1. Introduction to Computer graphics [8-10 M]
2. output primitives [20-24 M]
3. 2-Dimensional geometric transformations [15-18 M]
4. 2-Dimensional viewing and clipping [22-24 M]
5. 3-d Object representation, geometric transformation and 3-d viewing [20-24 M]
6. Visible surface detection [10 M]
7. Illumination models and surface rendering [8-10M]

Scanned by CamScanner

1: Introduction to Computer Graphics

Q1.) Define Computer Graphics. Explain the application areas of computer graphics.

A. 1.) Computer is an information processing machine. Computer graphics is the pictorial representation of information using a computer program.

2.) Computer graphics is one of the most effective & most commonly used to communicate processing information to the user. It displays the info. in the form of graphical objects such as pictures, charts and diagrams, images, audio-video, animated frames insted of plain text.

Application areas of C.G. -

1.) User interfaces - Graphics is user friendly. Graphical interface provides an attractive and easy interaction between users and computers.

2.) Plotting graphics and charts - In industry, business, government and educational organizations C.G. is most commonly used to create 2-D and 3-D graphs of mathematical, physical and economical functions in the form of histograms and charts.

3.) Computer added drafting & design (CAD) - CAD is used to design automobile bodies, structures of buildings, airplane, ships and VLSI chips.

4.) Simulation and animation - C.G. is used in the production of animated movies and cartoon films.

Scanned by CamScanner

5.) Education & Training - computer generated models of physical, financial and economical systems are used as education tools.

6.) Cartography - It is used to represent geographic maps, weather maps, population density maps, etc.

7.) Computer graphics is used in medical imaging to educate new doctors on understanding and experimenting surgery.

8.) The other areas like multimedia, animation, virtual reality, molecular biology where C.G. is used as an effective tool.

Q2.] Define the following terminology.

1. Pixel - In C.G. pictures are represented as collection of discrete picture element called pixels. A pixel is the smallest addressible screen element.

2. Resolution - It refers to the no. of dots on the screen. It is expressed as a pair of nos. that gives the no. of dots on a horizontal line and the no. of such vertical lines. The standard resolutions are 640 x 480, 800 x 600, 1024 x 768, 1280 x 1024.

3. Rasterization - The process of determining the appropriate pixel for representing pictures or object is called rasterization.

Scanned by CamScanner

④ Scan Conversion - The process of representing continuous pictures as a graphical object is called scan conversion.

Q3] Compare Raster and Random scan system.

Raster	Random
1.) The electron beam is deflected in a pre-determined manner, starting from top left scanning one scan line at a time and after reaching to the end of last scan line, e^- beam gets deflected back to the top left of the screen. This activity is repeated specified no. of times per second depending on the refresh rate.	The e^- beam is deflected in a random manner so as to only scan those straight line paths which constitutes a picture and the activity is repeated specified no. of times/sec depending on the refresh rate.
2.) Raster scan systems are more suitable to geometric area drawing applications such as pictures, images, audio-video device.	Random scan systems are more suitable ~~to~~ for line drawing application to draw basic mathematical shapes like arc, circle, curve, rectangle, etc.

Scanned by CamScanner

3)	Resolut^n of raster scan system is low	It has higher resolution than the raster scan system.
4)	cost is less.	Cost is more.
5)	eg. computer monitor, television.	CRO, pen-plotter

Scanned by CamScanner

2. Output Primitives

Line Generation Algorithms-

Q1.] Characteristics of good line drawing algorithm

i.) Input parameter required to draw line AB is the end point co-ordinates $A(x_1, y_1)$ and $B(x_2, y_2)$.
The eq^n of straight line is $y = mx + c$ where
m = slope of line AB $= \frac{dy}{dx}$ or $\frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}$.

Any line can be categorised into 2 types based on its slope. –

① $|m| \leq 1$ $\left[\text{i.e. } \left| \frac{dy}{dx} \right| \leq 1 \Rightarrow abs(dx) \geq abs(dy) \right]$

② $|m| > 1$ $\left[\text{i.e. } \left| \frac{dy}{dx} \right| > 1 \text{ i.e. } abs(dy) > abs(dx) \right]$

ii.) Line must appear straight.
iii.) Line should be drawn smoothly and rapidly.
iv.) Line should terminate accurately.
v.) Slope of any 2 intermediate pixel position along the line path must be same as slope of entire line.
vi.) Line should have equal density i.e. each and every pixel along the line path should have constant density or thickness or brightness.

