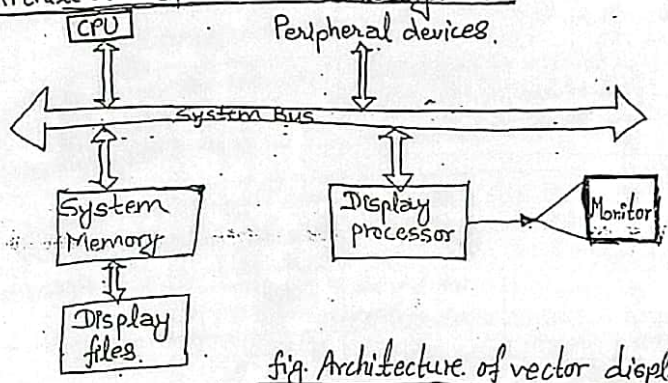


fig. Architecture of vector display system.

Random display system consists of additional processing unit along with CPU which is called the display processor. Picture definition is stored as a set of line drawing commands in an area of memory called display list. This display list is then accessed by the display processor to create an image.

Advantages:-

i) High resolution.
ii) Produce smooth line drawings.

Disadvantages:-

i) Expensive
ii) Just does wire frame.
iii) Complex scene cause visible flicker.

⊛. Differences or Comparision between Raster and Random display systems

Basis of Comparision.	Raster Scan System	Random Scan System.
Resolution	It has poor resolution because picture definition is stored as an intensity value.	It has high resolution because it stores picture definition as a set of line commands.
Electron Beam.	Electron beam is directed from top to bottom and one row at a time on screen.	Electron beam is directed to only that part of screen where picture is required to be drawn, one line at a time.
Cost	It is less expensive than Random Scan System.	It is more expensive than raster scan system.
Refresh Rate	The refresh rate is independant of picture complexity. Refresh rate is 60 to 80 frames per second.	Refresh rate depends on the number of lines to be displayed i.e, 30 to 60 times per second.
Line drawing	Zig-Zag lines are produced because plotted values are discrete.	Smooth line is produced because directly the line path is followed by electron beam.
Image Drawing.	It uses pixels along scan lines for drawing an image.	It is designed for line drawing applications and uses various mathematical functions to draw.

⊛. Input Devices: (Describe /Read Yourself)

i) Mouse
ii) Keyboard
iii) Joy stick
iv) Light pen
v) Track ball
vi) Scanner
vii) Microphone.
viii) Bar Code Reader etc.

⊛. Output Devices:

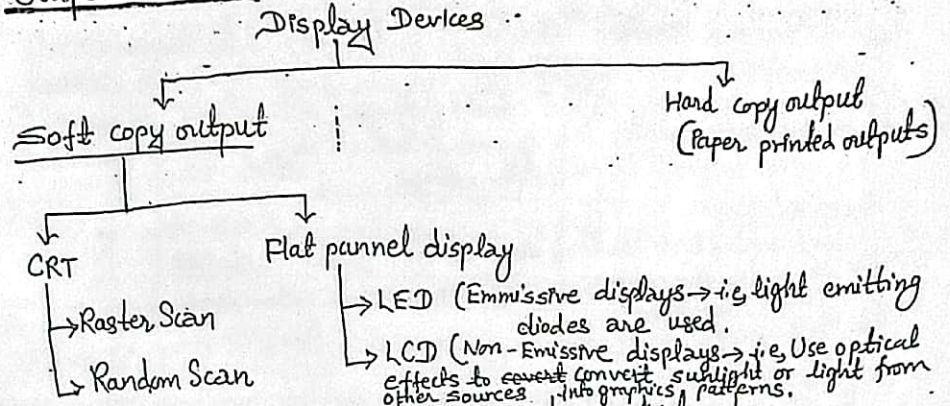


fig. Display devices in hierarchiel order.

⊛. Differences between CRT and LCD displays:-

Topic	CRT	LCD
Size	CRT monitors are thicker in size.	LCD monitors are thinner than CRT monitors.
Weight.	A CRT monitor has weight 40 pounds or more depending upon its size.	LCD monitors has pretty light weight as 8 to 10 pounds.
Price	Because of popularity of LCD monitors CRT has became cheap now.	LCD monitors are newer technology so they are expensive than CRT.
Picture	CRT has low quality picture.	LCD has high quality picture.
Viewing angle	CRT monitors has better viewing angle.	LCD monitors can not be viewed from different angles as CRT.

⊛. Color CRT: (Part of Video display Devices / Output Devices)

There are two popular techniques for producing color displays with CRT which are as follows:-

ⓐ Beam Penetration method:

The beam-penetration method for displaying color pictures has been used with random-scan monitors. Two layers of phosphor, usually red and green, are coated onto the inside of the CRT screen, and the displayed color depends upon how far the electron beam penetrates into the phosphor layers. A beam of slow electrons excites only the outer red layer. A beam of very fast electrons penetrates through the red layer and excites the inner green layer. At intermediate beam speeds, combinations of red and green light are emitted to show two additional colours, orange and yellow. The speed of electrons and hence the screen colour at any point, is controlled by the beam-acceleration voltage.

Advantages

→ Half cost as compared to that of shadow mask CRT.
→ It is an inexpensive way to produce color in random scan monitor.
→ Its resolution is better.

Disadvantages

→ Time consuming during switching of colours.
→ It consumes significant amount of accelerating potential in order to switch color.

ⓑ Shadow Mask method:

Shadow-mask methods are commonly used in raster scan system (including color TV) because they produce a much wider range of colors than the beam-penetration method. A shadow-mask CRT has three phosphor color dots at each pixel position. One phosphor dot emits a red light, another emits a green light and a third emits a blue light. This type of CRT has three electron guns, one for each color dot, and a shadow-mask grid just behind the phosphor-coated screen, with tiny holes in triangular shape

Advantages:

→ Produces much wider range of colors than beam penetration method.
→ It produces realistic images.

Disadvantages:

→ It is very difficult to coverage all three beams on same hole.
→ They have poor resolution.

⊛. Differences between Beam penetration method and shadow mask method.

Basis for Difference	Beam Penetration Method	Shadow Mask Method.
Where Used	It is used with Random Scan System to display color.	It is used with Raster Scan System to display color.
Colors	It can displays only four colors i.e, Red, Green, Orange and Yellow.	It can display millions of colours.
Cost	It is less expensive as compared to shadow mask.	It is more expensive than beam penetration.
Picture Quality.	Picture quality is poor.	Picture quality is high.
Resolution	It gives high resolution	It gives low resolution.

⊛ Graphics Software:

ⓐ General programming packages → It contains graphics functions used with high level programming languages like C, FORTRAN, JAVA etc. Example:- Open GL (Graphics Library)

ⓑ Special-purpose application packages → It is especially designed for particular applications. For example-CAD applications.

⊛ Software Standards:-

Software standards helps to make portability of software & rewriting code is not required. It can also be used in different implementations and applications. Following are some software standards.

(a) Graphics Kernal System (GKS) → GKS is the first graphics software standard adopted by the international standards organization (ISO). It was originally designed as a 2D graphics package. It includes various types of methods, reserved words.

Advantages

→ It provides improved algorithm.
→ It makes system portable.
→ Rewriting of code is not required.

(b) PHIGS (Programmer's Hierarchical Interactive Graphics Standard).

It is the extension of GKS which provides 3D graphics package. It includes additional functions for object modeling, Color specification, Surface rendering and picture manipulation.

(c) PHIGS+ :

It is the extension of earlier PHIGS. 3D surface shading capabilities are added to PHIGS+.

Unit-2

Scan Conversion Algorithm

The process in which the object is represented as the collection of discrete pixels is called scan conversion. It includes the concepts that are required for the understanding of two dimensional graphics. The graphics ~~used in~~ objects used in scan conversion are in continious but the pixels used are in discrete. Each pixel can have either on or off state.

Point, line, sector, arc, rectangle, ellipse, characters etc. are the examples of objects which can be scan converted. Different algorithms are used ~~to scan~~ for scan conversion of these types of objects which are called scan conversion algorithms.

⊛ Advantage of developing algorithms for scan conversion:

- Algorithms can generate graphics objects at a faster rate.
- Using algorithms memory can be used efficiently.
- Algorithms can develop a higher level of graphical objects.

⊛ Line Drawing Algorithm:

A line drawing algorithm is a graphical algorithm for approximating a line segment on discrete graphical media. There are mainly three most widely used line drawing algorithms as follows:

1. Direct use of line equation.
2. Digital Differential Analyzer Algorithm (DDA)
3. Bresenham line Drawing Algorithm (BSA)

Here we study about only two DDA algorithm and BSA algorithm.

1.# Digital Differential Analyzer Algorithm (DDA):

In any 2-Dimensional plane if we connect any two points (x_0, y_0) and (x_1, y_1) we get a line segment But in the case of computer graphics we can not directly join any two coordinate points, for that we should calculate intermediate points coordinate using a basic algorithm called DDA.

Working details of DDA Algorithm:

Step1:- Get the input of two end points (x_0, y_0) and (x_1, y_1).

Step2:- Calculate the difference between two end points as:-

$$dx = x_1 - x_0$$
$$dy = y_1 - y_0$$

Step3:- Based on the calculated difference in step-2, now we need to identify the number of steps to put pixel as follows:-

If $dx > dy$, then we need more steps in x co-ordinate; otherwise in y coordinate.

i.e. if (absolute(dx) > absolute(dy)), then steps = absolute(dx) else steps = absolute(dy).

(also we can do with $m<1$ where $m = \frac{dy}{dx}$)

Step4:- We calculate the increament in x coordinate and y coordinate as follows:-

$$x_{increament} = \frac{dx}{steps\ (float)}$$

$$y_{increament} = \frac{dy}{steps\ (float)}$$

Step5:- Finally we put the pixel by successfully incrementing x and y co-ordinates accordingly and complete the drawing of line.

```
for(int i=0; i<steps; i++){
    x = x + x_increament
    y = y + y_increament
    putpixel(Round(x), Round(y))
}
```

Example 1 (For m<1 OR dx>dy)

Using DDA plot (5,4) to (12,7).

Soln

Given:-

$(x_0, y_0) = (5, 4)$

$(x_1, y_1) = (12, 7)$

$dx = x_1 - x_0 = 12 - 5 = 7$

$dy = y_1 - y_0 = 7 - 4 = 3$

Since $dx > dy$ OR, Slope $m = \frac{dy}{dx} = \frac{3}{7}$ i.e. <1

So, no. of steps = absolute(dx) $= |7| = 7$

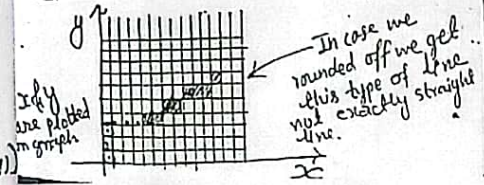
Now,

$x_{increment} = \frac{dx}{steps} = \frac{7}{7} = 1$

$y_{increment} = \frac{dy}{steps} = \frac{3}{7} = 0.4$

steps	x	y	y (Rounded off)
0	5	4	4
1	6	4.4	4
2	7	4.8	5
3	8	5.2	5
4	9	5.6	6
5	10	6	6
6	11	6.4	6
7	12	6.8	7

Note:- If x or y comes a fraction we write as it is but if m is decimal we round off them.



Example 2 (for m>1 OR dx<dy)

Using DDA plot line (5,7) to (10,15).

Soln

Given, $(x_0, y_0) = (5, 7)$

$(x_1, y_1) = (10, 15)$.

$dx = x_1 - x_0 = 10 - 5 = 5$

$dy = y_1 - y_0 = 15 - 7 = 8$

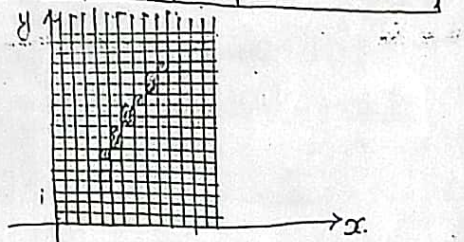
Now, $m = \frac{8}{5}$ i.e. >1

$\therefore$ Steps $= |8| = 8$

$x_{increament} = \frac{dx}{steps} = \frac{5}{8} = 0.6$

$y_{increament} = \frac{dy}{steps} = \frac{8}{8} = 1$

steps	x	y	x (Rounded off)
0	5	7	5
1	5.6	8	6
2	6.2	9	6
3	6.8	10	7
4	7.4	11	7
5	8	12	8
6	8.6	13	9
7	9.2	14	9
8	9.8	15	10



Note:- If $dx = dy$ i.e. $m = 1$ then, steps $= |dx|$ or we can take $|dy|$ & We proceed the steps as we did for $m<1$ and $m>1$.

2. Bresenham Line Drawing Algorithm (BSA):

There are two problems arised when using DDA algorithm,

1) Lines may not be smooth since we take round off values if floating points are seen in co-ordinates.
2) It takes floating points which takes extra time for calculation which makes DDA algorithm slower.

For to solve these two problems new algorithm was introduced called Bresenham line drawing algorithm.

Working details of BSA Algorithm (For positive slope)

Slope $= \frac{y_1 - y_0}{x_1 - x_0}$

Step1: Get the input of two endpoints (x_0, y_0) and (x_1, y_1).

Step2: Calculate the difference between two endpoints as:

Δx = absolute$(x_1 - x_0)$

Δy = absolute$(y_1 - y_0)$

Step3: Calculate initial decision parameter as:

$p_k = 2\Delta y - \Delta x$

Step4: If $x_1 > x_0$ then,

Set $x = x_0$;
Set $y = y_0$;
Set $x_{end} = x_1$;

Otherwise,

Set $x = x_1$;
Set $y = y_1$;
Set $x_{end} = x_0$;

Step5: Draw pixel at (x, y).

Step6: While $(x < x_{end})$ {

x++;

if $p_k < 0$ then
$p_{k+1} = p_k + 2\Delta y$

(i.e for $p_k \geq 0$) otherwise
y++;
$p_{k+1} = p_k + 2\Delta y - 2\Delta x$

Draw pixel at (x, y).

Example:

Q. Plot the line from the point (10, 15) to (15, 18) using BSA.

Soln

Given, $(x_0, y_0) = (10, 15)$
$(x_1, y_1) = (15, 18)$

Here, we calculate Δy and Δx as:

$\Delta x = |x_1 - x_0|$
$= |15 - 10|$
$= 5$

$\Delta y = |y_1 - y_0|$
$= |18 - 15|$
$= 3$

Now we calculate initial decision parameter as:

$p_k = 2\Delta y - \Delta x$
$= 2 \times 3 - 5$
$= 1$

since $x_1 > x_0$
So, $x = 10$;
$y = 15$;
$x_{end} = 15$;

We know that $p_k \geq 0$ so, using condition,

$p_{k+1} = p_k + 2\Delta y - 2\Delta x$
$= 1 + 2 \times 3 - 2 \times 5$
$= -3$

Continuing this process upto x_{end} table as follows:-

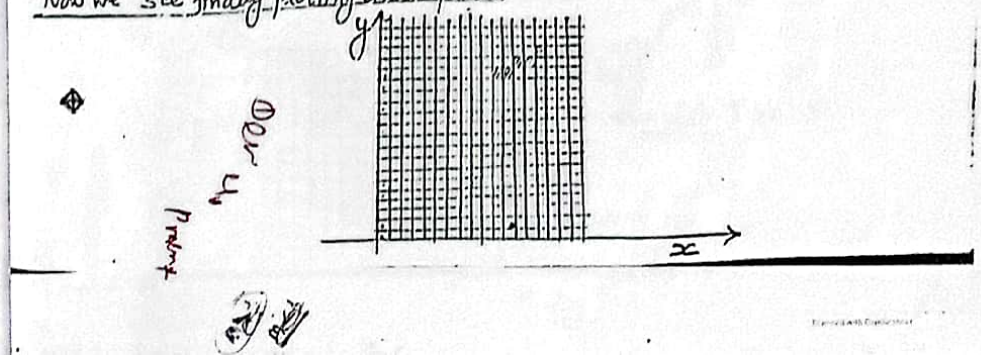
since $p_k > 0$, $p_{k+1} = p_k + 2\Delta y - 2\Delta x$
for $p_k < 0$, $p_{k+1} = p_k + 2\Delta y$

k	x	y	p_{k+1}	Pixel Points
0	10	15	1	(11, 16)
1	11	16	-3	(12, 16)
2	12	16	3	(13, 17)
3	13	17	-1	(14, 17)
4	14	17	5	(15, 18)

x always increases by one steps only while p_{k+1} is greater or equal to zero.

We can calculate step by step drawing algorithm or using table we can use either any one method.

Now we see finally plotting these points:-



Working details of BSA Algorithm (for negative slope questions):

Step1: Get the input of two end points (x_0, y_0) and (x_1, y_1).

Step2: Calculate the difference between two endpoints as:

$\Delta x = \text{absolute}(x_1 - x_0)$.

$\Delta y = \text{absolute}(y_1 - y_0)$.

Step3: Calculate initial decision parameter.

$P_k = 2\Delta y - \Delta x$.

Step4: for (i=1 to Δx){ Set pixel (x_s, y_s)

while ($P_k > 0$)

$y_0 = y_0 - 1$;

$P_k = P_k - 2\Delta x$.

end while, ($P_k > 0$)

$x_0 = x_0 + 1$

$P_k = P_k + 2\Delta y$

i++;

}

Set pixel (x_0, y_0).

Imp Example: Plot a line from (6,12) to (10,5) using BSA algorithm.

Sol^n

$(x_0, y_0) = (6,12)$

$(x_1, y_1) = (10,5)$

Now we calculate Δx and Δy as follows:-

Δx = absolute (10-6)

$= 4$.

Δy = absolute (5-12)

$= 7$

Now, initial decision parameter is,

$P_k = 2\Delta y - \Delta x$

$= 2 \times 7 - 4$

$= 10$.

for (i=1 to 4) using this condition we proceed as follows:-

for i=1 Set pixel (6,12) while (10>0)

$y_0 = 12 - 1$

$= 11$

$P_k = 10 - 2 \times 4$

$= 2$

while (2>0)

$y_0 = 11 - 1$

$= 10$

$P_k = 2 - 2 \times 4$

$= -6$

we input while upto >0. - since <0 so we go to end while

end while

$x_0 = x_0 + 1$

$= 6 + 1$

$= 7$

$P_k = P_k + 2\Delta y$

$= -6 + 2 \times 7$

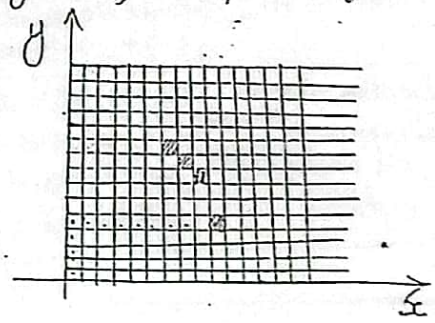
$= 8$

We can use similarly for i = 2,3 & 4 without making table step by step also

Now, continuing this process in table:-

i	x_0	y_0	P_k	Points
1	6	12	10	(7,10)
2	7	10	8	(8,9)
3	8	9	14	(9,7)
4	9	7	12	(10,5)

Now, finally plotting these points on graph



⊛ Differences between DDA line drawing algorithm and Bresenham line drawing algorithm:

Basis for difference	DDA line drawing algorithm	Bresenham line drawing algorithm.
Arithmetic	DDA algorithm uses floating points. i.e, Real Arithmetic.	Bresenham algorithm uses fixed points. i.e, Integer Arithmetic.
Operations	DDA algorithm uses multiplication and division in its operations.	Bresenham algorithm uses only subtraction and addition in its operations.
Speed.	It is slower than Bresenham algorithm because it uses floating points.	It is faster than DDA algorithm because it uses only integer arithmetic.
Accuracy & Efficiency	DDA algorithm is not as accurate and efficient as Bresenham algorithm.	Bresenham algorithm is more efficient and much accurate than DDA algorithm.
Drawing	DDA algorithm can draw circles and curves but not accurate as Bresenham algorithm.	Bresenham algorithm can draw circles and curves with much more accuracy than DDA algorithm.
Round off	DDA algorithm round off the coordinates to integer that is nearest to the line.	Bresenham algorithm does not round off but takes the incremental value in its operation.
Expensive.	DDA algorithm uses an enormous number of floating-point multiplications so it is expensive.	Bresenham algorithm is less expensive than DDA algorithm as it uses only addition and subtraction.

3. Mid Point Circle Algorithm:

Algorithm:

Step1: Start

Step2: Input radius r and circle centre (h,k).

Step3: Initialize $x=0, y=r$ and $p_0 = 1-r$

Step4: While $(x \leq y)$

```
//plot 8 points as
Plot pixel (x+h, y+k);
Plot pixel (-x+h, y+k);
Plot pixel (x+h, -y+k);
Plot pixel (-x+h, -y+k);
Plot pixel (y+h, x+k);
Plot pixel (-y+h, x+k);
Plot pixel (y+h, -x+k);
Plot pixel (-y+h, -x+k);
```

[x & y are coordinates
r → radius
p → decision parameter.]

Step5: If $p_0 < 0$ then,

```
Set, p0 = (p0+2x+1)
Set, x = x+1;
Otherwise,
Set, p0 = p0+2x+1-2y
Set, y = y-1;   and set, x=x+1;
```

Step6: End while loop.

Step7: Stop.

Example 1: Calculate the points to draw a circle having radius 10 and centre at (0,0).