Scanned by CamScanner

DDA line generation algorithm:-
(Digital Differential Analyzer)

Step 1:- Accept end pt. coordinates of line AB
i.e. $A(x_1, y_1)$ and $B(x_2, y_2)$
putpixel $(x, y, 15)$;

Step 2:- Calculate
$dx = x_2 - x_1$
$dy = y_2 - y_1$

Step 3:- if $(abs(dx) \geq abs(dy))$

then step = abs(dx); // $|m| \leq 1$

else
step = abs(dy); // $|m| > 1$

Step 4:- Calculate
$xinc = \frac{dx}{step}$;

$yinc = \frac{dy}{step}$;

Step 5:-
$x_{k+1} = x_k + xinc$;
$y_{k+1} = y_k + yinc$;
putpixel $(x_{k+1}, y_{k+1}, 15)$;

Step 6:-
Repeat step 5 till $x_{k+1} != x_2$ and $y_{k+1} != y_2$

Scanned by CamScanner

Q.] Rasterise line AB between end point co-ordinates A(12,20) and B(20,10) using DDA line generation algorithm.

Soln:-

① Accept end point coordinates of line AB i.e.
$A(x_1, y_1) = (12, 20)$ and $B(x_2, y_2) = (20, 10)$
putpixel(12,20,15);

② calculate $dx = x_2 - x_1 = 8$
$dy = y_2 - y_1 = -10$

③ Here abs(dy) > abs(dx)
∴ line AB is having $|m| > 1$
∴ step = abs(dy)
step = 10

④ calculate $x_{inc} = \frac{dx}{step} = \frac{8}{10} = 0.8$

$y_{inc} = \frac{dy}{step} = \frac{-10}{10} = -1$

⑤ calculate $x_{k+1} = x_k + x_{inc} = x_k + 0.8$
$y_{k+1} = y_k + y_{inc} = y_k - 1$
putpixel(x_{k+1}, y_{k+1}, 15);

Scanned by CamScanner

	x_{k+1}	y_{k+1}	Round (x_{k+1})	putpixel $(x_{k+1}, y_{k+1}, 15)$;
A →	12	20	12	putpixel (12, 20, 15);
	12.8	19	13	putpixel (13, 19, 15);
	13.6	18	14	putpixel (14, 18, 15);
	14.4	17	14	putpixel (14, 17, 15);
	15.2	16	15	putpixel (15, 16, 15);
	16	15	16	putpixel (16, 15, 15);
	16.8	14	17	putpixel (17, 14, 15);
	17.6	13	18	putpixel (18, 13, 15);
	18.4	12	18	putpixel (18, 12, 15);
	19.2	11	19	putpixel (19, 11, 15);
B →	20	10	20	putpixel (20, 10, 15);

Limitations of DDA line generation algorithm:-

1. DDA is a floating point arithematic algorithm to draw line. Due to floating pt. arithematic rounding is required.
2. Division operation is required. Due to rounding off to nearest pixel position and division operation this algorithm is slow as compared to Bresenham's line generation algorithm.
3. Due to rounding off manually line may have jagged or staircase or aliasing effect.

Scanned by CamScanner

Circle generation algorithm:-

- Circle is a set of infinite points which are equidistant from its centre (x_c, y_c).
- Input parameter required to draw a circle is the centre coordinates of circle (x_c, y_c) and radius r.
- The equation of circle in cartesian form is given by-

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

where (x_c, y_c) - centre of circle
r - radius.

The equation of standard circle when centre at origin is $x^2 + y^2 = r^2$

Equation of circle in polar form is given as-

$$x = x_c + r\cos\theta$$
$$y = y_c + r\sin\theta$$

- Scientifically the first point on the circumference of circle when centred at origin is

$$(x_0, y_0) = (0, r)$$

- Function of circle

$$x^2 + y^2 = r^2$$
$$x^2 + y^2 - r^2 = 0$$

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

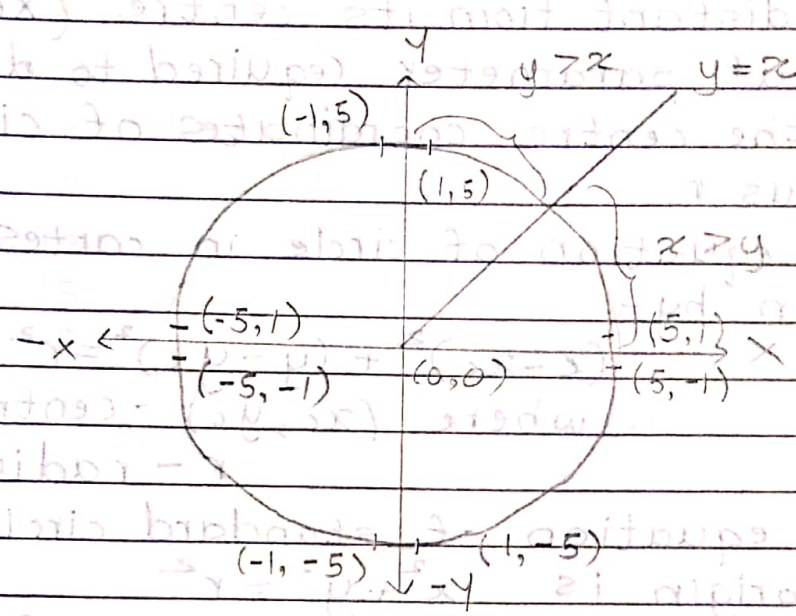
$= 0$ (x, y) lies on the boundary of circle

< 0 (x, y) inside circle

> 0 (x, y) outside circle.