Solution:

we have r=10.

centre (h,k) = (0,0)

Set x=0, y=r=10

Now, first pixel to be drawn = (0, 10)

Initial decision parameter (p_0) = 1−r
= 1−10
= −9

Now, points on the octant are given by,

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

Now, using 8 point symmetry we get all points on the circle as below:-

(x,y)	(0,10)	(1,10)	(2,10)	(3,10)	(4,9)	(5,9)	(6,8)	(7,7)
(-x,y)	(0,10)	(-1,10)	(-2,10)	(-3,10)	(-4,9)	(-5,9)	(-6,8)	(-7,7)
(x,-y)	(0,-10)	(1,-10)	(2,-10)	(3,-10)	(4,-9)	(5,-9)	(6,-8)	(7,-7)
(-x,-y)	(0,-10)	(-1,-10)	(-2,-10)	(-3,-10)	(-4,-9)	(-5,-9)	(-6,-8)	(-7,-7)
(y,x)	(10,0)	(10,1)	(10,2)	(10,3)	(9,4)	(9,5)	(8,6)	(7,7)
(-y,x)	(-10,0)	(-10,1)	(-10,2)	(-10,3)	(-9,4)	(-9,5)	(-8,6)	(-7,7)
(y,-x)	(10,0)	(10,-1)	(10,-2)	(10,-3)	(9,-4)	(9,-5)	(8,-6)	(7,-7)
(-y,-x)	(-10,0)	(-10,-1)	(-10,-2)	(-10,-3)	(-9,-4)	(-9,-5)	(-8,-6)	(-7,-7)

Now plotting these points in graph we get circle.

Example 2:- Digitize a circle $(x-2)^2+(y-3)^2=25$

Solution;

Given, $(x-2)^2+(y-3)^2=25$

Comparing with general equation of circle $(x-h)^2+(y-k)^2=r^2$

$\therefore$ we get, centre $(h,k)=(2,3)$

radius $r=5$

First pixel $=(0,5)$ $(\because$ Set $x=0, y=r)$.

Initial decision parameter $P_0=1-r$

$=1-5$

$=-4$

Now points on the octant are given by,

k	P_k	(x_{k+1}, y_{k+1})	$2x_{k+1}$	$2y_{k+1}$	Actual pixel
1	-4	(1,5)	2	10	(3,8)
2	-1	(2,5)	4	10	(4,8)
3	4	(3,4)	6	8	(5,7)
4	3	(4,3)	8	6	(6,6)
5	6	(5,2)	10	4	(7,5)
6	13	(6,1)	12	2	(8,4)
7	24	(7,0)	14	0	(9,3)

Centre (0,0) [illegible] Centre (0,0) [illegible]
We have centre (2,3) in this question so we add (2,3) to each pixel to get actual pixel.

Now using 8 point symmetry we get all points on the circle as we did in example 1 & plotting these points on graph we get required circle.

Midpoint - Ellipse Algorithm

Algorithm

Midpoint - Ellipse(a,b)

① $x=0, y=b$

② $d_1=b^2+\frac{a^2}{4}-a^2b$

For Region 1

③ while $(a^2y \geq b^2x)$

Plot (x,y)

if $(d_1 \leq 0)$

$d_1=d_1+b^2(2x+3)$

$x=x+1$

else

$d_1=d_1+b^2(2x+3)+a^2(2-2y)$

$x=x+1$

$y=y-1$

For Region 2

④ $d_2=(x+0.5)^2b^2+(y-1)^2a^2-a^2b^2$

⑤ while $(y \geq 0)$

Plot (x,y)

if $(d_2 \leq 0)$

$d_2=d_2+b^2(2x+2)+a^2(3-2y)$

$x=x+1$

$y=y-1$

else

$dx=d_2+(3-2y)a^2$

$y=y-1$

Example:- Input ellipse parameters $r_x = 8$ and $r_y = 6$ the mid-point ellipse algorithm by determining raster position along the ellipse path in the first quadrant. Initial values and increments for the decision parameter calculates are:-

$2r_y^2 x = 0$ (with increment $2r_y^2 = 72$)

$2r_x^2 y = 2r_x^2 r_y$ (with increment $-2r_x^2 = -128$).

Solution:

Mid-point Ellipse (8,6) ← (a,b)

$x = 0, \quad y = 6$

For Region 1

Initial point is (0,6)

∴ & Initial decision parameter $(d_1) = 6^2 + \frac{8^2}{4} - 8^2 \times 6$ ← from according to algorithm formula

$= 36 + 16 - 384$

$= -332$

Now, the successive mid-point decision parameter values and the pixel positions along the ellipse are listed in following table:-

d_1	(x,y)	b^2x	a^2y
−332	(0,6)	0	384
−224	(1,6)	36	384
−44	(2,6)	72	384
208	(3,6)	108	384
−108	(4,5)	144	320
288	(5,5)	180	320
116	(6,4)	216	256
272	(7,3)	252	192

Since $d_1 < 0$ so according to algorithm if part runs that is in step-3

Since $d_2 \geq 0$ so according to algorithm else part runs that is in step-3

Since $b = 6$ & $x = 0$ ∴ $b^2x = 6^2 \times 0 = 0$

Since $a = 8$ & $y = 6$ ∴ $a^2y = 8^2 \times 6 = 384$

Since $(a^2y \geq b^2x)$ condition is false here so we stop here. We run until condition becomes true

from algorithm we have final point (7,3) i.e. $x = 7+1 = 8$, $y = 3-1 = 2$

For Region 2

Initial point is (8,2)

& Initial decision parameter $(d_2) = (7+0.5)^2 \cdot 6 + (3-1)^2 \cdot 8 - 36 \times 64$

$= 2025 + 256 - 2304$

$= -23$

Now, the remaining positions along the ellipse path are as follows:-

d_2	(x,y)	b^2x	a^2y
−23	(8,2)	288	128
561	(9,1)	324	64
626	(9,0)	324	0

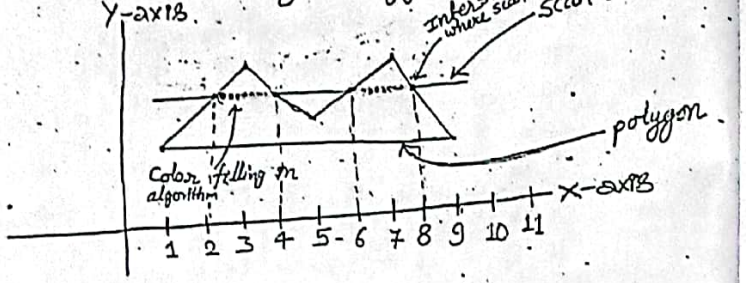
Since $y \geq 0$ condition meets so we stop here

After Finally plotting these points (i.e. (x,y)) on graph we get required ellipse.

⊛ Area Filling:

1) Scan line Polygon fill Algorithm:

Basic concept → Scan line polygon filling algorithm is used for solid color filling in polygons.



Algorithm Steps:

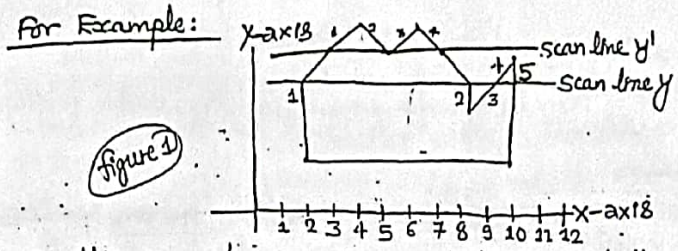
1. The horizontal scanning of the polygon from its lowermost to topmost vertex is done.
2. Find all the intersections of the scan line with all edges of the polygon, from left to right.
   eg. In the above figure point of intersections are 2, 4, 6 and 8.
3. Sort the intersections by increasing x-coordinate i.e, from left to right.
4. Make pairs of the intersections and fill in colour within all the pixels inside the pair.
   e.g. (2,4) and (6,8) between these pixels colour is filled.

Special Cases:

Some scan-line intersections at polygon vertices require special handling.

A scan line passing through a vertex intersects two polygon edges at that position, adding two points to the list of intersections for the scan line.

For Example:



In this example scan line y and scan line y' both passes through an vertex or an edge endpoint. Now in case of scan line y' the scan line is intersecting 4 edges i.e, even number of edges and also passing through a vertex/endpoint.

Also both the edges that are connected to the vertex are on same side, of the scan line. So we need to count the vertex/endpoint TWICE so that we can make pairs of intersection points as: (3,5) & (5,7).

Next Example:

We consider same above figure 1 for this example. Now in case of scan line y, the scan line is intersecting with 5 different edges i.e, ODD NUMBER and also passing through a vertex/endpoint. So in this case we need to do some extra processing i.e, we need to check if the 2 edges at the endpoint through which the scan line is passing are they on opposite sides or on same sides?

In case of scan line y' they are on same sides and in case of scan line y both edges at the vertex are on opposite sides. So now we need to count the vertex as a single intersection point. Now we just need to sort the intersection points and make pairs of them and fill all pixels which lie inside the pair.

⊛2) Inside-Outside test:

One simple way of finding whether the point is inside or outside a simple polygon is to test how many times a ray, intersects the edges of the polygon. If the point is on the outside of the polygon the ray will intersect its edge an even number of times. If the point is on the inside of the polygon the ray will intersect its edge an odd number of times. This method won't work if the point is on edge of polygon.

This method is also known as counting number method. There are two methods by which we can identify whether particular point is inside an object or outside.

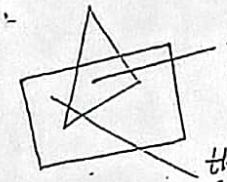
(a) Odd-Even method →

- In odd-even rule/method we draw a line from any position P to a distant point outside the coordinate extents of the object and count the number of edges crossings along the line.

  Condition → We need to make sure that the line drawn from point P does not intersect a vertex or endpoint.

- If the number of edges crossed by the line is ODD then the Point P is in the interior.
- If the number of edges crossed by the line is EVEN then the Point P is in the exterior.

Example:-



this line denotes EXTERIOR since it crossed 2 (i.e. even) number of edges.

this line denotes INTERIOR since it crossed 3 (i.e. odd) number of edges.

(b) Non-Zero winding number method →

- In non-zero winding number method we need to know the direction of each edge in the polygon, i.e, whether the edge is clockwise or counter-clockwise (i.e. anti-clockwise)
- The winding number is the number of times the polygon edges wind around a particular point in the counter clockwise direction.

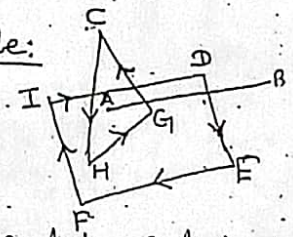
Steps to Apply this method:

1. First we keep the initial value of winding number = 0.
2. Then we imagine drawing a line from point P to outside the polygon which does not pass through any vertex.
3. Now we add 1 to the winding number every time we intersect a polygon edge that crosses the line from right to left and subtract 1 every time we intersect an edge that crosses from left to right

uts / Results

- The interior points are those which are having a non-zero value for winding number. (i.e, maybe +ve or -ve number except zero).
- Exterior points are those whose value of the winding number is zero.

Example:



Firstly we take initial value of winding number as zero (i.e, INITIAL VALUE = 0). Let we are testing at point A towards B. then G will be our rightside and C will be our leftside. The direction of G to C line is right to left (i.e, anti-clockwise direction). So, we add 1 to winding number. Now we reach at point where AB intersects DE which is left to right (i.e, clockwise direction). So, we subtract 1 to winding number. Then finally +1 and −1 becomes zero in winding number. So, final winding number for line AB is zero which shows it as exterior region.

Scan line fill of Curved Boundry Area:-

(a) Boundry fill algorithm → In boundry fill algorithm the basic concept is filling the color in closed area by starting at a point inside a region and paint the interior outward towards the boundry. One requirement for boundry fill algorithm is that the boundry has to have a single color.

Working:-

- In boundry fill algorithm, we start from a point inside the region and fill the color interior outward towards the boundry pixel by pixel.
- We should check the default color of the pixel before filling. If the color is not boundry color, then we fill it with the fill color, and move to next pixel and check for the same criteria till we encounter the boundry colored pixel or boundry.

Methods of implementation → There are two methods in which the boundry fill algorithm can be implemented.

i) 4-connected pixel
ii) 8-connected pixel.

4-connected pixel method

In 4 connected pixel method we check 4 pixels adjcent to the current pixel, namely towards the left, right, top & bottom. So we fill the area with 4-connected pixel method by following steps:-

Step 1. First initialize the 4 values namely x, y, fill-color & Default-color. Where x and y are coordinate positions of the initial interior pixel, fill colour is the colour we want to fill and Default-color is the default color of the interior pixel.

Step 2. Define the value of the boundry pixel color or boundry color.

Step 3. Now check if the current pixel is of Default-color and if yes then Repeat step 4 and step 5 till the boundry pixels are reached.

Step 4. Change the default color with the fill color at the current pixel

Step 5. Repeat step 3 and step 4 for the neighbouring 4 pixels.

8-connected pixel method

There is a problem with 4-connected pixel method that it cannot fill all the pixels. So the solution for this problem is 8-connected pixel method. So in 8-connected pixel method we instead of filling just 4 adjcent pixels we fill also the adjcent diagonal pixel positions, such as (x+1, y+1).

(b) Flood-fill algorithm → Flood-fill algorithm is useful in cases where there is no single color boundry for the polygon. i.e, the boundry has multiple colors. In flood fill algorithm instead of filling color till we encounter a specific boundry color we just fill the pixels with default color. It is used in the "bucket" fill tool of paint programs.

There are two methods in which flood-fill algorithm can be implemented: 4-connected pixel & 8-connected pixel.

4-connected pixel.

(Similar lines as we wrote before in boundry fill)

Steps:

Step1. Initialize the 4 values first namely x, y, Fill-color & Default-color. -- (Step 1 is also similar as we wrote before).

Step2. If the color of node is not equal to default-color, return.

Step3. Set the color of node to replacement-color.

Step4. Set the color of node to the south of node to replacement-color.

Step5. Set the color of node to the north of node to replacement color.

Step6. Set the color of node to the east of node to replacement color

Step7. Set the color of node to the west of node to replacement color.

8-connected pixel method

We write similar as we wrote for (8-connected pixel) of boundry fill algorithm.

Unit-3

Two-Dimensional Geometric Transformations:

The operations that are applied to geometrical description of an object to change its position, orientation or size are called geometric transformations. Following are the types of transformations:

i) Translation
ii) Rotation
iii) Reflection
} (transformations without change in shape) i.e Rigid body transformation.

iv) Scaling
v) Shearing
} (transformations with change in shape) i.e Non-rigid body transformation.

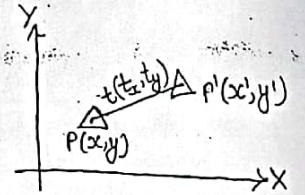
⊛. Why geometric transformations?

Ans:- In computer graphics, transformations of 2D objects are essential to many graphics applications like as a viewing aid, as a modeling tool, as an image manipulation tool etc. Transformations are needed to manipulate the initially created object and to display the modified object without redrawing it.

Rotation, Translation and scaling are major three transformations that are extensively used by most of all graphical packages. Other than these reflection and shearing transformations are also used by some graphical packages.

1) Translation:

The change in position of any object is called translation. Let $P(x,y)$ be any object translated by $t(t_x, t_y)$ to point $P'(x',y')$ as shown in the figure.



Here, $t(t_x, t_y)$ is the translation distance, t_x is translation in x-direction & t_y is translation in y-direction.

We can calculate translated points x' & y' as follows:-

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

i.e, $P' = T + P$

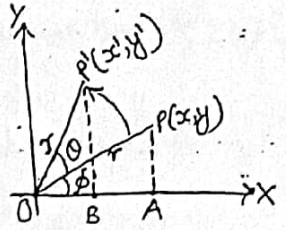
Now this can be written in matrix form as follows:-

$$P' = \begin{bmatrix} x' \\ y' \end{bmatrix}, \quad T = \begin{bmatrix} t_x \\ t_y \end{bmatrix}, \quad P = \begin{bmatrix} x \\ y \end{bmatrix}$$

2) Rotation:

It is the process of changing the angle of the object. Rotation can be done by clockwise or anti-clockwise.

Let point $P(x,y)$ is at distance 'r' (i.e, radius) from origion making an angle ϕ with x-axis. Now if we rotate it with an angle θ we get new points as $P'(x',y')$ as shown in figure. New rotated points $P'(x',y')$ can be calculated as follows:-



$$x' = r\cos(\phi+\theta) \quad \text{---(i)}$$

or, $\cos(\phi+\theta) = \frac{x'}{r}$ $\left(\because \cos\theta = \frac{\text{base}}{\text{hypotenuse}}\right)$

& $y' = r\sin(\phi+\theta)$ ---(ii)

or, $\sin(\phi+\theta) = \frac{y'}{r}$ $\left(\because \sin\theta = \frac{\text{perpendicular}}{\text{hypotenuse}}\right)$

Now taking eqn (i) and (ii)

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

or, $x' = r[\cos\phi.\cos\theta - \sin\phi.\sin\theta]$ $\left[\because \cos(A+B) = \cos A.\cos B - \sin A.\sin B\right]$

or, $x' = r\cos\phi.\cos\theta - r.\sin\phi.\sin\theta$ ---(iii)

& $y' = r\sin(\phi+\theta)$

or, $y' = r[\sin\phi.\cos\theta + \cos\phi.\sin\theta]$ $\left[\because \sin(A+B) = \sin A.\cos B + \cos A.\sin B\right]$

or, $y' = r\sin\phi.\cos\theta + r\cos\phi.\sin\theta$ ---(iv)

Now from triangle OAP

$$\cos\phi = \frac{x}{r} \quad \left(\because \cos\phi = \frac{\text{base}}{\text{hypotenuse}}\right)$$

or, $x = r\cos\phi$ ---(v)

& $\sin\phi = \frac{y}{r}$ $\left(\because \sin\phi = \frac{\text{perpendicular}}{\text{hypotenuse}}\right)$

or, $y = r.\sin\phi$ ---(vi)

Now using value of (vi) and (vii) in eqn (iii) and (iv).

$$x' = r\cos\phi.\cos\theta - r\sin\phi.\sin\theta.$$

or, $x' = x.\cos\theta - y\sin\theta$ — (viii)

& Similarly $y' = r\sin\phi.\cos\theta + r.\cos\phi.\sin\theta$

or, $y' = y.\cos\theta + x.\sin\theta$ — (viii).

Hence we get final result as:

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

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

This can be represented in matrix form as follows:-

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

$$P' = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

$$P = \begin{bmatrix} x \\ y \end{bmatrix}$$

$P' = R(\theta).P$ where, $R(\theta)$ is rotation matrix with angle θ about origin.

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

But when rotation is in clockwise direction then,

$P' = R(-\theta).P$

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

$\because x' = x\cos(-\theta) - y\sin(-\theta)$
$= x\cos\theta + y\sin\theta$
& $y' = x\sin(-\theta) + y\cos(-\theta)$
$= -x\sin\theta + y\cos\theta$

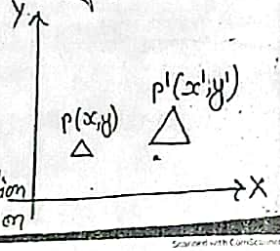
3) Scaling:

Scaling is the process that alters size of an object. The size may increase or decrease on the basis of scaling factor.

Let $P(x,y)$ be the object initially and $P'(x',y')$ be the object after scaling. So, final results x' and y' can be calculated as;

$$x' = x.S_x$$

$$y' = y.S_y$$ where, S_x = Scaling factor in x-direction & S_y = Scaling factor in y-direction



Now the General form for scaling is; $\boxed{P' = S_n.P}$

Now, $P' = S_n.P$ can be written in matrix form as;

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

or, $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x.S_x + 0 \\ 0 + y.S_y \end{bmatrix}$

or, $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x.S_x \\ y.S_y \end{bmatrix}$ //

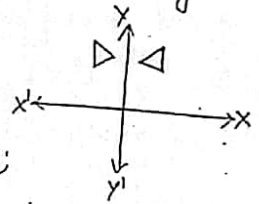
4) Reflection:

It is a transformation which produces a mirror image of an object. The mirror image can be either about x-axis or y-axis. The object is rotated by 180°. In 2D following types of reflections are performed.

i) Reflection about a line where x=0 or y-axis:-

In this reflection y-coordinate remains unchanged and sign of x-coordinate is altered.

Let $P(x,y)$ be point that is to be reflected about y-axis and $P'(x',y')$ be the resultant point then we can represent it in matrix form as follows;



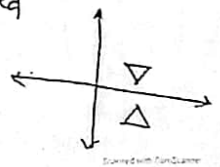
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

i.e, $\boxed{P' = R_y.P}$

ii) Reflection about a line y=0 or x-axis:-

In this reflection x-coordinate remains unchanged and sign of y-coordinate is altered.

Let $P(x,y)$ be point that is to be reflected about x-axis and $P'(x',y')$ be the result point. Now in matrix form;



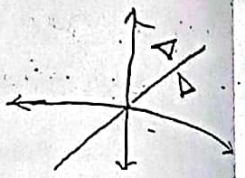
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

i.e. $\boxed{P' = R_x \cdot P}$

iii) Reflection about a line $y=x$

In this reflection the value of x and y-coordinate are swapped.

Let $P(x,y)$ be the point that is to be reflected about $y = x$ & $P'(x',y')$ be the resultant point.

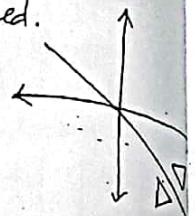


In Matrix form:

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

iv) Reflection about a line $y=-x$:

In this reflection the value of x and y-coordinate are swapped as well as sign of x & y are changed.



i.e. $x = -y$

$y = -x$.

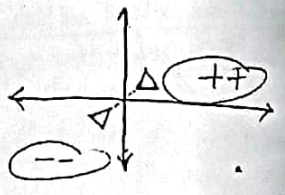
Equivalent matrix form is;

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

v) Reflection about a line perpendicular to origin

In Matrix Form

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



Signs of x and y coordinates are changed.

5) Shearing:

A shear is a transformation that distorts the shape of an object along coordinate axes. It disturb the shape of an object such that the transformed shape appears as if the object were composed of interval layers that has been caused to slide over each other. Two common shearing transformations are those that shift coordinate x-values and those that shift y-values.

X-direction Shear:

A x-direction shear relative to x-axis is produced with transformation matrix equation.

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

which transforms $x' = x + sh_x \cdot y$

and $y' = y$.

Y-direction Shear:

A y-direction shear relative to y-axis is produced by following transformation matrix equation.

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

which transforms $x' = x$

and $y' = sh_y \cdot x + y$.

Homogenous co-ordinate representation of 2D Transformations:

The homogenous co-ordinate system provides a uniform frame-work for handling different geometric transformation, simply as multiplication of matrices. In homogenous coordinate representation each 2D point (x,y) is represented as homogenous coordinate triple (x_h, y_h, h)

where, $x = \frac{x_h}{h}$, $y = \frac{y_h}{h}$

Note: h is usually 1 for 2D case.

It is called homogenous because it is possible to transform functions such as $f(x,y)$ into the form of $f(x_h/h, y_h/h)$ without disturbing degree of curve

Q. Why homogenous coordinates?

Ans:- It is needed for following reasons;

i) It provides a uniform frame-work for handling different geometr transformations, simply as multiplication of matrices.

ii) To express any two-dimensional transformation as matrix multiplication.

iii) To represent all the transformation equations as matrix multiplicati

iv) To perform more than one transformation at a time.

v) To reduce unwanted calculations of intermediate steps, to save time and memory and produce a sequence of transformations.

Homogenous co-ordinate representation for translation:

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

$\therefore P' = T_H(t_x, t_y) \cdot P$

Image = transformation matrix × object.

where, t_x = translation along x-a
& t_y = translation along y-a

Q. Translate a point P(7, 8) by translation vector T(-2,4).

Soln Here, $P(7,8) = P(x,y)$ & $T(t_x, t_y) = (-2,4)$

$$x' = x + t_x = (7-2) = 5$$
$$y' = y + t_y = (8+4) = 12$$
$$P'(x',y') = (5,12)$$

This is a method without homogenous coordinate system, na we solve using homogenous coordinate system as follows;-

$$P(x,y) \xrightarrow{H} P(x,y,1)$$
$$P'(x',y') \xrightarrow{H} P'(x',y',1)$$

So, $T_H(t_x, t_y) = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$

$$\therefore P' = T_H(t_x, t_y) \cdot P$$
$$= \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 7 \\ 8 \\ 1 \end{bmatrix}$$
$$= \begin{bmatrix} 5 \\ 12 \\ 1 \end{bmatrix}$$

Homogenous coordinate representation for rotation:

Let P(x,y) be the point that is rotated about origin with rotation angle θ and $P'(x',y')$ be the resultant point.

$$P(x,y) \xrightarrow{H} P(x,y,1)$$
$$P(x',y') \xrightarrow{H} P'(x',y',1)$$

we know that,

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

Now in homogenous coordinate matrix form:

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

$\therefore P' = R_H(\theta) \cdot P$

For inverse rotation $P' = R_H(-\theta) \cdot P$

Homogenous coordinate representation for scaling:

Let P(x,y) be the point that is to be scaled w.r.t. origin with scaling factor $S(S_x, S_y)$ and $P'(x',y')$ be the resultant point.

$$P(x,y) \xrightarrow{H} P(x,y,1)$$
$$P'(x',y') \xrightarrow{H} P'(x',y',1)$$

we know that,

$$x' = x \cdot S_x$$
$$y' = y \cdot S_y$$

Now in homogeneous coordinate matrix form:

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

$$\therefore P' = S_{11}(S_x, S_y) \cdot P$$

For inverse scaling

$$P' = S_{11}\left(\frac{1}{S_x}, \frac{1}{S_y}\right) \cdot P$$

⊛ Numerical Questions:

Q1. A point (4,3) is rotated clockwise by an angle of 45°. Find the rotation matrix and the resultant point.

Solution:

Clockwise rotation matrix with angle θ (i.e. Transformation matrix)

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

$$= \begin{bmatrix} \cos 45° & -\sin 45° \\ \sin 45° & \cos 45° \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$

we have, $P' = R(\theta) \cdot P$

i.e, Image = Transformation matrix × Object.

or, $P' = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix} = \begin{bmatrix} \frac{4}{\sqrt{2}} - \frac{3}{\sqrt{2}} \\ \frac{4}{\sqrt{2}} + \frac{3}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{7}{\sqrt{2}} \end{bmatrix}$

Q2. Find the transformation of triangle A(1,0), B(0,1) and C(1,1) by rotating 90° about the origin and then translating one unit in x and y direction.

Solution:

$$\text{Object} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Transformation Matrix = T(1,1) · R(90°)

$$= \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos 90° & -\sin 90° & 0 \\ \sin 90° & \cos 90° & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(translation in homogeneous coordinate system $\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$; Rotation in homogeneous coordinate system)

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

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

Q3. Perform a counter clockwise 15° rotation of a triangle A(2,3), B(5,5) & C(1,3) about point (1,1).

Solution:

$$\text{Object} = \begin{bmatrix} 2 & 5 & 4 \\ 3 & 5 & 3 \\ 1 & 1 & 1 \end{bmatrix}$$

Composite Transformation Matrix = T(1,1) × R(45°) × T(−1,−1) (→ translation by (1,1), → rotation by 45°, for counter clockwise (translation with −ve))

$$= \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 45° & -\sin 45° & 0 \\ \sin 45° & \cos 45° & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 1 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\sqrt{2}+1 \\ 0 & 0 & 1 \end{bmatrix}$$

we have, Image = Transformation matrix × Object.

or, Image $= \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 1 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\sqrt{2}+1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 5 & 4 \\ 3 & 5 & 3 \\ 1 & 1 & 1 \end{bmatrix}$

$$= \begin{bmatrix} -\frac{1}{\sqrt{2}}+1 & 1 & \frac{1}{\sqrt{2}}+1 \\ \frac{3}{\sqrt{2}}+1 & \frac{0}{\sqrt{2}}+1 & \frac{5}{\sqrt{2}}+1 \\ 1 & 1 & 1 \end{bmatrix}$$

⊛ Fixed point Scaling:

To control the location of scaled object we can choose the position called fixed point. Let coordinates of fixed point $= (x_f, y_f)$. For a vertex with coordinate (x, y) the scaled co-ordinates (x', y') are calculated as;

$$x' = x_f + (x - x_f) \cdot S_x$$

or $\quad x' = x \cdot S_x + (1 - S_x) x_f$

& $\quad y' = y_f + (y - y_f) \cdot S_y$

or $\quad y' = y \cdot S_y + (1 - S_y) \cdot y_f$

where, $(1 - S_x) \cdot x_f$ and $(1 - S_y) \cdot y_f$ are constant for all points in object.

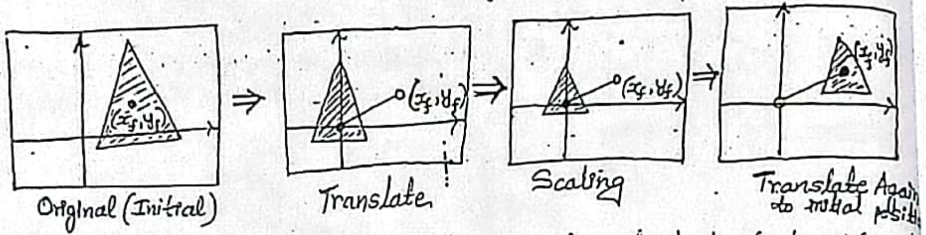


figure:- Process of Scaling of a triangle about a fixed point (x_f, y_f).

(translate) (Scaling) (origin translate to get original) (Gives scaling (S_x, S_y) at fixed point (x_f, y_f))

$$T(x_f, y_f) \cdot S(S_x, S_y) \cdot T(-x_f, -y_f) = S(x_f, y_f, S_x, S_y)$$

Now, in matrix form

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

Hence the homogenous matrix equation for fixed point scaling is;

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

Q1. Scale the polygon with coordinates A(2,5), B(7,10) and C(10,2) by two units in x-direction and two units in y-direction.

Solution

Here, $S_x = 2$ and $S_y = 2$

Therefore transformation matrix is given as $S = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$

The object matrix is; Object $= \begin{bmatrix} 2 & 7 & 10 \\ 5 & 10 & 2 \end{bmatrix}$

We have,

Image = Transformation Matrix × Object.

$$= \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \times \begin{bmatrix} 2 & 7 & 10 \\ 5 & 10 & 2 \end{bmatrix}$$

$$= \begin{bmatrix} 4 & 14 & 20 \\ 10 & 20 & 4 \end{bmatrix}$$

∴ The coordinates of the image after scaling are A'(4,10), B'(14,20) and C'(20,4).

Q2. Find the transformation matrix that transforms the given square ABCD to half its size with centre still remaining at the same position. The coordinates of the square are; A(1,1), B(3,1), C(3,3), D(1,3) and centre at (2,2). Also find the resultant coordinates of square.

Solution:-

This transformation can be carried out in the following steps.

i) Translate the square so that its centre coincides with origin

ii) Scale the square with respect to the origin.

iii) Translate the square back to its original position.

Here,

$$\text{Object} = \begin{bmatrix} 1 & 3 & 3 & 1 \\ 1 & 1 & 3 & 3 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

— जतिको point हरु, matrix form मा।

— base मा सँधै 1 homogenous बनाउँन object लाई

Now the transformation matrix is obtained by multiplication of three matrices as;

Composite Transformation matrix $= T(x_f, y_f) \times S(S_x, S_y) \times T(-x_f, -y_f)$

(Simply Transformation matrix)

(Half its size given in question)

$$= T(2, 2) \times S(0.5, 0.5) \times T(-2, -2)$$

$$= \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.5 & 0 & 1 \\ 0 & 0.5 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

$x_f(1 - S_x) = 2(1 - 0.5) = 1$

$y_f(1 - S_y) = 2(1 - 0.5) = 1$

यो दुईटा बाहेक अरु S(0.5,0.5) को सँधै जस्तै हुन्छ

Scanned with CamScanner

We have,

Image = Transformation Matrix × Object.

$$\text{Image} = \begin{bmatrix} 0.5 & 0 & 1 \\ 0 & 0.5 & 1 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 3 & 3 & 1 \\ 1 & 1 & 3 & 3 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1.5 & 2.5 & 2.5 & 1.5 \\ 1.5 & 1.5 & 2.5 & 2.5 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Hence coordinates of resultant image are; A'(1.5, 1.5), B'(2.5, 1.5), C'(2.5, 2.5) and D'(1.5, 2.5).

Q3. Find out the final coordinates of a figure bounded by the coordinates (1,1), (3,4), (5,7), (10,3) when rotated about a point (8,8) by 30° in clockwise direction and scaled by two unit in x-direction and three unit in y-direction.

Solution:

$$\text{Object} = \begin{bmatrix} 1 & 3 & 5 & 10 \\ 1 & 4 & 7 & 3 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Now, the transformation matrix is obtained by multiplication of four matrices as;

Transformation Matrix = $T(x_f, y_f) \times S(S_x, S_y) \times R(-\theta) \times T(-x_f, -y_f)$.

$= T(8,8) \times S(2,3) \times R(-30°) \times T(-8,-8)$.

$$= \begin{bmatrix} 1 & 0 & 8 \\ 0 & 1 & 8 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \cos 30° & \sin 30° & 0 \\ -\sin 30° & \cos 30° & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & \\ 0 & 1 & \\ 0 & 0 & \end{bmatrix}$$

On multiplying T(8,8) and T(-8,-8) we get same matrix. S(2,3) only change its done at points $x_f(1-S_x) = 8(1-2) = -8$ & $y_f(1-S_y) = 8(1-3) = -16$ & Matrix R(-30°) is as it is written. This is same that we did in previous question difference is, additional R(-30°) only.

$$= \begin{bmatrix} 2 & 0 & -8 \\ 0 & 3 & -16 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.866 & 0.5 & 0 \\ -0.5 & 0.866 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1.732 & 1 & -8 \\ -1.5 & 2.598 & -16 \\ 0 & 0 & 1 \end{bmatrix}$$

Now, We have,

Image = Transformation Matrix × Object.

i.e, $$\text{Image} = \begin{bmatrix} 1.732 & 1 & -8 \\ -1.5 & 2.598 & -16 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 3 & 5 & 10 \\ 1 & 4 & 7 & 3 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1.732+1-8 & 5.196+4-8 & 8.66+7-8 & 17.32+3-8 \\ -1.5+2.598-16 & -4.5+10.392-16 & -7.5+18.186-16 & -15+7.794-16 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} -5.268 & 1.196 & 7.66 & 12.32 \\ -14.902 & -10.108 & -5.314 & -23.206 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Hence the final coordinates after transformation are; (-5.268, -14.902), (1.196, -10.108), (7.66, -5.314) and (12.32, -23.206).

Two Dimensional Viewing:

2-D viewing is the process of producing different views of a 2D secence. For e.g. translated view, rotated view etc. Graphical package allows us to define part of the scene to be displayed and part of screen where defined view is displayed. In computer graphics several coordinate systems are used to construct and display a 2D scene. Some of which are as follows:-

i) Modeling coordinate system → It defines shape and size of individual objects to create a graphical scene. Arbitrary units are used to specify modeling coordinates (MC's).

ii) World coordinate system → It is used to organize the individual objects (i.e, points, lines, circles etc) into a scene. A scene is made up of collection of objects. The objects make up the world (or scene) that we want to view, and the coordinates that we use to define the scene are called world coordinates (WC's).

iii) Viewing coordinate system → It is used to define particular views of a 2D scene. For a 2-D picture, a view is selected by specifying a subarea of the total picture area. In short it is called VC's.

iv) Normalized viewing coordinate system → Normalized viewing coordinates (NVC's) are the viewing coordinates between 0 and 1. They are used to make the viewing process independent of the output device.

v) Device coordinate or Screen coordinate system → Device coordinate system (DC's) are used to define coordinates in an output device. Device coordinates are integers within the range $(0,0)$ to (x_{max}, y_{max}) for a particular output device.

Note: Above these 5 points ① to ⑤ are viewing pipeline

⑥ What is 2D viewing transformation? (Please refer video link provided to drive by starting this for better understanding)

Ans: The process of mapping of a part of world coordinate scene to device coordinate is called the 2D viewing transformation. Transformations from world to device coordinates involves translation, rotation and scaling operations, as well as procedures for deleting those parts of picture that are outside the limits of a selected display area (clipping).

→ Clipping is necessary to discard that part of a 2D scene that lie outside of window.

→ Clipping can be performed in world co-ordinate or device coordinate.

Window: A world coordinate area selected for display is called a window. It is the section of 2D scene selected for viewing.

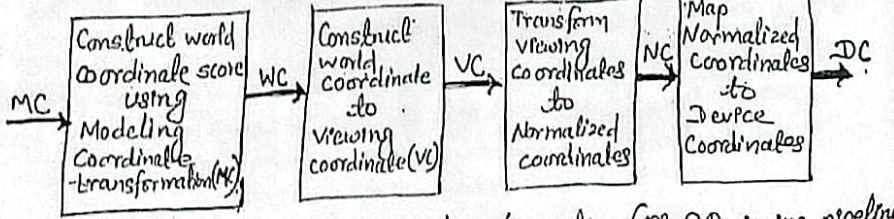


Figure:- 2D viewing transformation (OR 2D viewing pipeline)

⑦ Viewport:- A area on a display device to which a window is mapped is called a viewport. The viewport indicates where selected part will be displayed on an output device.

Window and viewport are often rectangular in standard positions, because it simplifies transformation and clipping process.

⑧ Window to viewport coordinate transformation:

→ A window can be specified by four, world coordinates; xw_{min}, xw_{max}, yw_{min} and yw_{max}.

→ Similarly a viewport can be described by four coordinates; xv_{min}, xv_{max}, yv_{min} and yv_{max}.

The window-to-viewport transformation can be explained in three steps as below;

Step1: Translate object along with window such that the lower left corner of the window is at origin.

i.e, apply $T(-xw_{min}, -yw_{min})$

Step2: Scale the object and the window such that window has the same dimension as that of a viewport. Simply we can say that converting the object into image and window into viewport.

i.e, apply $S(S_x, S_y)$

Step3: Finally apply another translation to move the viewport to its original position on the screen.

i.e, apply $T(xv_{min}, yv_{min})$

Therefore, net transformation (composite transformation) is;

$$T_{wv} = T(xv_{min}, yv_{min}) \cdot S(S_x, S_y) \cdot T(-xw_{min}, -yw_{min}) \quad \text{— (1)}$$

where, S_x and S_y are scaling factors: $S_x = \frac{xv_{max} - xv_{min}}{xw_{max} - xw_{min}}$

& $S_y = \frac{yv_{max} - yv_{min}}{yw_{max} - yw_{min}}$

Now, writing in matrix form:

$$T_{wv} = \begin{bmatrix} 1 & 0 & xv_{min} \\ 0 & 1 & yv_{min} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -xw_{min} \\ 0 & 1 & -yw_{min} \\ 0 & 0 & 1 \end{bmatrix}$$

This is required window-to-viewport transformation. Let $P(x,y)$ be the world coordinate point that is mapped onto the viewport point $P'(u,v)$, then we must have,

$$P' = T_{wv} \cdot P$$

Example: Find the normalization transformation window to viewport, with window, lower left corner at (1,1) and upper right corner at (3,5) onto a viewport with lower left corner at (0,0) and upper right corner at (0.5, 0.5).

Solution:

Coordinates for window:

$xw_{min} = 1$ $\quad yw_{min} = 1$

$xw_{max} = 3$ $\quad yw_{max} = 5$

Coordinates for viewport:

$xv_{min} = 0$ $\quad yv_{min} = 0$

$xv_{max} = 0.5$ $\quad yv_{max} = 0.5$

We know that,

$$S_x = \frac{xv_{max} - xv_{min}}{xw_{max} - xw_{min}} = \frac{0.5 - 0}{3 - 1} = 0.25$$

&

$$S_y = \frac{yv_{max} - yv_{min}}{yw_{max} - yw_{min}} = \frac{0.5 - 0}{5 - 1} = 0.125$$

Now, Composite Transformation matrix is; $T_{wv} = T(xv_{min}, yv_{min}) \cdot S(S_x, S_y) \cdot T(-xw_{min}, -yw_{min})$

or,

$$T_{wv} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0.25 & 0 & 0 \\ 0 & 0.125 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.25 & 0 & -0.250 \\ 0 & 0.125 & -0.125 \\ 0 & 0 & 1 \end{bmatrix}$$

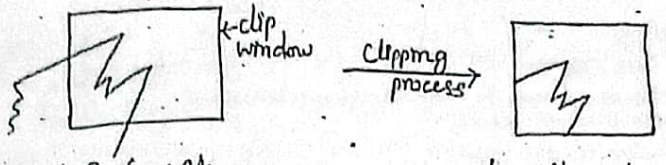
→ $xv_{min}(1 - S_x)$

→ $yv_{min}(1 - S_y)$

others same as it is in S

Clipping:

Clipping is the process of discarding (cutting off) those parts of an object which are outside of specified region (windows) is called clipping. Any procedure that identifies those parts of a picture that are either inside or outside of the specified region is called a clipping algorithm. The region against which the clipping operation is performed is called a clip window.



a) Before clipping b) After clipping.

fig: Clipping process

Applications of Clipping:

i) Extracting part of a defined scene for viewing.
ii) Identifying visible surfaces in three-dimensional views.
iii) Anti-aliasing line segments or object boundaries.
iv) Creating objects using solid-modeling procedures.
v) Displaying a multi-window environment.

Types of clipping:

i) Point Clipping:

→ Given a clipping window x_{min}, x_{max}, y_{min} & y_{max} and coordinate point $P(x,y)$ as shown in figure:

→ $P(x,y)$ is accepted for display iff

$x_{min} \le x \le x_{max}$

$y_{min} \le y \le y_{max}$

→ Otherwise point is clipped. i.e, $P(x,y)$ is discarded.

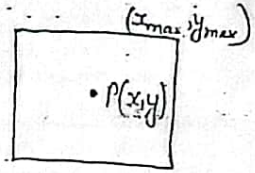


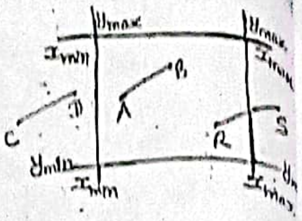
fig: Point Clipping

ii) Line clipping:

→ It is the process of discarding those parts of a line that lie outside of a specified region. (i.e, clipping window).

→ Given a clipping window $(x_{min}, x_{max}, y_{min}, y_{max})$ then there are three possible cases:-

i) **Visible:** → Line lies completely inside the window.
→ Accept and display the line.
→ E.g. AB line in the figure.



ii) **Completely invisible:**
→ Line lies completely outside of the clipping window.
→ Discard the line segment.
→ E.g. CD line in the figure.

iii) **Partially visible:**
→ Part of line segment lies inside of the clipping window.
→ In this case we need to compute intersection points and clip at the intersection points.
→ E.g. RS line in the figure.

1) **Cohen Sutherland line clipping:**

This is one of the oldest and most popular line-clipping algorithm. In this method, coordinate system is divided into nine regions. All regions have their associated region codes. ~~Every line endpoint is assigned a four digit binary code.~~ Each bit in the code is set to either 1 (true) or 0 (false). Each region is assigned a four bit pattern as shown in figure.

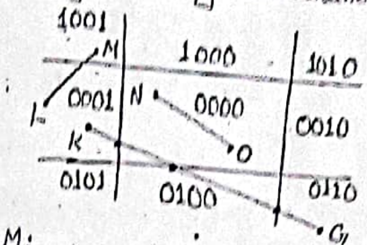
1001	1000 Top	1010
0001 Left	0000 Window	0010 Right
0101	0100 Bottom	0110

fig. Region codes according to TBRL

Any point inside the clipping window has a region code 0000. For any endpoint (x,y) of a line, the code can be determined that identifies in which region the endpoint lies. The region code bits are set according to following conditions:
i) First bit is set to 1 if point lies towards left of window.
ii) Second bit is set to 1 if point lies towards right of window.
iii) Third bit is set to 1 if point lies at bottom of window.
iv) Fourth bit is set to 1 if point lies at top of the window.