Scanned by CamScanner

- Symmetricity property of circle-
circle can be divided into 2^n equal parts for symmetricity.

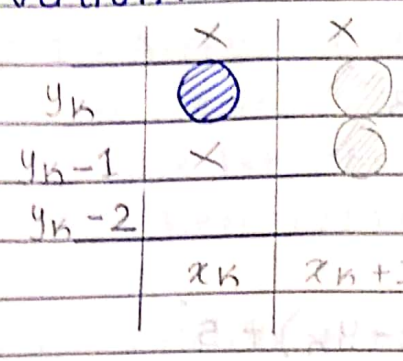



```
Symmetry of pixel (x,y)
{
    putpixel(x,y,15);
    putpixel (y,x,15);
    putpixel (-y,x,15);
    putpixel (-x,y,15);
    putpixel (-x,-y,15);
    putpixel (-y,-x,15),
    putpixel (y,-x,15);
    putpixel (x,-y,15);
}
```

Scanned by CamScanner

may-19

Mid-point circle generation algorithm derivation:-



let (x_k, y_k) be the current pixel position and (x_{k+1}, y_{k+1}) be the next pixel position along circular path such that,

$$x_{k+1} = x_k + 1$$
$$y_{k+1} = y_k \text{ or } y_k - 1$$

Decision parameter P_k

avg. of y_k and $y_k - 1$

$$P_k = f_{circle}(x_k + 1, y_k - 1/2)$$
$$P_k = (x_k+1)^2 + (y_k - 1/2)^2 - r^2$$
$$P_k = x_k^2 + 2x_k + 1 + y_k^2 - y_k + 1/4 - r^2 \quad \text{—— (1)}$$

$$P_{k+1} = f_{circle}(x_{k+1} + 1, y_{k+1} - 1/2)$$

But $x_{k+1} = x_k + 1$

$$P_{k+1} = f_{circle}(x_k + 2, y_{k+1} - 1/2)$$
$$P_{k+1} = (x_k+2)^2 + (y_{k+1} - 1/2)^2 - r^2$$
$$\cancel{P_{k+1} = (x_k+2)^2 +}$$
$$P_{k+1} = x_k^2 + 4x_k + y + y_{k+1}^2 - y_{k+1} + 1/4 - r^2 \quad \text{—— (2)}$$

(2) − (1)

$$P_{k+1} - P_k = 2x_k + 3 + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k)$$
$$P_{k+1} = P_k + 2x_k + 3 + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) \quad \text{—— (3)}$$

Scanned by CamScanner

if $P_K < 0$

$x_{K+1} = x_K + 1$

$y_{K+1} = y_K$

$P_{K+1} = P_K + 2x_K + 3$

else $P_K > 0$

$x_{K+1} = x_K + 1$

$y_{K+1} = P_K + 2(x_K - y_K) + 5$

calculate initial decision parameter P_0

$P_0 = f_{circle}(x_0 + 1, y_0 - 1/2)$

$(x_0, y_0) = (0, r)$

$P_0 = f_{circle}(0 + 1, r - 1/2)$

$= f_{circle}(1, r - 1/2)$

$P_0 = 1^2 + (r - 1/2)^2 - r^2$

$= 1 + r^2 - r + \frac{1}{4} - r^2$

$= \frac{5}{4} - r \approx 1 - r$

Scanned by CamScanner

Midpoint circle algorithm:-

Step 1:- Accept centre (x_c, y_c) and radius r of a circle.
The 1st pt. on circumference of circle when centred at origin is $(x_0, y_0) \equiv (0, r)$

Step 2:- Calculate initial decision parameter

$$P_0 = \frac{5}{4} - r \approx 1 - r$$

Step 3:- At each x_k position perform the following test -

if $P_k < 0$

{

$x_{k+1} = x_k + 1$

$y_{k+1} = y_k$

$P_{k+1} = P_k + 2x_k + 3$

}

else $P_k > 0$

{

$x_{k+1} = x_k + 1$

$y_{k+1} = y_k - 1$

$P_{k+1} = P_k + 2(x_k - y_k) + 5$

}

Step 4:- Using symmetricity property calculate the pixel positions in all other 7 octants.