Algorithm:
Step1: Given a line segment with endpoints $P_1(x_1,y_1)$ and $P_2(x_2,y_2)$.
Step2: Compute the 4-bit region codes (outcodes) for two endpoints of line segment.
Step3: If outcodes of P_1 = outcode of P_2 = 0000 (i.e. $P_1 \| P_2 = 0000$) then, line lies completely inside the window, so draw the line segment.
Step4: else if outcode of P_1 and outcode of $P_2 \neq 0000$ (i.e. $P_1 \& P_2 \neq 0000$) then, line lies completely outside the window, so discard the line segment.
Step5: else if outcode of P_1 and outcode of $P_2 = 0000$ (i.e. $P_1 \& P_2 = 0000$) then compute the intersection of line segment with window boundary and discard the portion of line segment that falls completely outside of the window. Assign a new four-bit code to the intersection and repeat until either 3 or 4 steps are satisfied.

Q1. Clip the following lines using Cohen Sutherland line clipping algorithm.



Solution:

(window में हैं तो logical OR का logical AND)

(Logical AND = 0000 accepted
& Logical AND ≠ 0000 rejected)

i) **For line LM:**
The outcode for L is 0001
and outcode for M is 1001
The logical AND for LM = 0001 AND 1001 = 0001
which is non zero hence it is rejected.

ii) **Also for line ON:**
The outcode for O is 0000
& the outcode for N is 0000
The logical OR for ON = 0000 OR 0000 = 0000
which is zero and inside the window so, it is accepted.

iii) **Again for line KG:**
The outcode for K is 0001
and outcode for G is 0110
The logical AND for KG = 0001 AND 0110 = 0000
This is zero. This means that this line crosses more lines which contain the clipping boundry.
Now, the line segment KG is broken into KJ, JI, IH and HG and again each segment is tested as shown below:-

For line segment KJ:
The outcode for K is 0001
and outcode for J is 0001
The logical AND for KJ = 0001 AND 0001 = 0001.
Which is non-zero, hence rejected.

For line segment JI:
The outcode for J is 0000
and outcode for I is 0000
The logical AND for JI = 0000 AND 0000 = 0000
Which is zero, hence both end points are inside the window.
So, it is accepted.

for line segment IH:

The outcode for I is 0100
and outcode for H is 0100
The logical AND for IH = 0100 AND 0100 = 0100
which is non-zero hence rejected.

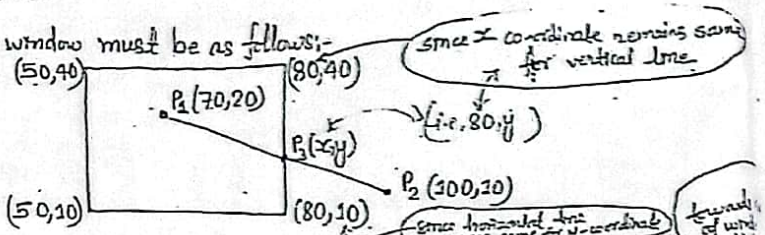
for line segment HG:

The outcode for H is 0110
and outcode for G is 0110
the logical AND for HG = 0110 AND 0110 = 0110
which is non-zero hence rejected.

Q2. Use Cohen-Sutherland algorithm to clip the line $P_1(70,20)$ and $P_2(100,$ against a window lower left hand corner (50,10) and upper right hand corner (80,40).

Solution:

The window must be as follows:-



Since the point P_1 lies inside the window hence its outcode is 0000.
Similarly the outcode of P_2 = 0010

The logical AND for P_1P_2 = 0000 AND 0010 = 0000
which is zero hence line is partially visible.

The slope of line $P_1P_2 = \frac{y_2-y_1}{x_2-x_1} = \frac{10-20}{100-70} = \frac{-10}{30} = \frac{-1}{3}$.

From the above figure $P_3(x,y) = P_3(80,y)$ here we need to find the value of y.

Now slope of line $P_2P_3 = \frac{y-y_2}{x-x_2} = \frac{y-10}{80-100} = \frac{y-10}{-20}$

Since slope of P_1P_2 = slope of P_2P_3

Hence, $\frac{-1}{3} = \frac{y-10}{-20}$

or, $y = 10 + \frac{20}{3} = 16.667$

Now the parts P_2P_3 of line P_1P_2 is clipped (cut off) as it is outside the window.

Hence, $P_3(x,y) = P_3(80, 16.667)$.

2) Liang-Barsky Line Clipping:

→ It is an efficient line clipping algorithm which is based on parametric form of a line.

→ It reduces no. of intersections to be calculated hence performs much better than Cohen-Sutherland algorithm.

The parametric equations of line segment can be written in the form;

$x = x_1 + u\Delta x$

$y = y_1 + u\Delta y$ where, $0 \le u \le 1$.

$\Delta x = x_2 - x_1$

& $\Delta y = y_2 - y_1$.

Now, Following the Liang-Barsky approach, we first write the point clipping in parametric way:

$x_{min} \le x_1 + u\Delta x \le x_{max}$

$y_{min} \le y_1 + u\Delta y \le y_{max}$.

Each of these four inequalities can be expressed as $p_k u \le q_k$ for k = 1,2,3,4 where parameters p and q are defined as;

$p_1 = -\Delta x$ $\quad q_1 = x_1 - x_{wmin}$ ← or simply we can write $x_1 - x_{min}$ neglecting w

$p_2 = \Delta x$ $\quad q_2 = x_{wmax} - x_1$

$p_3 = -\Delta y$ $\quad q_3 = y_1 - y_{wmin}$

$p_4 = \Delta y$ $\quad q_4 = y_{wmax} - y_1$

Now, if $P_1 = 0 \Rightarrow -\Delta x = 0 \Rightarrow \Delta x = 0 \Rightarrow x_2 - x_1 = 0 \Rightarrow x_2 = x_1$

Similarly

if $P_2 = 0 \Rightarrow x_2 = x_1$

if $P_3 = 0 \Rightarrow y_2 = y_1$

& if $P_4 = 0 \Rightarrow y_2 = y_1$

Conditions:

i) If $(P_k = 0)$ then line segment is parallel to window boundy. for k = 1, 2, 3, 4.

ii) If $(P_1 = 0)$ & if $(q_1 < 0)$

$\Rightarrow x_1 - x_{min} < 0 \Rightarrow x_1 < x_{min}$

Then line lies outside of x_{min} boundy.

iii) If $(P_2 = 0)$ & if $(q_2 < 0)$

$\Rightarrow x_{max} - x_1 < 0 \Rightarrow x_{max} < x_1$

Then line lies outside x_{max} boundy.

iv) If ($p_3 = 0$) and if ($q_3 < 0$)
$\Rightarrow y_1 - y_{min} < 0 \Rightarrow y_1 < y_{min}$
Then line lies outside of y_{min} boundry.

v) If ($p_4 = 0$) and if ($q_4 < 0$)
$\Rightarrow y_{max} - y_1 < 0 \Rightarrow y_{max} < y_1$
Then line lies outside of y_{max} boundry.

$\Rightarrow$ Hence, if ($p_i = 0$) and if ($q_i < 0$) then the line lies completely outside of window.

* If $p_i = 0$ and $q_i \geq 0$ then line lies inside of ith boundry.
* If $p_i \neq 0$ then we need to compute intersection point.

To calculate intersection point the below algorithm determine two value of t_i based on those values, line is clipped.

Algorithm:

1.) Input $P_1(x_1, y_1)$ & $P_2(x_2, y_2)$.
2). Input clipping window boundaries $(x_{min}, x_{max}, y_{min}, y_{max})$.
3). Determine p_i and q_i as follows:

$p_1 = -\Delta x$	$q_1 = x_1 - x_{min}$	(Left boundry)
$p_2 = \Delta x$	$q_2 = x_{max} - x_1$	(right ")
$p_3 = -\Delta y$	$q_3 = y_1 - y_{min}$	(bottom ")
$p_4 = \Delta y$	$q_4 = y_{max} - y_1$	(top ")

4). If ($p_i = 0$) and $q_i < 0$ for any i;
$\rightarrow$ line is parallel to window boundry and is outside of window so discard line segment.

5). Set two parameters t_1 & t_2 as;
$t_1 = 0$
$t_2 = 1$

6) Determine r_i as follows;
$r_i = \frac{q_i}{p_i}$ where, $i = 1, 2, 3, 4$.

7) Find all r_i for which $p_i < 0$
Set $t_1 = \max(0, r_i)$.

8) Find all r_i for which $p_i > 0$.
Set $t_2 = \min(1, r_i)$.

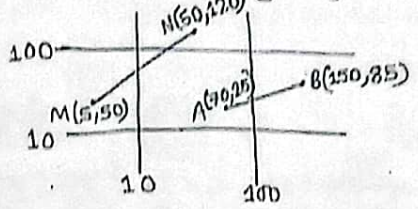
9). If ($t_1 > t_2$)
$\rightarrow$ line is completely outside window so discard line segment.
else {
if ($t_1 = 0$ & $t_2 = 1$)
$\rightarrow$ line is completely inside window, so display the line segment.
else {
clip the line segment and determine (x_1', y_1') and (x_2', y_2') as;

$x_1' = x_1 + t_1 \Delta x$
$x_2' = x_1 + t_2 \Delta x$
$y_1' = y_1 + t_1 \Delta y$
$y_2' = y_1 + t_2 \Delta y$

}
}

10). End.

Q1. Clip the following lines using Liang - Barsky line clipping algorithm.



Solution:

For line AB:

Here, $A(70, 25) = A(x_1, y_1)$
$B(150, 85) = B(x_2, y_2)$
$\Delta x = x_2 - x_1 = 150 - 70 = 80$
$\Delta y = y_2 - y_1 = 85 - 25 = 60$

$x_{min} = 10$, $x_{max} = 100$
$y_{min} = 10$, $y_{max} = 100$

Now, $p_1 = -\Delta x = -80$, $q_1 = x_1 - x_{min} = 70 - 10 = 60$, $r_1 = \frac{q_1}{p_1} = \frac{60}{-80} = \frac{-3}{4}$

$p_2 = \Delta x = 80$, $q_2 = x_{max} - x_1 = 100 - 70 = 30$, $r_2 = \frac{q_2}{p_2} = \frac{30}{80} = \frac{3}{8}$

$p_3 = -\Delta y = -60$, $q_3 = y_1 - y_{min} = 25 - 10 = 15$, $r_3 = \frac{q_3}{p_3} = \frac{15}{-60} = \frac{-1}{4}$

$p_4 = \Delta y = 60$, $q_4 = y_{max} - y_1 = 100 - 25 = 75$, $r_4 = \frac{q_4}{p_4} = \frac{75}{60} = \frac{5}{4}$

Now, $t_1 = 0$, $t_2 = 1$

Again, $t_1 = \max\left(0, -\frac{3}{4}, -\frac{1}{4}\right) = 0$

$t_2 = \min\left(1, \frac{3}{8}, \frac{5}{4}\right) = \frac{3}{8}$

Here, $t_1 < t_2$ so we need to find intersection point. Let (xx_1, yy_1) and (xx_2, yy_2) be intersection point.

So, $xx_1 = x_1 + t_1 \Delta x = 70 + 0 \times 80 = 70$

$xx_2 = x_1 + t_2 \Delta x = 70 + \frac{3}{8} \times 80 = 100$

$yy_1 = y_1 + t_1 \Delta y = 25 + 0 \times 60 = 25$

$yy_2 = y_1 + t_2 \Delta y = 25 + \frac{3}{8} \times 60 = 47.5$

Hence AB line is clipped at (xx_1, yy_1) & (xx_2, yy_2)

i.e, $(70, 25)$ & $(100, 47.5)$

For line MN:

Here, $M(5,50) = (x_1, y_1)$

$N(50,120) = (x_2, y_2)$

$\Delta x = x_2 - x_1 = 50 - 5 = 45$

$\Delta y = y_2 - y_1 = 120 - 50 = 70$.

Now, $p_1 = -\Delta x = -45$, $q_1 = x_1 - x_{min} = 5 - 10 = -5$, $r_1 = \frac{q_1}{p_1} = \frac{-5}{-45} = \frac{1}{9}$

$p_2 = \Delta x = 45$, $q_2 = x_{max} - x_1 = 100 - 5 = 95$, $r_2 = \frac{q_2}{p_2} = \frac{95}{45} = \frac{19}{9}$

$p_3 = -\Delta y = -70$, $q_3 = y_1 - y_{min} = 50 - 10 = 40$, $r_3 = \frac{q_3}{p_3} = \frac{40}{-70} = -\frac{4}{7}$

$p_4 = \Delta y = 70$, $q_4 = y_{max} - y_1 = 100 - 50 = 50$, $r_4 = \frac{q_4}{p_4} = \frac{50}{70} = \frac{5}{7}$.

Now, $t_1 = \max(0, \frac{1}{9}, -\frac{4}{7}) = \frac{1}{9}$

$t_2 = \min(1, \frac{19}{9}, \frac{5}{7}) = \frac{5}{7}$

Here, $t_1 < t_2$, So we need to find intersection point. Let (x_1', y_1') & (x_2', y_2') be intersection points then;

$x_1' = x_1 + t_1 \Delta x = 5 + \frac{1}{9} \times 45 = 10$

$x_2' = x_1 + t_2 \Delta x = 5 + \frac{5}{7} \times 45 = 37$

$y_1' = y_1 + t_1 \Delta x = 50 + \frac{1}{9} \times 70 = 58$

$y_2' = y_1 + t_2 \Delta x = 50 + \frac{5}{7} \times 70 = 100$

Hence MN line is clipped at (x_1', y_1') and (x_2', y_2') i.e, $(10,37)$ & $(58,10$

Q.2. Find the clipping coordinates for a line AB where A = (10,10) and B (60,30), against window with $(x_{min}, y_{min}) = (15,15)$ and $(x_{max}, y_{max}) = (25,25)$ using Liang-Barsky line clipping algorithm.

Solution:

Here, $x_1 = 10$, $x_{min} = 15$

$y_1 = 10$, $y_{min} = 15$

$x_2 = 60$, $x_{max} = 25$

$y_2 = 30$, $y_{max} = 25$

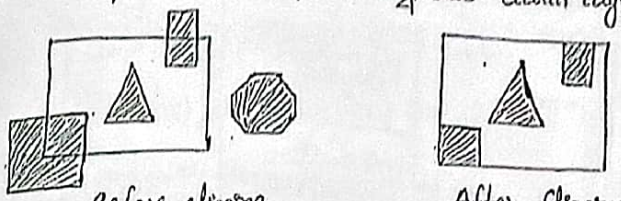
(We solve it similar way as before)

$p_1 = -\Delta x = -(x_2 - x_1) = -(60 - 10) = -50$, $q_1 = -5$, $r_1 = \frac{q_1}{p_1} = 0.1$

⇒ Same method/process that we proceed in before question. Easy solve yourself.

Polygon Clipping:

Polygon clipping is defined as the process of removing those parts of polygon that lie outside of a clipping window. A polygon can be defined as a geometric object consisting of number of vertices and an equal number of line segments called edges.



Before clipping

After Clipping

fig. polygon clipping

Sutherland - Hodgeman Polygon Clipping Algorithm:

Algorithm:

1). Construct an input vertex list $(v_0, v_1, \ldots, v_n)$ where $v_0 = v_n$.

2). Create empty output vertex list.

3). For each pair of adjcent vertices v_i and v_{i+1} perform the following inside-outside test:

a) If out-in (v_i is outside the window boundry, v_{i+1} is inside), add insertion point v_i' and v_{i+1} to the output vertex list. (new vertex)

b) If in-in (both v_i, v_{i+1} are inside the window boundry), add v_{i+1} to output vertex list.

c). If in-out (v_i is inside the window boundry and v_{i+1} is outside), add insertion point v_i to output vertex list.

d). If out-out (both v_i, v_{i+1} are outside the window boundry), add nothing to output vertex list.

4) Stop.

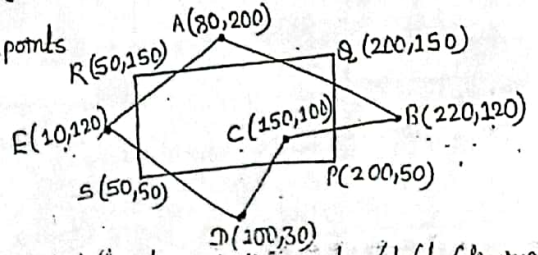
Four cases of polygon clipping against one edge:

S.No.	Case	Output
1.	Wholly inside visible region (in-in).	save endpoint.
2.	Exist visible region (in-out)	save the intersection.
3.	Wholly outside visible region (out-out)	save nothing.
4.	Enter. visible region (out-in).	save intersection and endpoint.

Q. Clip polygon ABCD against window PQRS. The co-ordinates of the polygon are A(80,200), B(220,120), C(150,100), D(100,30), E(10,120). Co-ordinates of the window are P(200,50), Q(200,150), R(50,150), S(50,50).

Solution:

Step1: Plot the points



Step2: Clipping against left edge of the window (Left Clipping)

Vertex	Case	Output Vertex	Remarks
AB	In→In	B	
BC	In→In	C	
CD	In→In	D	
DE	In→Out	D'	New vertex
EA	Out→In	E'A	New vertex

Step: 3: Right Clipping

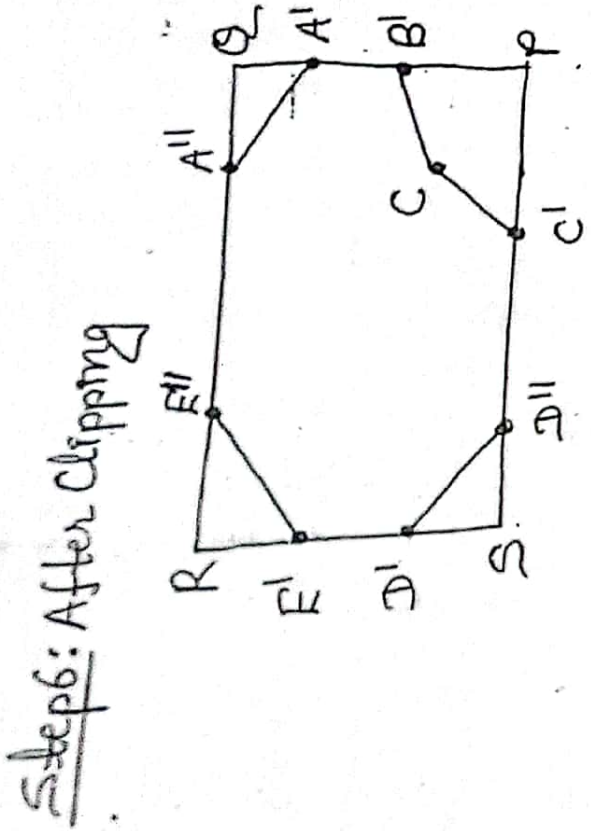
Vertex	Case	Output Vertex	Remarks
AB	In→Out	A'	New vertex
BC	Out→In	B'C	New vertex
CD	In→In	D	
DD'	In→In	D'	
D'E'	In→In	E'	
E'A	In→In	A	

Step4: Bottom Clipping

Vertex	Case	Output Vertex	Remarks
AA'	In-In	A'	
A'B'	In-In	B'	
B'C	In→In	C	
CD	In→Out	C'	New Vertex
DD'	Out→In	D''D'	New Vertex
D'E'	In→In	E'	
E'A	In→In	A	

Step5: Top Clipping

Vertex	Case	Output Vertex	Remarks
AA'	Out→In	A''A'	
A'B'	In→In	B'	New vertex
B'C	In→In	C	
CC'	In→In	C'	
CD''	In→In	D''	
D''D'	In→In	D'	
D'E'	In→In	E'	
E'A	In→Out	E''	New Vertex



Step6: After Clipping

Unit 4

Three-Dimensional Geometric Transformation:

The process of moving points in space extended from two-dimensional method by including considerations for the z coordinate is called three-dimensional geometric transformation.

Matrix representation of 3D transformation:

Let 3x3 be any matrix and transformation is applied to a point x', y', z' then it can be represented as;

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

i.e, Image = Transformation matrix × Object.

Homogenous co-ordinate representation of 3D Transformation:

Homogenous co-ordinate representation of 3D transformation has same idea as two dimensional transformations. In homogenous coordinate representation each 3D point (x, y, z) is represented as homogenous coordinate by four points (x_h, y_h, z_h, h), where, $x = \frac{x_h}{h}$ $y = \frac{y_h}{h}$ and $z = \frac{z_h}{h}$.

→ (x_h, y_h, z_h, h) represents a point at location $(x_h/h, y_h/h, z_h/h)$.
→ $(x_h, y_h, z_h, 0)$ represents a point at infinity
→ $(0, 0, 0, 0)$ is not allowed.

A three-dimensional position, expressed in homogenous coordinates, is represented as a four-element column vector. Thus, each geometric transformation operator is now a 4x4 matrix.

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ h \end{bmatrix}$$

⑧ Three Dimensional translation, rotation, scaling, reflection and shearing

ⓐ Translation: Translation is used to move a point, or a set of points linearly in space. It is same as 2D translation.

Let any point $P(x,y,z)$ is translated with translation $T(t_x, t_y, t_z)$ and $P'(x',y',z')$ is its image where,

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

Now this can be expressed as a single matrix equation. $P' = P + T$.

where, $P = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$, $P' = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}$ and $T = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$.

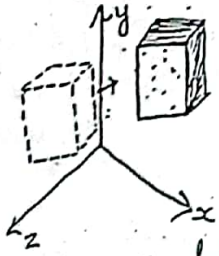


fig. 3D Translation

Homogenous Coordinates:

The homogenous coordinates for 3D translation can be expressed as:

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

On solving the RHS part of the matrix equation, we get;

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ z = t_z \\ 1 \end{bmatrix}$$

ⓑ Rotation: 3D rotation is not same as 2D rotation. In 3D rotation we have to specify the angle of rotation along with axis of rotation. We can perform 3D coordinate axes rotation as; Z-axis rotation (this is same as 2D rotation), Y-axis rotation (Yaw) and X-axis rotation (Pitch).

Z-axis rotation (Roll):

In this we ignore Z element now it becomes the same case as if we were rotating the 2D point $\langle x, y \rangle$ through angle θ. Z-axis rotation is same as the origin about the 2D for which we have the derived matrices already,

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

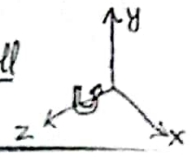


fig. Roll

Homogenous representation is:

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

i.e, $P' = R_z(\theta) \cdot P$

Y-axis rotation (Yaw)

The equations for Y-axis rotation are;

y element ignored

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

Homogenous representation is;

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

i.e, $P' = R_y(\theta) \cdot P$

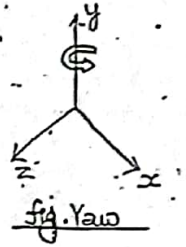


fig. Yaw

X-axis rotation (Pitch)

The equations for X-axis rotation are;

x element ignored

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

Homogenous representation is;

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

i.e, $P' = R_x(\theta) \cdot P$

fig. Pitch

ⓒ Scaling:

Coordinate transformations for scaling relative to the origin are.

$$x' = x \cdot S_x$$
$$y' = y \cdot S_y$$
$$z' = z \cdot S_z$$

i) Uniform Scaling: Original shape is preserved. ($S_x = S_y = S_z$).

ii) Differential Scaling: Original shape is not preserved. ($S_x \neq S_y \neq S_z$).

iii) Scaling relative to coordinate Origin: Scaling transformation of a position $P = (x, y, z)$ relative to the coordinate origin can be written as:

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

iv) Scaling with respect to a selected fixed position:-
Scaling with respect to a selected fixed position (x_f, y_f, z_f) can be represented with the following transformation sequence:

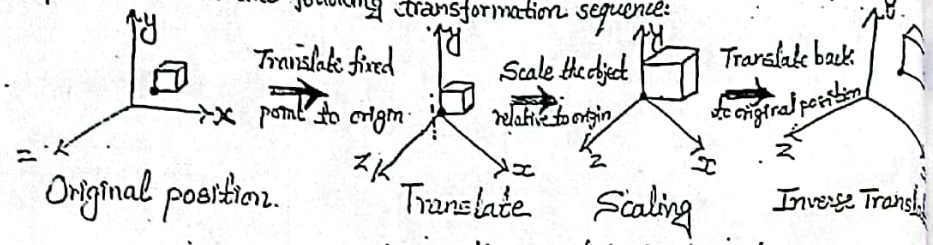


fig. 3D scaling with respect to fixed point

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

Where composite matrix for scaling is

$$\therefore T(x_f, y_f, z_f) \cdot S(S_x, S_y, S_z) \cdot T(-x_f, -y_f, -z_f) = \begin{bmatrix} S_x & 0 & 0 & (1-S_x)x_f \\ 0 & S_y & 0 & (1-S_y)y_f \\ 0 & 0 & S_z & (1-S_z)z_f \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

@Reflection:

i) The matrix representation for this reflection of points relative to X axis (i.e, Z-X plane) is $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

ii) The matrix representation for this reflection of points relative to the Y-axis (i.e, Y-Z plane) is $\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

iii) The matrix representation for this reflection of points relative to the Z-axis (i.e, X-Y plane) is: $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

@Shearing:

Y-axis shear

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

Parameters 'a' and 'c' can be assigned and real values.

X-axis shear

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

Parameters 'b' and 'c' can be assigned and real values.

Three Dimensional Viewing:

@Viewing Pipeline: The steps for viewing pipeline (i.e. view of 3D scene) are analogous (similar) to the process of taking photograph by a camera. For a snapshot, we need to position the camera at a particular point in space and then need to decide camera orientation. Finally when we snap the shutter, the scene is cropped to the size of window of the camera and the light from the visible surfaces is projected into the camera film.

The viewing-coordinate system is used in computer graphics packages as a refrence for specifying the observer viewing position of the projection plane.

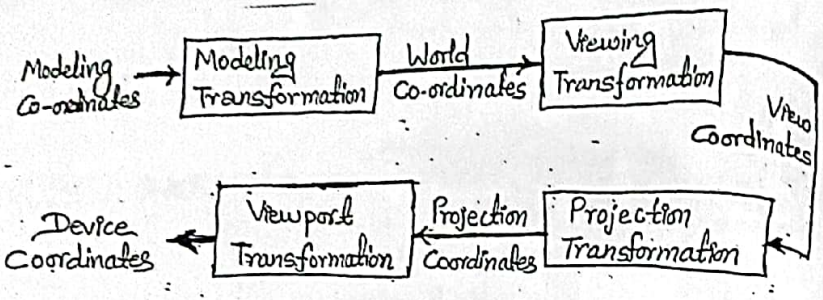


fig. 3D viewing pipeline

(B) World to screen viewing transformation:

In 3D viewing pipeline, the first step after a scene has been constructed is to transfer object descriptions to the viewing-coordinate refrence frame; Conversion of object descriptions from world to viewing coordinates is equivalent to a transformation that overlaps the viewing refrence frame onto the world frame using the basic geometic translate-rotate operations:

1) Translate the viewing coordinate origin to the origin of the world coordinate system.
2) Apply rotations to align the x-view, y-view and z-view axes with the world xw, yw and zw -axes respectively.

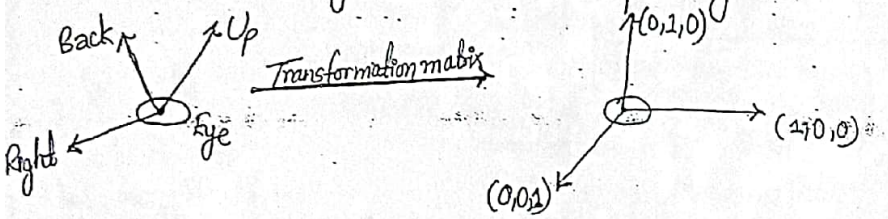


fig. Transformation matrix maps camera bias to canonical vectors in viewing co-ordinate system.

(A) Projection Concept:

[Projection types and concept important topic]

Projection is the process of representing n-dimensional object into n-1 dimension is known as projection. If the projection is in case of 3D then it is the process of converting a 3D object into a 2D object. It is also defined as mapping or transformation of the object in projection plane or view plane.

The mapping is determined by a projection line called the projector that passes through point and intersects the view plane.

Taxonomy of Projection (OR Projection Types):

Mainly there are two types of projections: Parallel projection and perespective projection. Orthographic projection is also a major projection which is one of the type of parallel projection. Following figure provides taxonomy of the families of parallel and perespective projections.

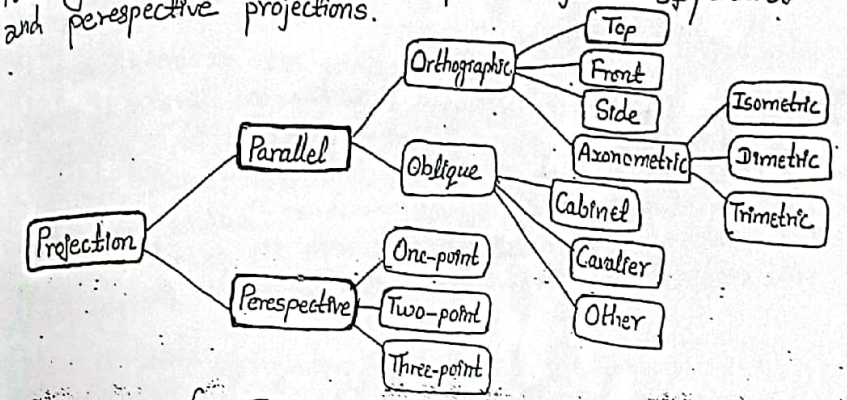


fig. Taxonomy of Projection:

(a) Parallel Projection:

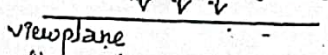
Parallel projection discards z-coordinate and parallel lines from each vertex on the object that are extended until they intersect the view plane. In parallel projection, we specify a direction of projection instead of center of projection.

In parallel projection, the distance from the center of projection to project plane is infinite. In this type of projection,

we connect the projected vertices by line segments which correspond to connections on the original object.

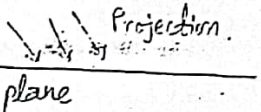
Parallel projections are less realistic, but they are good for exact measurements. In this type of projections, parallel lines remain parallel and angles are not preserved. It preserves relative proportion of 3D object hence it is used in mathematical drawings.

i) Orthographic Projection → In orthographic projection the direction of projection is normal to the projection of the plane.



There are three types of orthographic projection as:
- → Front projection
- → Top projection
- → Side projection

ii) Oblique Projection → In oblique projection, the direction of projection is not normal to the projection of plane. In oblique projection we can view the object better than orthographic projection. There are two types of oblique projections: Cavalier and Cabinet. The ~~cavalier~~ Cavalier makes 45° angle with the projection plane, the Cabinet projection makes 63.4° angle with the projection plane.



The transformation matrix for producing any parallel projection onto the xy-plane is written as;

$$M_{parallel} = \begin{bmatrix} 1 & 0 & L_1\cos\phi & 0 \\ 0 & 1 & L_1\sin\phi & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\therefore \begin{bmatrix} x_p \\ y_p \\ z_p \\ w_p \end{bmatrix} = \begin{bmatrix} 1 & 0 & L_1\cos\phi & 0 \\ 0 & 1 & L_1\sin\phi & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

where, L = length of line.
L_1 = value of L when $z=1$.
ϕ = angle

If projection line is perpendicular to projection plane then $L_1 = 0$.

i) Axonometric Orthographic Projection → Orthographic projections that show more than one side of an object are called axonometric orthographic projections. There are three ~~axonometric~~ axonometric projections they are:

a) Isometric → In isometric projection the direction of projection makes equal angles with all of the three principal axes.

b) Dimetric → The direction of projection makes equal angles with exactly two of the principal axes.

c) Trimetric → The direction of projection makes unequal angles with the three principal axes.

ii) What is Center of Projection (COP)? → meaning of projectors

⇒ The projectors (i.e. light rays reflecting from 3D object onto 2D plane) convergence point is called center of projection (COP).

→ If projectors are parallel then COP lies at infinity. In this case, projection is denoted by direction of projection (DOP). Normally COP denotes human eye or camera position.

(b) Perespective Projection:

In perespective projection, the distance from the center of projection to project plane is finite and the size of the object varies inversely with distance which looks more ~~retr~~ realistic. The distance and angles are not preserved and parallel lines do not remain parallel. Instead they all converge at a single point called center of projection (COP). There are 3 types of perespective projections which are as follows:

i) One point → One point perespective projection is simple to draw.

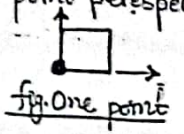


fig. One point

ii) Two point → It gives better impression of depth.

fig. two point

iii) Three point → It is most difficult to draw.

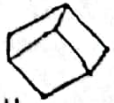
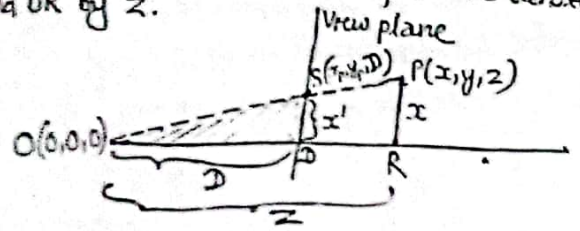


fig. three point

Computing the Perspective Projection:

Let any object $P(x,y,z)$ is being projected having origin at $O(0,0,0)$ at viewplane ~~=~~ as shown in figure below: let distance of OD be denoted by D, DS by x' and QR by z.



Now, we calculate x', y' in terms of x & y using the property for two similar triangles that the ratio of sides of two similar triangles remain always maintained. In above figure triangle OSD and triangle OPR are two similar triangles.

From figure $\frac{x_p}{D} = \frac{x}{z}$ (Using property for similar triangle).

i.e. $x_p = \frac{xD}{z} \Rightarrow x' = x$ (D≠z being common side of similar triangles so, D=z)

Similarly $\frac{y_p}{D} = \frac{y}{z}$

i.e. $y_p = \frac{yD}{z} \Rightarrow y' = y$.

& $z_p = D$ (Since common side of similar triangles).
$\Rightarrow z' = z$

For homogenous coordinates we have

$w_p = 1$ (Since in homogenous co-ordinate system 4th coordinate W=1)

$\Rightarrow w_p = \frac{z}{D}$ (Since z≠D being common side of similar triangle).

$\Rightarrow \frac{z}{D} = 1$

Now we can represent this in matrix form using homogenous coordinates as follows:-

$$\begin{bmatrix} x_p \\ y_p \\ z_p \\ w_p \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{D} & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

⇒ When center of projection is on the x-axis $M_{PER} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{D} & 0 & 0 & 1 \end{bmatrix}$

⇒ When center of projection is on the y-axis $M_{PER} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{1}{D} & 0 & 1 \end{bmatrix}$

Unit-5

3D Objects Representation

Representing Surfaces: Representation schemes for solid objects are often divided into two broad categories: Boundry representations (B-reps) which describes three-dimensional objects as a set of surfaces that seperate the object interior from the environment. This is one of the representation scheme. Polygon facets and spline patches are it's examples.

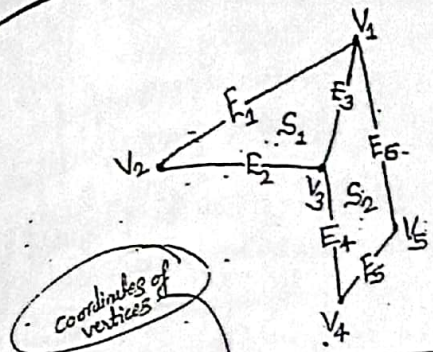
Another representation scheme is Space-partioning representation which are used to describe interior properties by partitioning the spatial region containing an object into a set of small, non-overlapping and contigious solids. A common space partitioning description for a 3D object is an octree representation.

Polygon Surfaces:-

The most commonly used boundry representation for a 3D graphics object is a set of surface polygons that enclose the interior object. Many graphics systems store all object descriptions as set of surface polygons. This simplifies and speeds up the surface rendering and display of objects.

1) Polygon Tables:

Polygon table is the specification of polygon surfaces using vertex coordinates and other attributes. Polygon tables or Polygon data tables can be organized into two groups: geometric tables and attribute tables. For storing geometric data we create three lists; a vertex table, an edge, and a polygon table. Co-ordinate values for each vertex in the object are stored in vertex table. The edge table contains pointers back into the vertex table to identify the vertices for each polygon edge. And the polygon table contains pointers back into the edge table to identify the edges for each polygon. Attribute table contains qualatitative properties like degree of transparency, surface reflectivity etc.



coordinates of vertices

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

Edge Table
$E_1 : V_1, V_2$
$E_2 : V_2, V_3$
$E_3 : V_3, V_1$
$E_4 : V_3, V_4$
$E_5 : V_4, V_5$
$E_6 : V_5, V_1$

Polygon-Surface table:
$S_1 : E_1, E_2, E_3$
$S_2 : E_3, E_4, E_5, E_6$

fig. Geometric data table representation for two adjcent polygon surfaces, formed with six edges and five vertices.

Edge table including pointers to polygon table.
$E_1 : V_1, V_2, S_1$
$E_2 : V_2, V_3, S_1$
$E_3 : V_3, V_1, S_1, S_2$
$E_4 : V_3, V_4, S_2$
$E_5 : V_4, V_5, S_2$
$E_6 : V_5, V_1, S_2$

Some consistency checks of the geometric data table are as follows:-

→ Every vertex is listed as an endpoint for at-least 2 edges.
→ Every edge is a part of at-least one polygon.
→ Every polygon is closed.

ii) Polygon Mesh:

Using a set of connected polygonally bounded planar surfaces to represent an object, which may have curved surfaces or curved edges, is called polygon mesh. The wireframe of such object can be displayed quickly to give general indication of the surface structure.

Realistic renderings can be produced by interpolating shading patterns across the polygon surfaces to eliminate or reduce the presence of polygon edge boundries. Fast hardware-implemented polygon renders are capable of displaying upto 1,000,000 or more shaded triangles per second, including the application of surface texture and special lighting effects. Common types of polygon meshes are triangle strip and quadrilateral mesh.

fig. A triangle strip formed with 11 triangles connecting 13 vertices.

fig. A quadrilateral mesh connecting 12 quadrilaterals constructed from a 5 by 4 input vertex array.

iii) Plane Equations:

The equation for a plane surface is $Ax+By+Cz+D=0$, where (x,y,z) is any point on the plane, and the coefficients A, B, C and D are constants describing the spatial properties of the plane.

We can obtain values of A, B, C and D by solving a set of three plane equations using the coordinate values for three non collinear points in the plane. For that we can select three successive polygon vertices (x_1,y_1,z_1), (x_2,y_2,z_2) and (x_3,y_3,z_3) and solve the following set of simultaneous linear plane equations for the ratios $\frac{A}{D}$, $\frac{B}{D}$ and $\frac{C}{D}$.

$$\frac{A}{D}x_k+\frac{B}{D}y_k+\frac{C}{D}z_k=-1 \quad ; \text{ for } k=1,2,3.$$

The solution for this set of equations can be obtained in determinant form, using Cramer's rule as:

$$A=\begin{vmatrix} 1 & y_1 & z_1 \\ 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \end{vmatrix} \quad B=\begin{vmatrix} x_1 & 1 & z_1 \\ x_2 & 1 & z_2 \\ x_3 & 1 & z_3 \end{vmatrix} \quad C=\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}$$

$$\text{and } D=-\begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix}$$

Expanding the determinants, we can write the calculations for the plane coefficients in the form;

$$A=y_1(z_2-z_3)+y_2(z_3-z_1)+y_3(z_1-z_2)$$
$$B=z_1(x_2-x_3)+z_2(x_3-x_1)+z_3(x_1-x_2)$$
$$C=x_1(y_2-y_3)+x_2(y_3-y_1)+x_3(y_1-y_2)$$
$$D=-x_1(y_2z_3-y_3z_2)-x_2(y_3z_1-y_1z_3)-x_3(y_1z_2-y_2z_1)$$

⇒ We can identify the point as either inside or outside the plane surface according to the sign (negative or positive) of $Ax+By+Cz+D$

If $Ax+By+Cz+D<0$, then the point (x,y,z) is inside the surface.
If $Ax+By+Cz+D>0$, then the point (x,y,z) is outside the surface.

These inequality tests are valid in a right handed Cartesian system, provided the plane parameters A, B, C and D were calculated using vertices selected in a counter clockwise order when viewing the surface in an outside-to-inside direction.

Note: If $Ax+By+Cz+D\neq 0$, then it means the point is not on the plane.

iv) Surface normal and Spatial orientation of surfaces:-

A normal is a term used in computer graphics to describe the orientation of a geometric object at a point on the surface. The normal to a surface at point P can be seen as the vector perpendicular to a plane tangent to the surface at P. At the same time, the direction of this vector determines the orientation of the surface. In the case of polygons, this direction is usually determined by the right hand rule. Normal plays an important role in shading where they are used to compute the brightness of the objects.

Spatial Orientation of a polygon face is the vertex coordinates values and the equations of the polygon surfaces. The general equation of a plane containing a polygon is

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

where (x, y, z) is any point on plane.

⊛ Wireframe Representation:-

If the object is defined only by a set of nodes, and a set of connecting the nodes, then the resulting object representation is called a wireframe model. In this method, a 3D object is represented as a list of straight lines each of which is represented by its two end points, (x_1, y_1, z_1) and (x_2, y_2, z_2). This method only shows skeletal structure of objects.

A wireframe model consists of edges, vertices and polygons. The edges maybe curved or straight line segments.

Advantages and Disadvantages:-

Wireframe model are used in engineering applications. They are easy to construct. If they are compound of straight lines they are easy to clip and manipulate.

But for building realistic models, we must use a very large number of polygons to achieve the illusions of roundness and smoothness.

⊛ Blobby Objects:-

The objects that do not maintain a fixed shape but change their surface characteristics in certain motions are known as blobby objects. For example: molecular structures, water droplets, melting objects etc. Several models have been developed for representing blobby objects as distribution functions over a region of space. One way is to use Gaussian density function. Other methods for generating blobby objects use quadratic density function.

#Representing Curves:

In computer graphics, we often need to draw different types of objects onto the screen. Objects are not flat all the time and we need to draw curves many times to draw an object.

A curve is an infinitely large set of points. Curves are broadly classified into three categories- explicit, implicit and parametric curves.

i) Implicit curves → Implicit curve representations define the set of points on a curve employing a procedure that can test to see if a point is on the curve. Usually, an implicit curve is defined by an implicit function of the form:

$$f(x, y) = 0 \quad \text{(In 2D)}$$
$$f(x, y, z) = 0 \quad \text{(In 3D)}.$$

A common example is the circle, whose implicit representation is;

$$x^2 + y^2 - R^2 = 0.$$

ii) Explicit curves → A mathematical function $y = f(x)$ can be plotted as a curve. Such a function is the explicit representation of curve. The explicit representation is not general, since it cannot represent vertical lines and is also single-valued. For each value of x, only a single value of y is normally computed by the function.

Parametric Curves → Curves having parametric form are called parametric curves. A two-dimensional parametric curve has the following form:

$$P(t) = f(t), g(t) \quad \text{or} \quad P(t) = x(t), y(t).$$

The functions of f and g becomes the (x,y) coordinates of any point on the curve, and the points are obtained when the parameter t is varied over a certain interval [a,b] normally [0,1].

@ Parametric cubic curves:

Once we decide parametric polynomial curves, we must choose the degree of the curve. If we choose a high degree, there is more danger that the curve will become rougher. On other hand, if we pick too low degree, we may not have enough parameters with which to work.