Symmetry of pixel( )

{

}

Scanned by CamScanner

Step 5:- Repeat step 3 to 4 untill $x \geq y$

Step 6:- Stop

Q.] Draw a circle with centre (0,0) and radius using midpoint circle generation algo.

Step 1:- Accept center $(x_c, y_c) = (0,0)$ and radius $r = 10$ of a circle.
First pt. on the circumference of circle is
$(X_0, Y_0) = (0, r) = (0, 10)$.
putpixel$(X_0, Y_0, 15)$;

Step 2:- Calculate initial dec. parameter

$$P_0 = \frac{5}{4} - r \approx 1 - r$$

$$P_0 = 1 - 10 = -9$$

Step 3:- At each x_k position perform the following test.

if $P_k < 0$	else $P_k > 0$
$x_{k+1} = x_k + 1$	$x_{k+1} = x_k + 1$
$y_{k+1} = y_k$	$Y_{k+1} = Y_k - 1$
$P_{k+1} = P_k + 2x_k + 3$	$P_{k+1} = P_k + 2(x_k - Y_k) + 5$

Scanned by CamScanner

x_{k+1}	y_{k+1}	P_{k+1}
0	10	-9
1	10	$-9+2(0)+3=-6$
2	10	$-6+2(1)+3=-1$
3	10	$-1+2(2)+3=6$
4	9	$6+2(3-10)+5=-3$
5	9	$-3+2(4)+3=8$
6	8	$8+2(5-9)+3=3$
7	7	

Bresenham's Circle generation algorithm:-

Step 1:- Accept centre (x_c, y_c) and radius r of a circle. The first pt. on circumference of circle is $(x_0, y_0) = (x_c, y_c + r)$. putpixel $(x_0, y_0, 15)$

Step 2:- Calculate initial decision parameter
$P_0 = 3 - 2r$

Step 3:- At each x_k position perform the following test.

if $P_k < 0$
$x_{k+1} = x_k + 1$
$y_{k+1} = y_k$
$P_{k+1} = P_k + 4x_k + 6$

else $P_k > 0$
$x_{k+1} = x_k + 1$
$y_{k+1} = y_k - 1$
$P_{k+1} = P_k + 4(x_k - y_k) + 10$

Step 4:- Using symmetricity property calculate the pixel positions in all other 7 octants.

Scanned by CamScanner

Step 5 :- Repeat step 3 to 4 till $x \geq y$

Step 6 :- Stop.

Q.] Draw a circle with center (10,10) and radius 5 using Bresenham's circle generation algorithm.

Step 1 :- Accept center $(x_c, y_c) \equiv (10,10)$ and radius $r=5$ of a circle.
First pt. on the circumference of circle is $(x_0, y_0) = (x_c, y_c + r) \equiv (10,15)$
putpixel $(x_0, y_0, 15)$

Step 2 :- Calculate initial decision parameter
$P_0 = 3 - 2r = 3 - 2(5)$
$P_0 = -7$

Step 3 :- At each x_k position perform following test

if $P_k < 0$
$x_{k+1} = x_k + 1$
$y_{k+1} = y_k$
$P_{k+1} = P_k + 4x_k + 6$

else $P_k > 0$
$x_{k+1} = x_k + 1$
$y_{k+1} = y_k - 1$
$P_{k+1} = P_k + 4(x_k - y_k) + 10$

Scanned by CamScanner

x_{k+1}	y_{k+1}	P_{k+1}
10	15	-7
11	15	$-7+4(10)+6=39$
12	14	$39+4(11-15)+10=33$
13	13	

Area filling algorithm:-

1. Scanline polygon fill
2. Boundary fill
3. Floodfill

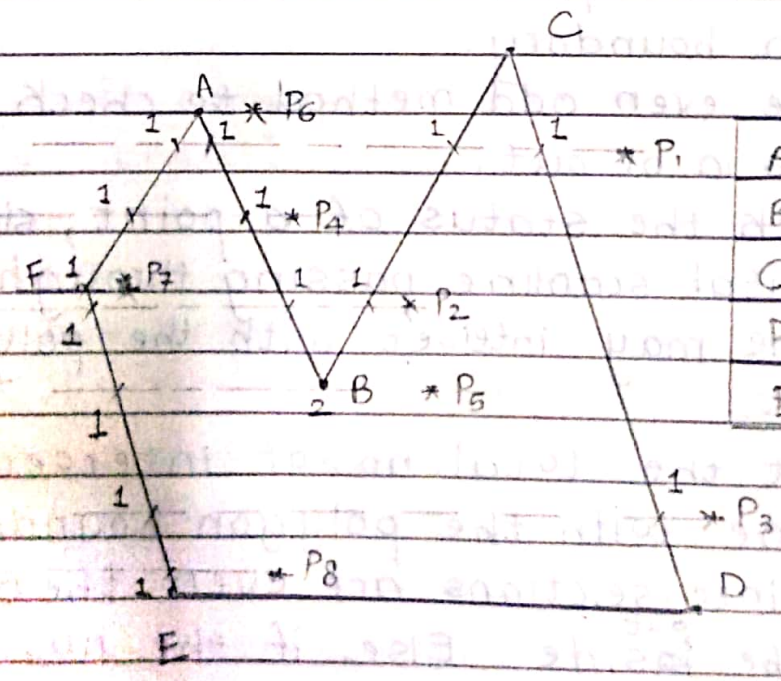
Inside-outside test -

- Inside-outside test is used to determine whether the pixel lies inside or outside of a polygon boundary.
- We use even odd method to check whether the point is in or out.
- To check the status of a point, simply draw a horizontal scanline passing through that pt.
- Scan line may intersect with the polygon edges or vertices.
- Count the total no. of intersections of a scan line with the polygon boundary, if the no. of intersections are even then the pt. should be ~~in~~ outside. Else if the no. of intersections are odd the pt. should be inside the polygon boundar.

Scanned by CamScanner

- If a scan line is passing through vertex then find other 2 adjacent vertices of that vertex.
- If both the adjacent vertices lies on the same side of scanline i.e. either above or below then count that intersection twice.
- If the adjacent vertices lies on the opposite side of scan line i.e. one above and one below then count no. of intersections only once.
- Finally count all the intersections of a scanline with the polygon, if it is even ⇒ pt. is outside otherwise pt. will be inside.

Q.] Define polygon and its types. Write a C code to draw a polygon.



A	AB
B	BC
C	CD
D	DE
E	EF
F	FA

Scanned by CamScanner

Points	No. of intersections	Status
P_1	4 (Even)	Out
P_2	3 (odd)	In
P_3	2 (Even)	Out
P_4	2 (Even)	Out
P_5	3 (odd)	In
P_6	2 (Even)	Out
P_7	1 (odd)	In
P_8	1 (odd)	In

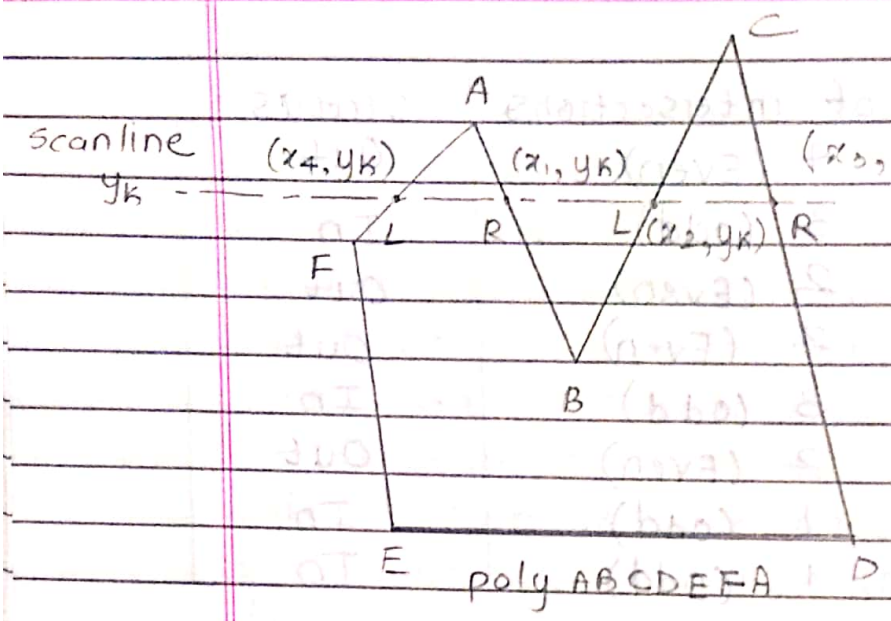
① Scanline polygon fill algorithm:-

- Scan line polygon fill algorithm proceeds scan line by scan line.
- For each scan line crossing a polygon, the area fill algorithm locates the intersection points with the polygon edges. These intersection points are then sorted on increasing order of their x values. The corresponding locations in the framebuffer are set to the specified fill colour.

Algorithm:-

Step 1:- Accept the no. of vertices in the polygon and the co-ordinates of the vertices.

Scanned by CamScanner



Vertex list
A
B
C
D
E
F

Edge list
AB
BC
CD
DE
EF
FA

Step 2:- Prepare active edge list for scan line y_k.
Active edge list contains the list of active edges which are intersected by scan line y_k.

AB
BC
CD
FA

Step 3:- calculate edge intersection pt. for scan line y_k.
It is a point of intersection of a scan line with the polygon edge.