Algebraic representation of parametric curves

→ Parametric linear curve: $p(u) = au + b$

$$x = a_x u + b_x$$
$$y = a_y u + b_y$$
$$z = a_z u + b_z.$$

→ Parametric cubic curve: $p(u) = au^3 + bu^2 + cu + d$

$$x = a_x u^3 + b_x u^2 + c_x u + d_x$$
$$y = a_y u^3 + b_y u^2 + c_y u + d_y$$
$$z = a_z u^3 + b_z u^2 + c_z u + d_z.$$

Advantages

→ More degrees of freedom.
→ Directly transformable.
→ Dimension independent
→ No infinite slope problems.
→ Seperates dependent and independent variables.
→ Inherently bounded.
→ Easy to express in vector and matrix form.
→ Common form for many curves and surfaces.

ⓑ Spline Representation:

Spline means a flexible strip used to produce a smooth curve through a designated set of points. Several small weights are distributed along the length of the strip to hold it in position on the drafting table as the curve is drawn.

We can mathematically describe such a curve with a piecewise cubic polynomial function i.e. spline curves. Then a spline surface can be described with 2 sets of orthogonal spline curves. Splines are used in graphics applications to design curve and surface shapes.

ⓒ Cubic Spline interpolation:-

This method gives an interpolating polynomial that is smoother and has smaller error than some other interpolating polynomials such as Lagrange polynomial and Newton polynomial. Cubic polynomials provide a reasonable compromise between flexibility and speed of computation. Cubic spline requires less calculations compared to higher order polynomials and consume less memory. They are also more flexible for modeling arbitrary curve shape.

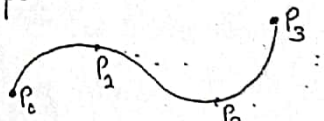


fig: Interpolation with cubic splines between 4 control points.

Suppose we have n+1 control points specified with coordinates

$$P_k = (x_k, y_k, z_k), \quad k = 0, 1, 2, \ldots n.$$

The parametric cubic polynomial that is to be fitted between each pair of control points with the following set of equations.

$$x(u) = a_x u^3 + b_x u^2 + c_x u + d_x \quad (0 \le u \le 1)$$
$$y(u) = a_y u^3 + b_y u^2 + c_y u + d_y$$
$$z(u) = a_z u^3 + b_z u^2 + c_z u + d_z$$

Values for coefficients a,b,c,d are determined by setting enough boundary conditions at control-point positions.

④ Hermite Curves:-

Hermite curves are very easy to calculate but also very powerful. They are used to smoothly interpolate between key-points. Hermite curves work in any number of dimensions. To calculate hermite curve we need the following vectors:

P1: the start point of the curve.
T1: The tangent to how the curve leaves the start point.
P2: The endpoint of the curve.
T2: The tangent to how the curve meets the endpoint.

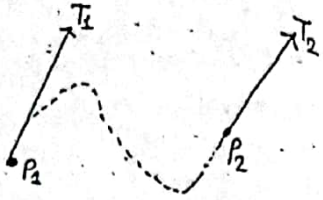


fig. Hermite curve

Matrix form of Hermite Curve:-

$$S = \begin{bmatrix} s^3 \\ s^2 \\ s \\ 1 \end{bmatrix} \quad C = \begin{bmatrix} P1 \\ P2 \\ T1 \\ T2 \end{bmatrix} \quad h = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Vector S: The interpolation-point and its powers upto 3.
Vector C: The parameters of our hermite curve.
Matrix h: The matrix form of the 4 hermite polynomials.

⇒ To calculate a point on the curve we build the vector S, multiply it with the matrix h and then multiply with C.
i.e, $P = S \times h \times C$.

⑤ Bezier Curves and Surfaces:

This spline approximation method was developed by French engineer Pierre Bezier. Bezier splines have a number of properties that make them highly useful and convinent for curve and surface design. They are also easy to implement. For these reasons Bezier splines are widely available in various CAD systems.

Bezier Curves:

This method employs control points and produces an approximating curve. Bezier curve can be specified with boundry conditions, with a characterizing matrix, or with blending functions. For general Bezier curves, the blending function specification is the most convinent.

Suppose we are given $n+1$ control-point positions: $P_k = (x_k, y_k, z_k)$, with k varying from 0 to n. These coordinate points can be blended to produce the following position vector $P(u)$, which describes the path of an approximating Bezier polynomial function between P_0 and P_n.

$$P(u) = \sum_{k=0}^{n} P_k \, BEZ_{k,n}(u) \quad , \quad 0 \le u \le 1$$

The Bezier blending functions $BEZ_{k,n}(u)$ are the Bernstein polynomial.

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

where $C(n,k)$ are the binomial coefficients;

$$C(n,k) = \frac{n!}{k!\,(n-k)!}$$

The equation $P(u) = \sum_{k=0}^{n} P_k \, BEZ_{k,n}(u)$, where $0 \le u \le 1$ represents a set of three parametric equations for individual curve condition.

$$x(u) = \sum_{k=0}^{n} x_k \, BEZ_{k,n}(u)$$

$$y(u) = \sum_{k=0}^{n} y_k \, BEZ_{k,n}(u)$$

$$z(u) = \sum_{k=0}^{n} z_k \, BEZ_{k,n}(u)$$

Two points generate simple Bezier, three points generate a parabola, four points a cubic curve. i.e, $(n-1)$ degree of polynomial equation for the n control points.

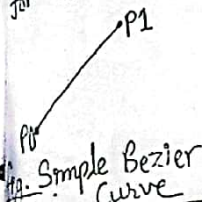


fig: Simple Bezier curve

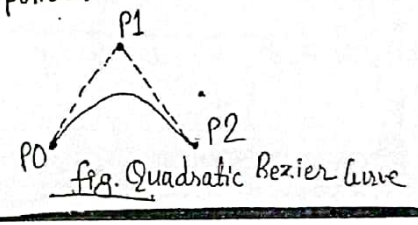


fig. Quadratic Bezier Curve

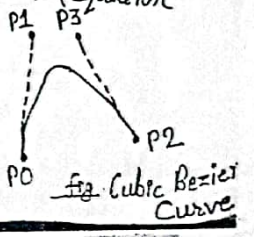


fig. Cubic Bezier Curve

Properties of Bezier Curve:

i) It always passes through the first and last control points. That is, the boundry conditions at the two ends of the curve are;

$$P(0) = P_0$$
$$P(1) = P_n.$$

ii) It lies within the convex hull (convex polygon boundry) of the control points.

iii) The degree of polynomial defining the curve segment is one less than the number of defining polygon point. Therefore, for 4 control points, the degree of polynomial is 3 i.e, cubic polynomial.

iv) A Bezier curve generally follows the shape of the defining polygon.

v) The direction of the tangent vector at the end points is same as that of the vector determined by first and last segments.

Example: Construct the Bezier curve of order 3 and with 4 polygon vertices A(1,1), B(2,3), C(4,3) and D (6,4).

Solution:

The equation for the Bezier curve is given as.

$$P(u) = (1-u)^3 P_1 + 3u(1-u)^2 P_2 + 3u^2(1-u)P_3 + u^3 P_4, \quad \text{for } 0 \le u \le 1.$$

where, $P(u)$ is the point on curve P_1, P_2, P_3, P_4.

$0 \le u \le 1$ condition given in question

Let us take, $u = 0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}$

$\therefore P(0) = P_1 = (1,1)$: i.e, coordinates 1st given in Q

$$P\left(\frac{1}{4}\right) = \left(1-\frac{1}{4}\right)^3 P_1 + 3\times\frac{1}{4}\left(1-\frac{1}{4}\right)^2 P_2 + 3\left(\frac{1}{4}\right)^2\left(1-\frac{1}{4}\right)P_3 + \left(\frac{1}{4}\right)^3 P_4$$

$$= \frac{27}{64}(1,1) + \frac{27}{64}(2,3) + \frac{9}{64}(4,3) + \frac{1}{64}(6,4)$$

$$= \left(\frac{27}{64}\times 1 + \frac{27}{64}\times 2 + \frac{9}{64}\times 4 + \frac{1}{64}\times 6,\ \frac{27}{64}\times 1 + \frac{27}{64}\times 3 + \frac{9}{64}\times 3 + \frac{1}{64}\times 4\right)$$

$$= \left(\frac{123}{64}, \frac{159}{64}\right)$$

$$= (1.9218, 2.1718).$$

$$P\left(\frac{1}{2}\right) = \left(1-\frac{1}{2}\right)^3 P_1 + 3\times\frac{1}{2}\left(1-\frac{1}{2}\right)^2 P_2 + 3\left(\frac{1}{2}\right)^2\left(1-\frac{1}{2}\right)P_3 + \left(\frac{1}{2}\right)^3 P_4.$$

$$= \frac{1}{8}(1,1) + \frac{3}{8}(2,3) + \frac{3}{8}(4,3) + \frac{1}{8}(6,4)$$

$$= \left(\frac{1}{8}\times 1 + \frac{3}{8}\times 2 + \frac{3}{8}\times 4 + \frac{1}{8}\times 6,\ \frac{1}{8}\times 1 + \frac{3}{8}\times 3 + \frac{3}{8}\times 3 + \frac{1}{8}\times 4\right)$$

$$= \left(\frac{25}{8}, \frac{23}{8}\right)$$

$$= (3.125, 2.875).$$

$$P\left(\frac{3}{4}\right) = \left(1-\frac{3}{4}\right)^3 P_1 + 3\times\frac{3}{4}\left(1-\frac{3}{4}\right)^2 P_2 + 3\left(\frac{3}{4}\right)^2\left(1-\frac{3}{4}\right)P_3 + \left(\frac{3}{4}\right)^3 P_4.$$

$$= \frac{1}{64}(1,1) + \frac{9}{64}(2,3) + \frac{27}{64}(4,3) + \frac{27}{64}(6,4)$$

$$= \left(\frac{1}{64}\times 1 + \frac{9}{64}\times 2 + \frac{27}{64}\times 4 + \frac{27}{64}\times 6,\ \frac{1}{64}\times 1 + \frac{9}{64}\times 3 + \frac{27}{64}\times 3 + \frac{27}{64}\times 4\right)$$

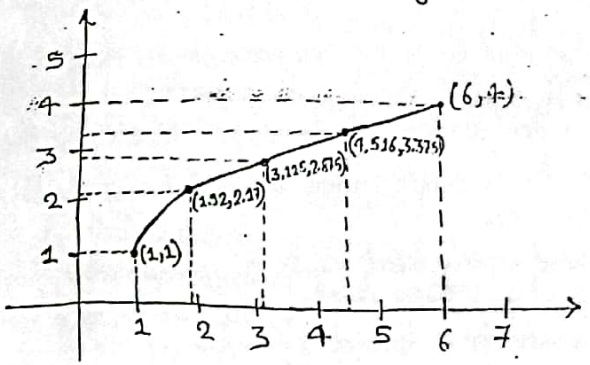
$$= \left(\frac{289}{64}, \frac{217}{64}\right)$$

$$= (4.5156, 3.375).$$

i.e, last coordinate given in question

$\therefore P(1) = P_3 = (6,4)$.

The graph below shows that calculated points of the Bezier curve and curve passing through it.



Bezier Surface:

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

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

with $P_{j,k}$ specifying the location of the $(m+1)$ by $(n+1)$ control points. A smooth transition is assured by establishing both zero-order and first-order continuity at the boundry line. Zero-order continuity is obtained by matching control points at the boundry. First-order continuity is obtained by choosing control points along a straight line across the boundry and by maintaining a constant ratio of collinear line segments for each set of specified control points across section boundaries

(f) B-Spline Curves and Surface:

These are the most widely used class of approximating splines. B-Spline have following two advantages over Bezier Splines:-

1. The degree of a B-spline can be set independently of the number of control points.
2. B-splines allow local control over the shape of a spline curve or surface.

→ The disadvantage is that B-splines are more complex than Bezier splines.

B-Spline Curves:- The designation "B" stands for Basis, so the full name of this approach is basis spline which contains the Bernstein basis as a special case.

There is most widely used class of approximating splines. B-spline has a general expression for the calculation of coordinate positions along a curve in a blending function as:

$$P(u) = \sum_{k=0}^{n} B_i N_{i,k}(u), \quad u_{min} \le u \le u_{max}, \quad 2 \le k \le n+1;$$

where B_i are the position vectors of the $n+1$ defining polygon vertices and the $N_{i,k}$ are the normalized B-spline basis functions. For the i^{th} normalized B-spline basis function of order k, the basis function $N_{i,k}(u)$ are defined as

$$N_{i,1}(u) = \begin{cases} 1 & \text{if } x_i \le u < x_{i+1} \\ 0 & \text{otherwise.} \end{cases}$$

And: $$N_{i,k}(u) = \frac{(u - x_i) N_{i,k-1}(u)}{x_{i+k-1} - x_i} + \frac{(x_{i+k} - u) N_{i+1,k-1}(u)}{x_{i+k} - x_{i+1}}.$$

The values of x_i are the elements of a knot vector satisfying the relation $x_i \le x_{i+1}$. The parameter u varies from u_{min} to u_{max} along the curve $P(u)$. The choice of knot vector has a significant influence on the B-spline basis functions $N_{i,k}(u)$ and hence on the resulting B-spline curve. There are three types of knot vector: uniform, open uniform and on uniform.

→ In uniform knot vector, individual values are equally spaced. For example [0 1 2 3 4].

- For a given order k, uniform knot vectors give periodic uniform basis functions;

$$N_{i,1}(u) = N_{i-1,k}(u-1) = N_{i+1,k}(u+1)$$

→ An open uniform knot vector has multiplicity of knot values at the ends equal to the order k of the B-spline basis function. Internal ~~values are~~ knot values are equally spaced. Examples are:-

k=2 [0 0 1 2 3 3]

k=3 [0 0 0 1 2 3 3 3]

k=4 [0 0 0 0 1 2 2 2 2]

Generally an open uniform knot vector is given by, $x_i = 0$ where $1 \le i \le k$.

$x_i = i - k$ where $k+1 \le i \le n+1$

$x_i = n - k + 2$ where $n+2 \le i \le n+k+1$

The resulting B-spline curve is a Bezier curve.

Properties of B-spline curve:-

i) The degree of B-spline polynomial is independent on the number of vertices of defining polygon.

ii) The maximum order of the curve is equal to the number of vertices of defining polygon.

iii) Each basis function is positive or zero for all parameter values.

iv) Each basis function has precisely one maximum value, except for k=1.

v) The sum of B-spline basis functions for any parameter value is 1.

vi) B-spline allows local control over the curve surface because each vertex affects the shape of a curve.

Example:- Construct the B-spline curve of order 4 and with 4 polygon vertices A(1,1), B(2,3), C(4,3) and D(6,2).

Solution:

Here, n=3 and k=4, we have open uniform knot vector as

X = [0 0 0 0 1 1 1 1] and we have basis functions for various parameters are as follows:-

$0 \le u < 1$

$N_{4,1}(u) = 1$; $N_{i,1}(u) = 0,\ i \neq 4$

$N_{3,2}(u) = (1-u)$; $N_{4,2}(u) = u$, $N_{i,2}(u) = 0,\ i \neq 3,4$.

$N_{2,3}(u) = (1-u)^2$; $N_{3,3}(u) = 2u(1-u)$;

$N_{4,3}(u) = u^2$; $N_{i,3}(u) = 0,\ i \neq 2,3,4$.

$N_{1,4}(u) = (1-u)^3$; $N_{2,4}(u) = u(1-u)^2 + 2u(1-u)^2 = 3u(1-u)^2$;

$N_{3,4}(u) = 2u^2(1-u) + (1-u)u^2 = 3u^2(1-u)$; $N_{4,4}(u) = u^3$.

The parametric B-Spline is

$$P(u) = AN_{1,4}(u) + BN_{2,4}(u) + CN_{3,4}(u) + DN_{4,4}(u).$$

$$\therefore P(u) = (1-u)^3 A + (1-u)^2 B + (1-u) C + u^3 D.$$

Let us take, $u = 0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1$.

$\therefore P(0) = A = (1,1)$.

$$P\left(\frac{1}{4}\right) = \left(1-\frac{1}{4}\right)^3 A + 3\times\frac{1}{4}\left(1-\frac{1}{4}\right)^2 B + 3\left(\frac{1}{4}\right)^2\left(1-\frac{1}{4}\right) C + \left(\frac{1}{4}\right)^3 D.$$

$$= \frac{27}{64}(1,1) + \frac{27}{64}(2,3) + \frac{9}{64}(4,3) + \frac{1}{64}(6,2)$$

$$= \left(\frac{27}{64}\times 1 + \frac{27}{64}\times 2 + \frac{9}{64}\times 4 + \frac{1}{64}\times 6,\ \frac{27}{64}\times 1 + \frac{27}{64}\times 3 + \frac{9}{64}\times 3 + \frac{1}{64}\times 2\right)$$

$$= \left(\frac{123}{64}, \frac{137}{64}\right)$$

$$= (1.9218, 2.14).$$

$$P\left(\frac{1}{2}\right) = \left(1-\frac{1}{2}\right)^3 A + 3\times\frac{1}{2}\left(1-\frac{1}{2}\right)^2 B + 3\left(\frac{1}{2}\right)^2\left(1-\frac{1}{2}\right) C + \left(\frac{1}{2}\right)^3 D$$

$$= \frac{1}{8}(1,1) + \frac{3}{8}(2,3) + \frac{3}{8}(4,3) + \frac{1}{8}(6,2)$$

$$= \left(\frac{1}{8}\times 1 + \frac{3}{8}\times 2 + \frac{3}{8}\times 4 + \frac{1}{8}\times 6,\ \frac{1}{8}\times 1 + \frac{3}{8}\times 3 + \frac{3}{8}\times 3 + \frac{1}{8}\times 2\right)$$

$$= \left(\frac{25}{8}, \frac{21}{8}\right)$$

$$= (3.125, 2.625).$$

$$P\left(\frac{3}{4}\right) = \left(1-\frac{3}{4}\right)^3 A + 3\times\frac{3}{4}\left(1-\frac{3}{4}\right)^2 B + 3\left(\frac{3}{4}\right)^2\left(1-\frac{3}{4}\right) C + \left(\frac{3}{4}\right)^3 D$$

$$= \frac{1}{64}(1,1) + \frac{9}{64}(2,3) + \frac{27}{64}(4,3) + \frac{27}{64}(6,2)$$

$$= \left(\frac{1}{64}\times 1 + \frac{9}{64}\times 2 + \frac{27}{64}\times 4 + \frac{27}{64}\times 6,\ \frac{1}{64}\times 1 + \frac{9}{64}\times 3 + \frac{27}{64}\times 3 + \frac{27}{64}\times 2\right)$$

$$= \left(\frac{289}{64}, \frac{163}{64}\right)$$

$$= (4.5156, 2.5468)$$

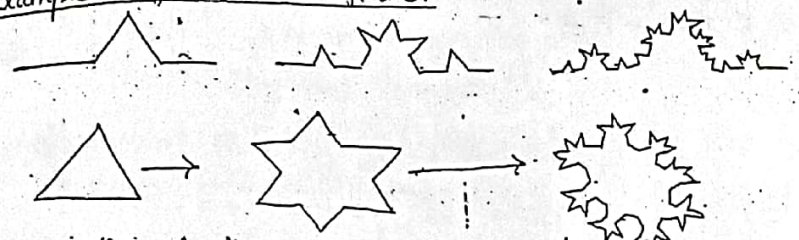
$\therefore P(1) = D = (6,2)$

Hence we can draw graph in similar way as we did finally in Bezier spline.

⑧. Fractals and its applications:-

Those objects which are self-similar at all resolutions are called fractal objects. Most of the natural objects such as trees, mountains, coastlines etc. are considered as fractal objects because no matter how far or how close one ~~clicks~~ looks at them, they always appear somewhat similar. Fractal objects can also be generated recursively by applying the same transformation fraction to an object.
e.g. Scale down + rotate + translate.

For example, a fractal Snowflake:



The name "Fractal" comes from its property: fractional dimension.

Applications:

i) Fractals are used to predict or analyze various biological phenomena such as growth pattern of bacteria.
ii) Fractals are used to capture images of complex structures such as clouds.
iii) Speaking of imaging is one of the most important use of fractals with regards to image compressing.
iv) It is widely used in image synthesis and computer animation.

⑨. Quadric Surface:-

A frequently used class of objects are the quadric surfaces, which are described with second-degree equations (quadratics). They include spheres, ellipsoids, tori, paraboloids and hyperboloids. Quadric surfaces; particularly spheres and ellipsoids are common elements of graphics scenes.

Sphere: A spherical surface with radius r centered on the coordinates origin is defined as the set of points (x,y,z) that satisfy the equation;

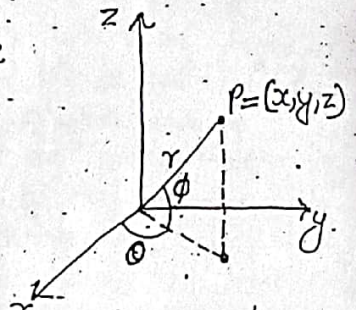
$$x^2+y^2+z^2=r^2.$$

The spherical surface can be represented in parametric form by using latitude and longitudes angles as;

$$x = r\cos\phi\cos\theta, \quad \frac{-\pi}{2} \le \phi \le \frac{\pi}{2}$$

$$y = r\cos\phi\sin\theta, \quad -\pi \le \phi \le \pi$$

$$z = r\sin\phi.$$



The parameter representation in above equation provides a symmetric range for the angular parameter θ and ϕ.