AEL	EIP
AB	(x_1, y_k)
BC	(x_2, y_k)
CD	(x_3, y_k)
FA	(x_4, y_k)

Scanned by CamScanner

Step 4:- Prepare Sorted EIP (left to right)
Sort all edge intersection points in the increasing order of their x values.

Sorted EIP	AEL	
(x_4, y_k)	FA	L
(x_1, y_k)	AB	R
(x_2, y_k)	BC	L
(x_3, y_k)	CD	R

Step 5:- In sorted EIP the first x value gives left intersection boundary and second x value gives right intersection boundary of a scan line with polygon. Fill all the pixel positions between left and right intersection boundary with the specified fill colour value.

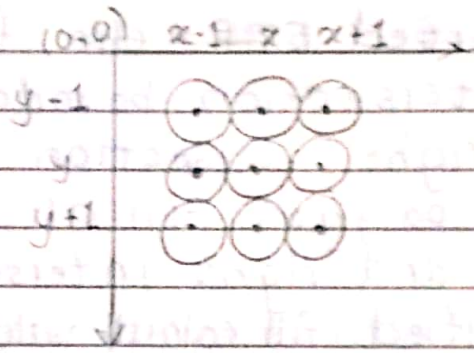
Step 6:- Repeat step ② to ⑤ for all the scan lines

Step 7:- Stop.

② Boundary fill algorithm -
- Boundary fill algorithm starts at a point inside a region and paints the interior portion outwards towards the boundary.
- If the boundary is specified in a single colour the algorithm proceeds outwards pixel by pixel until the boundary colour is encountered.
- A boundary fill algorithm accepts -
  i.) Co-ordinates of an interior pixel $P(x, y)$
  ii.) Fill colour value
  iii.) Boundary colour value.

Scanned by CamScanner

- Starting from (x, y) the algorithm tests the neighbouring pixels to determine if they are the boundary colour. If not they are painted with the fill colour and their neighbourers are tested. This process continues until all pixels upto the boundary colour for the areas are tested.
- Neighbouring pixels are tested using either 4 connected or 8 connected method as shown below -



- The recursive procedure for boundary fill is as follows -

```
boundaryfill (int x, int y, int bc, int fc)
{
  if ((getpixel (x,y) != bc) && (getpixel(x,y) != fc))
  {
     putpixel (x, y, fc);
     boundaryfill (x, y-1, bc, fc);
     boundaryfill (x+1, y-1, bc, fc);
     boundaryfill (x+1, y, bc, fc);
     boundaryfill (x+1, y+1, bc, fc);
     boundaryfill (x, y+1, bc, fc);
     boundaryfill (x-1, y+1, bc, fc);
     boundaryfill (x-1, y, bc, fc);
     boundaryfill (x-1, y-1, bc, fc);
  }
}
```

Scanned by CamScanner

~~(3) Floodfill algorithm:-~~

Limitations of boundaryfill algo-

- This algorithm is only applicable to fill a region having single boundary colour. If an object is having multiple boundary then this algorithm fails.
- More than 2 connected objects cannot befilled by this algorithm.

(3.) Floodfill algorithm-

- Flood fill algorithm is used to fill an area i.e. not defined with a single boundary colour. Floodfill algo. replaces the specified interior colour with the fill colour.
- Floodfill algorithm accepts -
  - i.) Co-ordinates of interior pixel(x,y)
  - ii.) Interior colour value (ac)
  - iii.) Desired fillcolour value (fc)
- Starting from (x,y) the algorithm tests the neighbouring pixels to determine if they are of interior colour. If YES they are painted with the desired fillcolour and their neighboures are tested.
- Neighbouring pixels are tested either using 4 connected or 8 connected method as shown below -
- Recursive procedure to fill an area using floodfill is -

Scanned by CamScanner

```
floodfill (int x, int y, int ac, int fc)
{
  if ((getpixel(x,y)==ac) && (getpixel(x,y)!=fc))
  {
      putpixel(x,y,fc);
      floodfill(x, y-1, ac,fc);
      floodfill(x+1, y-1, ac,fc);
      floodfill(x+1, y, ac,fc);
      floodfill(x+1, y+1, ac,fc);
      floodfill(x, y+1, ac,fc);
      floodfill(x-1, y+1, ac,fc);
      floodfill(x-1, y, ac,fc);
      floodfill(x-1, y-1, ac,fc);
  }
}
```

Polygon drawing and various transformations on polygon -

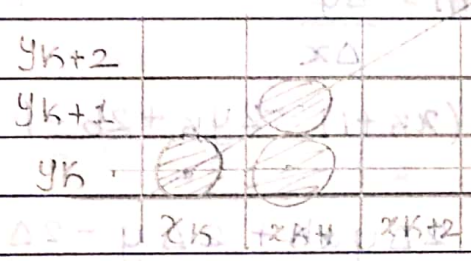
Photo (Only for theory)

Scanned by CamScanner

Bresenham line Generation:-

- ~~Br~~ In Bresenham line drawing algorithm the pixel positions along a line path are obtained by determining the pixel i.e. nearer the line at each step.

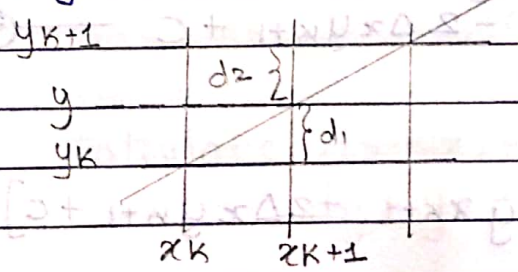
Case 1:- line AB having slope $|m| < 1$ 1
i.e. $abs(dx) \geq abs(dy)$



Let (x_k, y_k) be the current pixel position and (x_{k+1}, y_{k+1}) be the next pixel position along the line path such that -

$$x_k = x_k + 1$$

$$y_k = y_k \text{ or } y_{k+1}$$



Eq^n of line AB

$$y = mx + b$$

but at $x = x_k + 1$

$$y = m(x_k + 1) + b$$

$$d_1 = y - y_k$$

$$d_1 = [m(x_k + 1) + b] - y_k \quad \text{— (1)}$$

$$d_2 = y_k + 1 - y$$

$$d_2 = y_k + 1 - [m(x_k + 1) + b] \quad \text{— (2)}$$

Scanned by CamScanner

$$\therefore d_1 - d_2 = [m(x_k+1)+b-y_k] - [y_k+1-m(x_k+1)-b]$$

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

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

Now $m = \frac{\Delta y}{\Delta x}$

$$d_1-d_2 = 2\frac{\Delta y}{\Delta x}(x_k+1)-2y_k+2b-1$$

$$\Delta x(d_1-d_2) = 2\Delta y x_k + 2\Delta y - 2\Delta x y_k + 2b\Delta x - \Delta x$$

$$P_k = \Delta x(d_1-d_2) = 2\Delta y x_k - 2\Delta x y_k + C$$

where $C = 2\Delta y + 2b\Delta x - \Delta x$

$$P_k = 2\Delta y x_k - 2\Delta x y_k + C \quad \text{—— (3)}$$

$$P_{k+1} = 2\Delta y x_{k+1} - 2\Delta x y_{k+1} + C \quad \text{—— (4)}$$

(4) - (3)

$$P_{k+1} - P_k = [2\Delta y x_{k+1} - 2\Delta x y_{k+1} + c] - [2\Delta y x_k - 2\Delta x y_k + c]$$

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

Scanned by CamScanner

if $P_K < 0$ $\quad d_1 - d_2 < 0$ $\quad d_1 < d_2$

$x_{K+1} = x_K + 1$

$y_{K+1} = y_K$

$P_{K+1} = P_K + 2\Delta y (x_{K+1} - x_K) - 2\Delta x (y_K - y_K)$

$= P_K + 2\Delta y$

else $P_K > 0$, $d_1 - d_2 > 0$, $d_1 > d_2$

$x_{K+1} = x_K + 1$

$y_{K+1} = y_K + 1$

$P_{K+1} = P_K + 2\Delta y - 2\Delta x$

Generalised Bresenham algorithm:-

Step 1:- Accept the end pt. co-ordinates of line AB. i.e. $A(x_1, y_1)$ and $B(x_2, y_2)$.

putpixel $(x_1, y_1, 15)$

Step 2:- Calculate $dx = x_2 - x_1$

$dy = y_2 - y_1$

Step 3:- if (abs(dx) >= abs(dy))

(a) Calculate initial decision parameter

$P_0 = 2 * abs(dy) - abs(dx)$

(b) At each x_K position perform the following test

if $P_K < 0$ $\left[\begin{matrix} d_1 - d_2 < 0 \\ d_1 < d_2 \end{matrix}\right]$

$Xinc = \dfrac{dx}{abs(dx)}$

$Yinc = 0$

$P_{K+1} = P_K + 2 * abs(dy)$

else $P_K > 0$ $\left[\begin{matrix} d_1 - d_2 > 0 \\ d_2 < d_1 \end{matrix}\right]$