Ellipsoid: Ellipsoid surface is an extension of a spherical surface where the radius in three mutually perpendicular directions can have different values. The cartesian representation for points over the surface of an ellipsoid centred on the origin is;

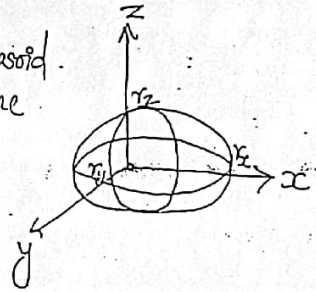
$$\left(\frac{x}{r_x}\right)^2+\left(\frac{y}{r_y}\right)^2+\left(\frac{z}{r_z}\right)^2=1.$$

The parametric representation for the ellipsoid in terms of the latitude angle ϕ and the longitude angle θ is;

$$x = r_x\cos\phi\cos\theta, \quad \frac{-\pi}{2} \le \phi \le \frac{\pi}{2}$$

$$y = r_y\cos\phi\sin\theta, \quad -\pi \le \phi \le \pi$$

$$z = r_z\sin\phi.$$



Unit-6
Solid Modeling

⊛. Solids and solid modeling:-

A solid is a state of matter characterized by particles arranged such that their shape and volume are relatively stable. Solid modeling is the representation of the solid parts of the object on our computer. It is the most advanced method of geometric modeling in three dimensions. It is a complete geometric data representation of an object that enables points in space to be classified relative to the object, if it is inside, outside or on the object. Solid modeling is the foundation of 3D-computer-aided design (CAD) and in general support the creation, exchange, visualization, animation etc. The solid modeling CAD software helps the designer to see the designed object as if it were the real manufactured product. This helps the designer to be sure that the object looks exactly as they wanted it to be.

To make the solid models we have to first make the wire frame model of the object and convert it into 3D view. After that, surfaces are added to the 3D wire model to convert it into 3D solid model. For creating the solid models we need to have special CAD software that can create solid models. One of the most popular CAD software for solid modeling is SolidWorks. Solid modeling software are being used in engineering, medical industry, entertainment industry etc.

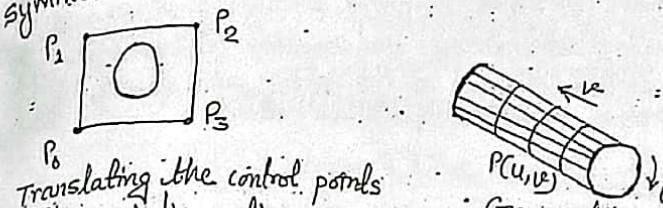
Boundry points → Points where distance to the object and the objects complement is zero.

Interior points → All the other points than boundry points in the object.

Closure → Union of interior points and boundry points.

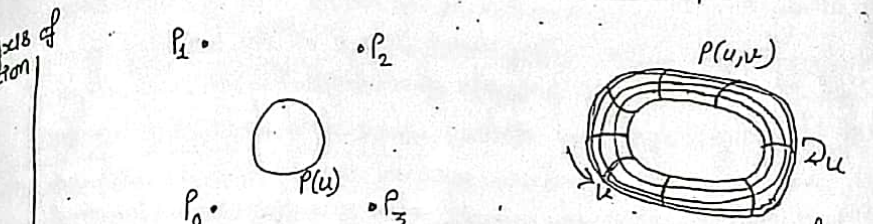
Sweep Representation:-

Sweep representations are used to construct 3D object from 2D shape that have some kind of symmetry. Sweep representations are useful for constructing 3D objects that possess translational, rotational and other symmetries.



Translating the control points of the periodic spline curve. Generating solid with point function P(u,v).

fig. Illustration of a translational sweep.



Rotating the control points of the periodic spline curve. Generating solid about given rotation axis.

fig. Constructing a solid with rotational sweep.

In general we can specify sweep constructions using any path. For rotational sweeps, we can move along a circular path through any angular distance from 0 to 360°. For noncircular paths we can specify the curve function describing the path and the distance of travel along the path. In addition we can vary the shape or size of the cross section along the sweep path. Sweeping is based on the notation of moving a point, curve or a surface along a given path.

⊛. Boundry Representation (B-rep):-

A boundry representation (B-rep) of an object is a geometric and topological description of its boundry. The object boundry is segmented into a finite number of bounded subsets, called faces. A face is represented in a B-rep by its bounding edges and vertices. Thus a B-rep consists of three primitive topological entities: faces (2D entities), edges (1-D entities) and vertices (0-D entities). Geometric information consist of the shape and location in space of each of the primitive topological entities.

Advantages and Disadvantages of B-rep:

The main advantage of B-rep is that it is very appropriate to construct solid models of unusual shapes that are difficult to build using primitives. Another major advantage is that it is relatively simple to convert a B-rep model into a wireframe model.

The disadvantage of the boundry model is that it requires large amounts of storage. The model is defined by its faces, edges and vertices which tend to grow fairly fast for complex models. If B-rep systems do not have a CSG-compatible user interface then it becomes slow and inconvinent to use Euler operations in a design and production environment.

⊛. Spatial-Partitioning Representation:-

It describes the objects as collections of adjoining non-intersecting solids. It creates collections of solids that may or may not be the same type as the original object. It creates collection of solids that are like building blocks and can vary in type, size, position and orientation. Solid objects can be formed with spatial-partioning using cell decomposition, octrees, quadtrees etc.

It is used in computer graphics in ray tracing where it is frequently used to organize the objects in virtual scene. Recursively partitioning space using planes in this way produces a BSP tree, which is one of the most common forms of space partitioning.

⊛. Binary Space Partition Trees (BSP):-

In this representation scheme, we subdivide a scene into two two sections at each step with a plane that can be at any position and orientation. Many 3D modeling and rendering programs utilize a Binary Space Partition tree (BSP tree) to make rendering go faster. It is a generic process of dividing a scene into two until the partitioning satifies one or more requirements. It is a way of grouping data so it can be processed faster.

BSP trees provide more efficient partitioning since we can position and orient the cutting planes to suit the spatial distribution of objects. This reduces the depth of the tree representation for a scene compared to octree and thus reduce the time to search the tree. BSP trees are useful for identifying visible surfaces and for space partioning in ray-tracing algorithms.

⊛. Octree Representation:-

Hierarchical tree structures are called octrees. Octrees are used to represent solid objects in some graphics systems. Medical imaging and other applications that require displays of object commonly use octree representations.

Octree is 3D quadtrees, its three dimensions are recursively subdivided into octants and have quadrants. The octree encoding procedure for a 3D space is an extension scheme for a 2D space called quadtree encoding. Quadtrees are generated by successively dividing a 2D region into quadrants An algorithm for generating a quadtree tests pixel-intensity values and sets up the quadtree nodes accordingly.

Octrees are typically used when the interior of object is important. Octrees are usually applied in raycasting and shadow casting.

Fig. Octree

fig. Simple Binary Space Partition Tree (Simple BSP)

Unit-7

Visible Surface Detection:

Visible surface detection is the process of identifying those parts of a scene that are visible from a chosen viewing position. There are numerous algorithms for efficient identification of visible objects for different types of applications. These various algorithms are referred to as visible-surface detection methods or also referred as hidden-surface elimination methods.

When a picture contains non-transparent objects, the surfaces behind the objects can not be viewed. To obtain a realistic screen image, the hidden surfaces need to be removed. This process of identification and removal of these surfaces is known as hidden-surface problem. These processes helps the computer to decrease processing time for rendering image, eliminating extra information of image.

The hidden surface problems can be solved by two methods:- Object-Space method and Image-space method. In physical coordinate system, object-space method is implemented and in case of screen coordinate system, image-space method is implemented.

Coherence for visibility: (Less imp concept only) Coherence denotes similarities between items or entities. It describes the extent to which these items or entities are locally constant. We may use knowledge about similarities between areas within the scene to make large reductions in the number of calculations. Such an approach is known as 'coherence' and it occurs in many forms.

1) **Object coherence** → If two objects are completely seperate, then we may apply global tests when comparing them rather than testing each vertex of an object.

2) **Face coherence** → Even when the faces are general surfaces and not plane, it is usual for properties of the face to vary smoothly across it.

iii) edge coherence → In edge coherence object changes visibility only when it crosses behind a visible edge or intersects a visible face.

iv) Frame coherence → Pictures of the same scene at two successive times will usually have small changes with much of the scene unchanged. This can be used to reduce the amount of data that has to be calculated each time.

(a) Object-space Methods:-

An object space method compares objects and parts of objects to each other within the scene definition to determine which surfaces, as a whole, we should label as visible. This method can be used effectively to locate visible surfaces in some cases. Line-display algorithms generally use object-space methods to identify visible lines in wire-frame displays. This method is easy to implement but slow than other methods. This method is implemented in physical coordinate system.

Basic Procedure:

For each object in the scene do
1. Begin
   - Determine those parts of the object whose view is unblocked by other parts of it or any other object with respect to the viewing specification.
   - Draw those parts in the object color.
2. End.

Characteristics:

i) If there are n objects in the scene, complexity = $O(n^2)$.
ii) Calculations are performed at the resolution in which the objects are defined.
iii) Display is more accurate but computationally more expensive compared to image-space method.
iv) It is suitable for scene with small number of objects with simple relationship with each other.
v) Process is unrelated to display resolution or the individual pixel in the image.

(b) Image-space method:-

In this method visibility is decided point at each pixel position on the projection plane. This method can be easily implemented to visible-line detection. It is implemented in screen coordinate system. Most of the hidden line/surface algorithms use image-space methods.

Basic Procedure

For each pixel in the image do
1. Begin
   - Determine the object closest to that pixel.
   - Draw the pixel in the object colour.
2. End.

Characteristics:

i) If there are p pixels in the image, complexity depends on n and p $O(np)$.
ii) For each pixel, determine all n objects to determine the one closest to the viewer.
iii) Accuracy of the calculation is bounded by the display resolution.
iv) A change of display resolution requires re-calculation.

(a) Back Face Detection:- (Plane Equation method)

Object space method

In this method objects and parts of objects are compared to find out the visible surfaces. The idea is to check if the object will be facing away from the viewer or not. If it does so, discard it from current frame and move onto the next one. Each surface has a normal vector. If this normal points in the direction of center of projection then it is front face and can be seen by viewer. If this normal is pointing away from the direction of center of projection then it is back face and cannot be seen by viewer.

A point (x, y, z) is inside a polygon surface if;

$$Ax + By + Cz + D < 0.$$

We can simplify this test by considering normal vector N to a polygon surface which has cartesian components (A,B,C). If V is the vector in viewing direction from the eye position then this polygon is a back face if $V.N > 0$.

In the equation $Ax+By+Cz+D=0$, if A,B,C remains constant, then varying value of D results in a whole family of parallel planes. One of which $(D=0)$ contains the origin of the coordinate system and,

- If $D>0$, plane is away from the observer.
- If $D<0$, plane is towards the observer.

If the object is centered at origin, the all surface that are viewable will have negative D and un-viewable surface have positive D.

Limitations

→ This method works fine for convex polyhedra, but not necessarily for concave polyhedra.

→ This method can only be used on solid objects modeled as a polygon mesh.

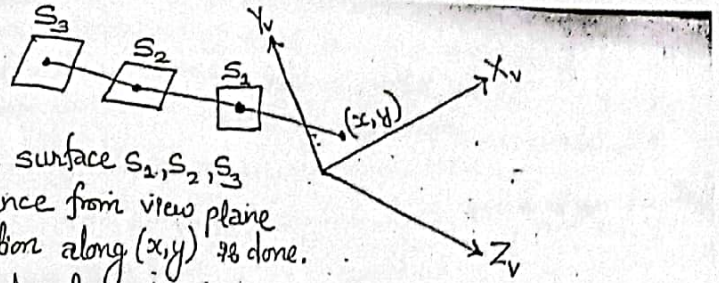
←→ With this method we can not identify visible surface in case of overlapping objects.

Most Popular Methods for Image-Space Method:-

1> Depth Buffer (Z-buffer): (Imp)

In this method two buffers are used named as frame buffer and depth buffer. Depth buffer is used to store depth values for (x,y) position, as surfaces are processed. $(0 \leq \text{depth} \leq 1)$. The frame buffer is used to store the intensity value of color value at each position (x,y).

The z-coordinates are usually normalized to the range $[0,1]$. The 0 value for z-coordinate indicates back clipping plane and 1 value for z-coordinates indicates front clipping plane.



In figure, three surface S_1, S_2, S_3 at varying distance from view plane $X_v Y_v$ the projection along (x,y) is done. Surface S_1 is closest to the view-plane so surface intensity value of S_1 at (x,y) is saved.

Initially all the positions in depth buffer is set to 0 and refresh buffer is initialized to background color. Each surface listed in polygon table is processed one scan line at a time, calculating the depth (z-value) for each position (x,y). The calculated depth is compared to the value previously stored in depth buffer at that position. If calculated depth is greater than stored depth value in depth buffer, new depth value is stored and the surface intensity at that position is determined and placed in refresh buffer.

After all surfaces are processed, the depth value of surface position (x,y) is calculated by plane equation surface.

$$Z = \frac{-Ax-By-D}{C}$$

Let depth Z' at position $(x+1,y)$

$$Z' = \frac{-A(x+1)-By-D}{C}$$

$$\Rightarrow Z' = Z - A/C$$

$-A/C$ is constant for each surface so succeeding depth value across a scan line are obtained from preceding values by simple calculation.

Algorithm:

1. Start
2. Set the buffer values
   - Depthbuffer (x,y) = 0
   - framebuffer (x,y) = background color.
3. Process each polygon (One at a time).
   For each projected (x,y) pixel position of a polygon, calculate depth z.
   If z > depth buffer (x,y).
   - Compute surface color.
   - set depthbuffer (x,y) = z.
   - frame buffer (x,y) = surfacecolor (x,y).
4. Stop.

Implementation steps for Depth buffer

1. Initially each pixel of the z-buffer is set to the maximum depth value.
2. The image buffer is set to the background color.
3. Surfaces are rendered (provided) one at a time.
4. For the first surface, the depth value of each pixel is calculated.
5. If this depth value is smaller than the corresponding depth value in the z-buffer then both the depth value in the z-buffer and the color value in the image buffer are replaced by the depth value and the color value of this surface calculated at the pixel position.
6. Repeat step 4 and 5 for the remaining surfaces.
7. After all the surfaces have been processed, each pixel of the image buffer represents the color of a visible surface at that pixel.

Advantages:

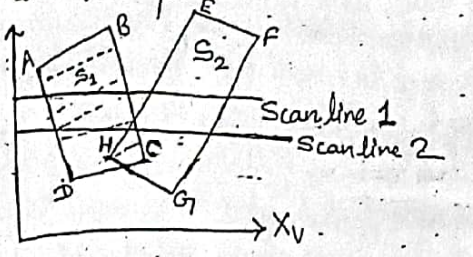
- → Simple and does not require additional data structures.
- → No object-comparision required.
- → Processes one object at a time.
- → Can be applied to non-polygonal objects.

Disadvantages:

- → This method requires additional buffer.
- → It requires large memory.
- → It is time consuming process.

Scan-line Method:

In this method, each scan line is processed. All polygon surfaces intersecting that line are examined to determine which are visible. Across each scan line, depth calculations are made for each overlapping surface to determine which is nearest to the view plane.



active edge → scan line जो काटेगा side E तक

ON → indicate a scan line is inside of surface

OFF → indicate a scan line is outside of surface

Note:- Select surface according to chosen sides

Working Procedure:-

The active list for scan line one contains information from the edge table for edges AD, BC, EH and FG.

For scan line 1, between edges AD and BC, flag (S_1) = ON and flag (S_2) = Off. Since no overlapping case so, no depth calculations are required and intensity information for surface S_1 is entered from the polygon table into the refresh buffer.

Similarly, between edge EH and FG flag (S_1) = OFF and flag (S_2) = ON.

For scan line 2 : the active edges AD, EH, BC and FG.

Along the line 2, from edge AD to edge EH, flag (S_1) = ON and flag (S_2) = OFF

But between edges EH and BC, flag (S_1) = ON and flag (S_2) = ON, since this is the case of overlapping. Therefore depth calculation must be made using the plane coefficient for the two surfaces.

For this example the depth surface S_1 is assumed to be less than that of S_2. So intensity of surface S_1 ~~is assumed to be less than~~ are loaded into the refresh buffer until boundry BC is encountered. Then the flag for the surface S_1 goes off. The intensity for surface S_2 is stored until edge FG is passed.

Algorithm:

1. Start
2. Established data structure.
   a. Edge table with line endpoints.
   b. Polygon table with plane coefficients, color and pointer to the edge table.
   c. Active edge list sorted in increasing order of x.
   d. A flag for each surface.
3. Repeat for each scan line.
   a. Update AEL (active edge list) for each scan line y.
   b. When flag is ON, enter intensity of that surface to refresh buffer.
   c. When two or more flags are ON, calculate depth and store the intensity of the surface nearest to view plane.
4. Stop.

Characteristics:-

i) Processing of the scan line start from the top to the bottom of the display.
ii) It requires an edge table, polygon table, active edge list and flag.
iii) It deals with multiple polygons.
iv) This method calculates the z-value for only the overlapping surface which is tested by scan line.
v) Across each scan line, depth calculations are made for each overlapping surface to determine which surface is nearest to the view plane.

Advantages:-

→ Any number of overlapping polygon surfaces can be processed with this method.
→ Deals with transparent, translucent, and opaque surfaces.
→ Can be applied to non-polygonal objects.

Disadvantages:-

→ Depth calculations are performed only when there are overlapping polygons.
→ Additional memory buffer is required.
→ Complex.

Depth Sorting Method (Painter's Algorithm):

This method uses both object space and image space method. The depth sorting method performs following two basic functions:

i) First, the surfaces are sorted in order of decreasing depth.
ii) Second, the surfaces are scan-converted in order, starting with the surface of greatest depth.

The intensity values for farthest surface are entered into the refresh buffer. The farthest polygon is displayed first, then the second farthest polygon and so on. After all surfaces have been processed, the refresh buffer stores the final intensity values for all visible surfaces.

When there are only a few objects in the scene then this method can be very fast. However, as the number of objects increases, the sorting process can become very complex and time consuming.

Advantages:

→ The sorting computation is very fast, so quick calculation of the image display is possible.
→ The finished image data remains object based and can be edited in terms of line weight, exploded views etc.
→ It will also print at the highest resolution available on devices such as postScript laser printers.

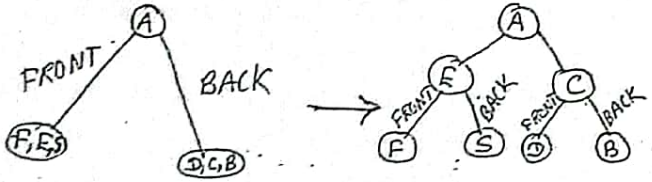
Disadvantages:

→ Different polygons may have same depth.
→ The nearest polygon could also be farthest.
→ We can not use simple depth-sorting to remove the hidden surfaces in the images.

⑤ Binary Space Partitioning (BPS) Tree Method:-

It is an enhancement of depth sorting method/algorithm that divides entire 3D scene into two parts by selecting any polygon surface as a dividing partitioning plane. This method is best suited when a 3D scene is static but, view port changes frequently. This method first displays polygon surfaces that are far from viewport and enters their intensity into frame buffer. It can be divided into two steps;

1) Construction of BSP tree → A space that contains 3D scene is recursively divided into two parts i.e FRONT and BACK with respect to viewport, by selecting any surface as a dividing plane. The recursive sub-division process is repeated until each half space contains exactly one object or zero. This recursive division process is represented as binary tree with initial partitioning plane as a root node. FRONT object is represented as a left node of a tree and BACK object is represented as a right node of a tree.



2) Display of BSP tree → If view port is infront of root node, then BSP tree is displayed in reverse inorder traversal i.e, right, root and left.

B, C, D, A, S, E, F

If view port is in back of root node, then BSP tree is displayed in inorder traversal. i.e, left, root and right.

F, E, S, A, D, C, B

⑥ A-Buffer Method:-

It is the extension of Z-Buffer method that deals with all kinds of surfaces. It is called accumulation method because it can accumulate color information of all background surfaces that are visible through foreground transparent surface.

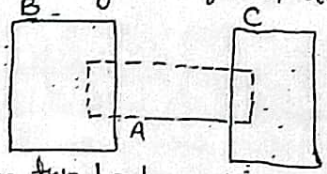


fig: Viewing two background surfaces B & C through a transparent surface A.

It extends frame buffer of foreground surface hence it can point all the background surfaces that are visible. A buffer has two fields;

1) Depth Field → It stores +ve and -ve depth value of a particular surface. If it is +ve then that surface is opaque surface otherwise it is translucent surface.

Depth Field	Intensity field

2) Intensity Field → It stores surface intensity value or pointer to background surfaces.
   → If depth is +ve then intensity field contains color value of that surface.
   → If depth value is -ve then intensity field contains pointer to all visible background surfaces.
   → Intensity field can contain
   - RGB information
   - Surface Opacity
   - Spatial orientation
   - Pointer value to other surfaces.

Pros and Cons of A-Buffer

→ It deals with all types of surfaces.
→ Requires more computation than Z-buffer method.
→ Depth computation is always performed if single surface is visible surface.

⊛ Ray Tracing Method:-

In this method rays are casted from each pixel to the scene and visible surfaces are identified based on nearest intersection point. The number of casted light rays depends on the no. of pixels in the viewplane i.e. resolution. This method is capable of producing high degree of visual realism which makes suits best to applications where image can be rendered slowly ahead of time such as still images, television, visual effects etc. It is capable of wide variety of optical effects such as reflection and refraction, scattering and dispersion phenomena.

Advantages:

→ Conceptually simple.
→ Automatic hidden surface removal
→ Ray-object insertion is fundamental in computer graphics like visibility testing.

⊛ Octree Method:-

An octree is a tree data structure in which each internal node has exactly eight children. Octrees are used to partition 3D space recursively subdividing it into eight octants. Octrees are often used in 3D graphics and 3D game engines.

When an octree representation is used for viewing volume, hidden surface elimination is accomplished by projecting octree nodes into viewing surface in a front to back order.

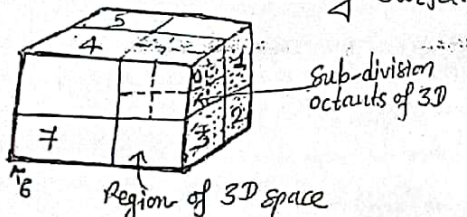


fig Front face with octants 0,1,2,3. (Visible to viewer).

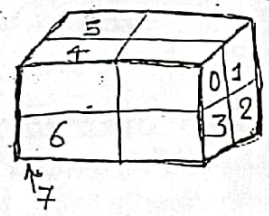
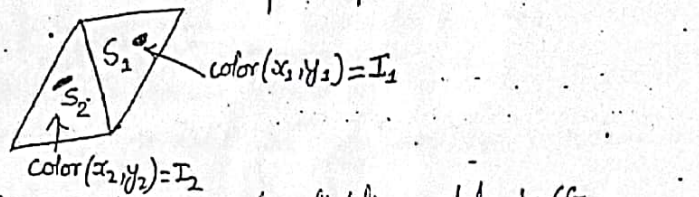


fig. Back face with octants 4,5,6,7 (Not visible to viewer)

Unit-8

Illumination Models and Surface Rendering Techniques:

Illumination models are mathematical model to determine color calculation or intensity calculation of a single pixel within a particular surface. It is sometimes also refered as shading model. The colour seen on particular point of the surface depends on various optical parameters. as below:-



1) Light source Type → There are two lighting models in CG.

a) Point light source → Light rays radially diverge from a light source and there is a point of a origin of light rays. Light intensity appears equal in all directions
e.g. Bulb, Sun.

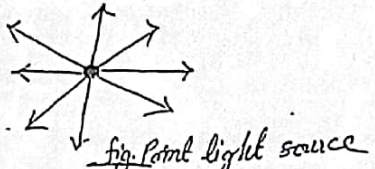
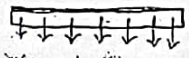


fig. Point light source

b) Distributed light source → There is no point origin and light rays are parallel to each other. Light rays are focused to particular direction. Eg. Lazer light, Torch, etc



2) Surface Characteristics → It determines type of reflection from a particular surface. Surface can be rough, shiny, transparent and can produce specular reflection or diffuse reflection. Illumination model are necessary in CG for producing realistic displays.

Types of Illumination Models:

1) Ambient Light: Objects that are not in direct control with light source can still be visible if nearby objects are illuminated this is called ambient light. It is constant in all directions irrespective of viewing direction.

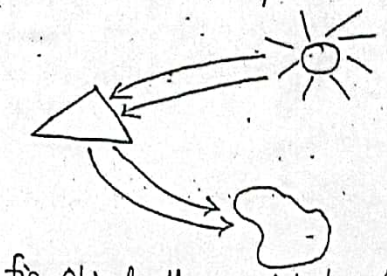


fig. Object illuminated due to ambient light

If a surface is exposed only to ambient light, then the intensity of the diffuse reflection at any point on surface is; $I = K_a I_a$.

where, I_a is the intensity of ambient light
& K_a is the ambient coefficient reflection.

2) Diffuse reflection: It is a reflection due to rough regular surfaces. Reflection of light is equal in all directions. It is the background light reflected from walls, floor and ceilings.

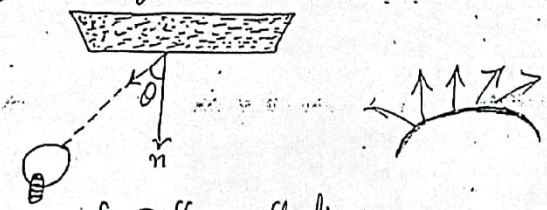


fig. Diffuse reflection

This effect of light reflection for purely dull surfaces can be computed according to Lambertis cosine law by illumination equation;

$$I = I_l \cdot K_d \cdot \cos\theta$$

where, I_l is intensity of light hitting surface, $0 \le K_d \le 1$ is the reflection coefficient of surface and θ angle between normal vector n to surface.

3) Specular Reflection:-

Specular light is the white highlight reflection seen on smooth, shiny objects. Specular reflection is a reflection due to shiny surfaces. This phenomena occurs due to total internal reflection of a incident light. In this phenomena maximum intensity appears in particular direction.

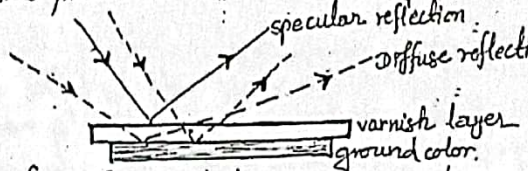


fig. Differences between diffuse and specular reflections.

It is calculated using Phong model. The Phong specular reflection model is described by the relation,

$$I = I_l \cdot w(\theta) \cdot \cos(\alpha)^n$$

where, I_l is intensity of light. The value $0 \le w(\theta) \le 1$ is the fraction of light which is directly reflected at the shiny surface.

4) Intensity Attenuation:- The rate of decrease in intensity w.r.t. distance between light source and objects is called intensity attenuation. If point light source is used the intensity attenuation is given by $\frac{1}{d^2}$. But if distributed light is used then intensity attenuation factor is given by a function $f(d) = \frac{1}{a_0 + a_1 d + a_2 d^2}$

where a_0, a_1 & a_2 are surface parameter
and d is distance betn object and distributed light source.

5) Color Consideration:- Most of graphics displays of realistic scenes are in colour. But the illumination model discussed so far considers only ~~mono~~ monochromatic ~~from each other~~. lighting effects. To incorporate color, we need to write the intensity equation as a function of the colour properties of the light sources and object surfaces.

Transparency:-
Transparent surfaces reflect light but objects behind them can also be seen. A typical transparent object is coloured glass plane. Transparency means that only a fraction of light can pass behind through the transparent surface.

Shadows:-
Shadow can help to create realism (means like as real). Shadows contribute a lot to the visual effect of the scene. Through shadows humans distinguish more clearly movement and depth of objects. There are number of techniques that can be used to create shadows for the objects.

Polygon (surface) Rendering Method / Surface Shading:-
Polygon rendering is the process of calculating intensity and color considerations for a polygon surface.

Scene description + Illumination Model + Rendering Technique = Image.

There are two ways of polygon rendering:
- Rendering each polygon surface with single intensity
- Calculate intensity at each point of the surface using interpolation scheme.

There are three approaches for surface rendering as below:

1) Constant Intensity shading / Flat shading:-
In this method illumination model is applied by selecting arbitary pixel inside the surface and calculated intensity is applied to all other pixels inside the surface. It requires less computations but can not produce realistic images. It is the simplest model for shading a polygon.

Algorithm:-
→ Calculate surface normal vector for each surface.
→ Apply illumination model to particular interior pixel to determine intensity value.
→ Assign calculated intensity value to all other pixels on the surface.

Gouraud shading / Intensity Interpolation Method:-
In this method, firstly the average surface normal vector at each vertex is determined as;

$$\vec{N_v} = \frac{\vec{N_1} + \vec{N_2} + \ldots + \vec{N_n}}{|\vec{N_1} + \vec{N_2} + \ldots + \vec{N_n}|}$$

where, $\vec{N_v}$ is the average surface normal vertex at vertex V, and $\vec{N_1}, \vec{N_2}, \ldots, \vec{N_n}$ are surface normal vectors on surfaces $S_1, S_2, \ldots, S_n$. Here, vertex V is shared by all n surfaces.

...n illumination model are applied to each vertex to determine ...ensity value at that vertex.
These calculated intensities are interpolated to determine intensity value of all other pixels.
It provides more realistic graphics than flat shading but suffered by Mach-Band Effect (i.e, Appearance of dark and bright spots at the corner of the objects surface).

...gorithm:-
Calculate average surface normal vector at each vertex.
Apply illumination model of vertices to calculate intensity value.
Apply interpolation to vertex intensities by using vertex coordinates and intensity at that vertex to determine intensities for all other pixel.

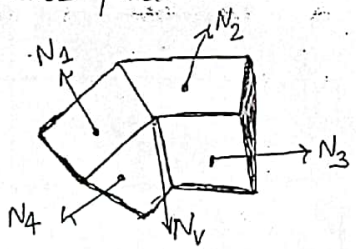


fig. Gouraud Shading

3) Phong Shading / Normal Vector Interpolation:-

An improvement of Gouraud shading suggested by Phong Bui Toung that interpolates average surface normal vector to calculate color of the surface. It is the most efficient shading method but requires large number of computation.

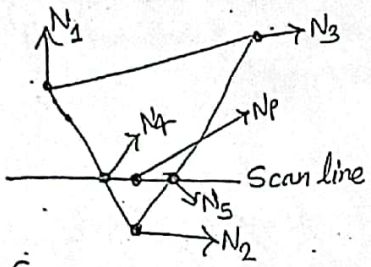
Algorithm:-

→ Calculate average surface normal vector at each vertex.

→ Apply interpolation to determine average normal vector for all other points by using average normal vector at vertices.

$$N = \frac{y - y_2}{y_1 - y_2} N_1 + \frac{y_1 - y}{y_1 - y_2} N_2$$

This is general form for interpolating normal vectors.



$$N_4 = \frac{y_4 - y_2}{y_1 - y_2} N_1 + \frac{y_1 - y_4}{y_1 - y_2} N_2$$

$$N_5 = \frac{y_5 - y_2}{y_3 - y_2} N_3 + \frac{y_3 - y_5}{y_3 - y_2} N_2$$

$$N_p = \frac{y_p - y_5}{y_4 - y_5} N_4 + \frac{y_4 - y_p}{y_4 - y_5} N_5$$

fig. Phong shading

Advantages:

→ It displays more realistic highlights on a surface.
→ It reduces the mach band effect.
→ It gives more accurate result.

Disadvantages:

→ It requires more calculations
→ It greatly increases the cost of shading steeply.

④ Differences between Gouraud Shading & Phong Shading

S.No	Gouraud Shading	Phong Shading
1.	Gouraud shading is named after Henri Gouraud.	Phong Shading model is named after Bui Tuong Phong.
2.	Computes illumination at border vertices and interpolates	Illumination at every point of polygon surface.
3.	Interpolates colors along edges and scan line.	Interpolates normal instead of colors.
4.	Not so expensive.	More expensive than Gouraud shading.
5.	Lighting equation is used at each vertex.	Lighting equation is used at each pixel.
6.	Requires moderate processing and time.	Required complex processing and it is slower but produces good quality.

Unit-9

Introduction to Virtual Reality:

Virtual reality is an artificial environment that is created with software and presented to the user in such a way that the user suspends belief and accepts it as a real environment. Virtual Reality (VR) is the use of computer technology to create a simulated environment.

In this type of view: instead of viewing a screen in front of them, users are able to interact with 3D worlds. Users are able to simulate as many senses as possible, such as vision, hearing, touch etc. Unlike traditional user interfaces, VR places the user inside an experience.

Virtual Reality can be divided into:

i) The simulation of real environment for training and education.
ii) The development of an imagined environment for a game or interactive story.

VR has five main components as:

i) Dimensionality.
ii) Motion or animation
iii) Interaction.
iv) Viewpoint
v) Immersion through enhanced multisensory experiences.

Advantages:

→ Virtual Reality creates a realistic world.
→ It helps user to experiment with an artificial environment.
→ VR is more personal than electronic mail or instant messaging.
→ VR helps to get the knowledge of different things more easily and with comfort.

Disadvantages:

→ It consists of complex technology.
→ Equipments used in virtual reality are very expensive.
→ In VR environment we can not move by our own, like in the real world.

Application Areas:-

i) In Education
ii) In Health care
iii) In environment.
iv) In engineering
v) In Scientific Visualizations
vi) In Media
vii) In: Telecommunications.
viii) In Construction.

Immersive meaning → Process of generating 3D image which appears to surround the user.

Types of Virtual Reality:-

i) Non-Immersive → Non-Immersive simulations are the least immersive implementation of virtual reality technology. In this only the subsets of the user's senses are simulated allowing for peripheral awarness of the reality outside the virtual reality simulation. Users enter into these three-dimensional virtual environments through a portal or window utilizing standard high resolution monitors typically found on conventional desktop stations.

ii) Semi-Immersive → Semi-Immersive simulations provide a more immersive experience, in which the user is partly but not fully immersed in a virtual environment. Semi-immersive simulations are powered by high-performance graphical computing systems.

iii) Fully-Immersive → Fully-Immersive simulations provide a most immersive implementation of virtual reality technology. In fully-immersive simulation, hardware such as head-mounted displays and motion detecting devices are used to simulate all of the user's senses. These type of VR are able to provide very ~~vet~~ realistic user experiences by ~~deter~~ delivering a wide field of view and high resolutions.

⊛ 3D positional tracking:-

Positional tracking is a technology that allows a device to estimate its position relative to the environment around it. It uses a combination of hardware and software to achieve the detection of its absolute position. It is an essential technology for virtual reality (VR), making it possible to track movement with six degrees of freedom (6DOF).

Positional tracking VR technology brings various benefits to the VR experience. It can change the viewpoint of the user to reflect different actions like jumping, ducking or leaning forward. It increases the connection between the physical and virtual world.

⊛ Key Components in a Virtual Reality System:-

1) PC / Console / Smart phone → Computers are used to process inputs and outputs sequentially. To power the content creation and production significant computing power is required for making PC/console/smart phones important part of VR systems. They act as the engine to power the content being produced.

2) Head-Mounted Display → A head-mounted display is a type of device that contains a display mounted in front of user's eyes. This display usually covers the users full field of view and displays virtual reality content. Smart phone displays, including the Google Cardboard and Samsung Gear VR. Head-mounted displays are often also accompanied with a headset to provide for audio simulation.

3) Input Devices → Input devices provide users with a more natural way to navigate and interact within a virtual reality environment. Some of the most common forms of virtual reality devices are :- Joysticks, Tracking Balls, Data Gloves, Motion Platforms etc.

⊛ Visual computation in virtual Reality:-

Visual computation is a computation that lets us to interact and control by manipulating visual images either as direct work objects or as objects representing other objects that are not necessarily visual themselves. The visual images can be photographs, 3-D scenes, block diagrams or simple icons. Visual computation is generally used to describe following two things:-

1) Computer environment in which a visual ~~paradiagram~~ paradigm rather than text paradigm used.
2) Applications that deal with large or numerous image files, such as video sequences and 3-D scenes.

⊛ Augmented Reality:-

Augmented Reality is the result of using technology to superimpose informations like sound, images and text on the world we see. Augmented Reality (AR) is the technology that expands our physical world, adding layers of digital information on it. There are four types of augmented reality today: markless AR, marker-based AR, projection-based AR and superimposition-based AR.

⊛ Virtual Reality (VR) vs. Augmented Reality (AR):

Virtual Reality	Augmented Reality
→ Virtual Reality creates an entire virtual world.	→ Augmented reality is a mix of real world and the virtual world.
→ In this case, it hard to differentiate between what is real and what is not real.	→ It lets people interact with both worlds and distinguish clearly between both.
→ This is generally achieved by wearing a helmet or goggles having VR technology.	→ This is achieved by holding smart phone infront of us.

Unit-10

- Introduction to OpenGL :

OpenGL is a software interface that allows a programmer to communicate with graphics hardware. OpenGL is the short form for "Open Graphics Library". It is an application programming interface (API) designed for rendering 2D and 3D graphics. It provides a common set of commands that can be used to manage graphics in different applications and on multiple platforms. Examples of OpenGL commands include drawing polygons, assigning colors to shapes, zooming in and out, rotating objects etc.

OpenGL Utility Toolkit (GLUT) has been created to aid in the development of more complicated 3D objects such as a sphere, a torus and even a teapot. Like all software libraries, OpenGL consists of a series of function calls that can be invoked from our own programs.

⊛. Why OpenGL ?

→ The first vendor-independent API for development of graphics applications.
→ There is no need to license it to use it.
→ It was designed to use the graphics card where possible to improve performance.
→ Originally based on a state machine procedural model; thus it can be used with a wide variety of programming languages.

⊛. Callback Functions :-

A callback function is a function which the library (GLUT) calls when it needs to know how to process something.
Eg. When GLUT gets a key down event it uses the glutKeyboardFunc callback routine to find out what to do with a key press.
The following are some valid callback functions :-

Callback	Description
GLU_TESS_BEGIN	The GLU_TESS_BEGIN callback is invoked like glBegin to indicate the start of a (triangle) primitive. The function takes a single argument of type GLenum.
GLU_TESS_BEGIN_DATA	GLU_TESS_BEGIN_DATA is same as GLU_TESS_BEGIN callback except that it takes an additional pointer argument.
GLU_TESS_EDGE_FLAG	The GLU_TESS_EDGE_FLAG callback is similar to glEdgeFlag. This function takes a single boolean flag that indicates which edges lay on the polygon boundry
GLU_TESS_EDGE_FLAG_DATA	The GLU_TESS_EDGE_FLAG_DATA callback is the same as GLU_TESS_EDGE_FLAG callback except it takes an additional pointer argument.
GLU_TESS_END	The GLU_TESS_END callback indicates the end of a primitive, and it takes no arguments.

⊛. Color Commands :-

OpenGL has two color models : the RGBA mode and color index mode. In RGBA mode, a color is specified by three intensities (for the Red, Green and Blue components of the color) and optionally a fourth value, Alpha, which controls transparency. The function, glColor4f (red, green, blue, alpha)

maps available red, green, and blue intensities onto (0.0, 1.0) where 0.0 means that the color component is absent (0.0, 0.0, 0.0) which indicates black and 1.0 is a saturated color (1.0, 1.0, 1.0) which indicates white.

The number of bits used to represent each color depends upon the graphics card. Current graphics cards have several Mbytes of memory and use 24 or 32 bits for color. The term bit plane refers to an image of single-bit values. Thus a system of 24 bits of color has 24 bit planes.

⊛ Drawing pixels, lines, polygons using OpenGL:-

Creating Lines in OpenGL → In OpenGL, the term line refers to a line segment. There are easy ways to specify a connected series of line segments, or even a closed, connected series of segments. In all cases the lines constituting the connected series are specified in terms of the vertices at their endpoints.

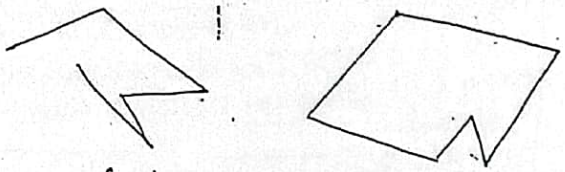


fig: two connected series of line segments.

With OpenGL, we can specify lines with different widths and lines that are stippled in various ways - dotted, dashed and so on. Drawing a line with OpenGL is apparently quite simple using the following code.

```
glBegin(GL_LINES);
    glVertex2i(x0,y0);
    glVertex2i(x1,y1);
glEnd();
```

Creating Polygons in OpenGL → Polygons are typically drawn by filling in all the pixels enclosed within the boundry, but we can also draw them as outlined polygons or simply points at the vertices. A filled polygon might be solidly filled or stippled with a certain pattern.

A polygon has two sides front and back and might be rendered differently depending on which side is facing the viewer. This allows us to have cutaway views of solid objects in which there is an obvious distinction between the parts that are inside and those that are outside. By default, both front and back faces are drawn in the same way. To change this, or to draw only outlines or vertices, use

```
glPolygonMode();
```

We can draw polygon by using following code segment;

```
glBegin(GL_POLYGON); //Draw a Quad
glVertex3f(-0.5f, 1.0f, 0.0f); //Top.
glVertex3f(-1.0f, 0.0f, 0.0f); //Left.
glVertex3f(-0.5f, 1.0f, 0.0f); // Bottom Left.
glVertex3f(0.05f, 1.0f, 0.0f); // Top right
glVertex3f(1.0f, 0.0f, 0.0f); // Right
glVertex3f(0.5f, 1.0f, 0.0f); // Bottom Right
glEnd();
```

⊛ Basic Lighting:-

Lighting in the real world is extremely complicated and depends on way to many factors. Lighting in OpenGL is based on approximations of reality using simplified models that are much easier to process and looks relatively similar. These lighting models are based on the physics of light as we understand it. One of those models is called the Phong lighting model. The major building blocks of the Phong model consists of following three components;

i) **Ambient lighting** → Even when it is dark there is usually still some light somewhere in the world (the moon, a distant light) so objects are never almost never completely dark. To simulate this we use an ambient lighting constant that always gives the object some color. Ambient illumination is light that's been scattered so much by the environment that its direction is impossible to determine. It seems to come from all directions.

ii) **Diffuse Lighting** → This is the most visually significant component of the lighting model. The more a part of an object faces the light source, the brighter it becomes. The diffuse component is the light that comes from one direction, so it's brighter if it comes squarely down on a surface than if it barely glances off the surface. Once it hits a surface it is scattered equally in all directions, so it appears equally bright.

iii) **Specular lighting** → It simulates the bright spot of a light that appears on shiny objects. Specular highlights are often more inclined to the color of the light than the color of the object. Specular light comes from a particular direction, and it tends to bounce off the surface in a preferred direction. Shiny metal or plastic has a high specular component.

∴ total error : $Q = \sum_{i=1}^{n} q_i^2 = \sum (a+bx_i-y_i)^2$

To be minimum, $\frac{\partial Q}{\partial a} = 0 \Rightarrow 2\sum(a+bx_i-y_i) = 0$

$\frac{\partial Q}{\partial b} = 0 \Rightarrow 2\sum (a+bx_i-y_i)\cdot x_i = 0$

By solving the two,

$$b = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum x_i^2 - (\sum x_i)^2}$$

$$a = \frac{\sum y_i}{n} - b\frac{\sum x_i}{n}$$

This is the required formula.

Some examples

Q.1 Fit a straight line : $y = ax+b$

x	1	2	3	4	5
y	3	4	5	6	8

& find y at $x = 2.5$

Soln :- Let $y = a+bx$

x	y	x^2	xy
1	3	1	3
2	4	4	8
3	5	9	15
4	6	16	24
5	8	25	40
$\sum x = 15$	$\sum y = 26$	$\sum x^2 = 55$	$\sum xy = 90$

Using formula, where $n = 5$

$$b = \frac{n\sum xy - \sum x \sum y}{n\sum x^2 - (\sum x)^2} = \frac{5\times 90 - 15\times 26}{5\times 55 - (15)^2}$$

$$= 1.2$$

$$a = \frac{\sum y}{n} - b\frac{\sum x}{n} = \frac{26}{5} - 1.2\times\frac{15}{5}$$

$$= 1.6$$