$Xinc = \dfrac{dx}{abs(dx)}$

$Yinc = \dfrac{dy}{abs(dy)}$

$P_{K+1} = P_K + 2 * abs(dy) - 2abs(dx)$

Scanned by CamScanner

Step 4:- if (abs(dy) > abs(dx)) $|m| > 1$

(a) Calculate initial decision parameter

$P_0 = 2 * abs(dx) - abs(dy)$

(b) At each x_k position perform following test -

if $P_k < 0$	else $P_k > 0$
$x_{inc} = 0$	$x_{inc} = \frac{dx}{abs(dx)}$
$y_{inc} = \frac{dy}{abs(dy)}$	$y_{inc} = \frac{dy}{abs(dy)}$
$P_{k+1} = P_k + 2 * abs(dx)$	$P_{k+1} = P_k + 2 * abs(dx) - 2\,abs(dy)$

Step 5:- putpixel $(x_{k+1}, y_{k+1}, 15)$

Step 6:- Repeat (3), (4), (5) until $x_{k+1} != x_2$ and/OR $y_{k+1} != y_2$

Q] Draw a line between end pt. co-ordinates A(10,20) to B(20,12) using Bresenham's line generation algorithm.

Soln:-

Step 1:- Accept end pt co-ordinates of line AB i.e. $A(x_1, y_1) = (10, 20)$ and B (20,12)

Step 2:- Calculate $dx = x_2 - x_1 = 10$

$dy = y_2 - y_1 = -8$

Step 3:- Here (abs(dx) ≥ abs(dy)) [$m \leq 1$]

(a) Cal initial dec. parameter

$P_0 = 2 * 8 - 10 = 6$

Scanned by CamScanner

(b.) At each x_k position perform the follo. test -

if $P_k < 0$

$xinc = \frac{dx}{abs(dx)} = 1$

$yinc = 0$

$P_{k+1} = P_k + 2 * abs(dy)$

$P_{k+1} = P_k + 16$

else $P_k > 0$

$xinc = \frac{dx}{abs(dx)} = 1$

$yinc = \frac{dy}{abs(dy)} = -1$

$P_{k+1} = P_k + 2 \cdot abs(dy) - 2$

$P_{k+1} = P_k - 4$

putpixel(x_{k+1}, y_{k+1}, 15)	x_k+1	y_k+1	P_k+1	P_k+1
	10	20	6-4 =2	6
	11	19	2-4 = -2	6-4=2
	12	18	-2+16 =14	2-4 = -2
	13	18	14-4 =10	-2+16=14
	14	17	10-4=6	14-4=10
	15	16	6-4 =2	10-4=6
	16	15	2-4 = -2	6-4 =2
	17	14	-2+16 =14	2-4 = -2
	18	14		-2+16=14
	19	13		14-4=10
	20	12		

Scanned by CamScanner

Geometric Transformations

- Any transformation that deals with the geometry of an object such as shape, size, orientation, angle focal length, etc is called as geometric transformation.

- If the geometry of an object changes in X-Y plane then it is called as 2-d geometric transformation.

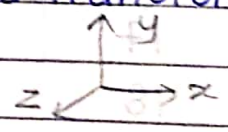
- If the geometry of an object changes in X-Y-Z plane then it is called as 3-d geometric transformations.

Geometric Transformations

2D Transformations

- Translation
- Rotation
- Scaling
- Reflection
- Shear

3D Transformations



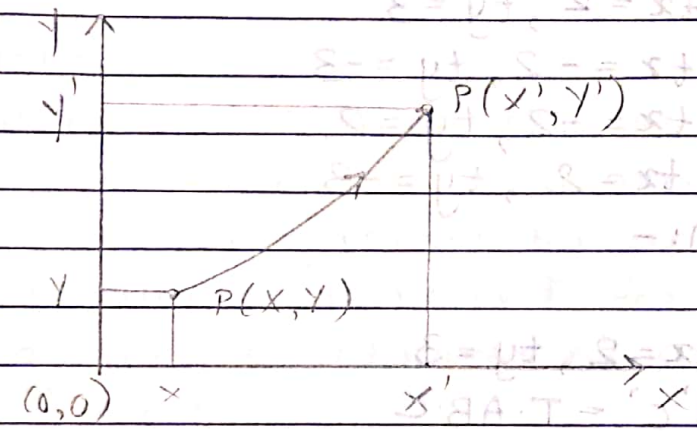
- Translation
- Rotation
- Scaling
- Reflection.

2-D Translation :-

- Translation transformation translates an object in a straight line path from one co-ordinate position to new co-ordinate.
- Input parameter required for 2-D translations are — end pt co-ordinates of an object and tx and ty i.e. translation parameter along

Scanned by CamScanner

x and y direction respectively



Input Point $P(X,Y)$
tx – Translation vector along X-axis
ty – Translation vector along Y-axis
Output Point $P'(X',Y')$

$$x' = x + tx \quad \text{—} \; (1)$$
$$y' = y + ty \quad \text{—} \; (2)$$

$$P' = T \cdot P$$

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

Note:- when tx and ty both are +ve then the object gets translated away from the origin.
when tx and ty both are -ve then object gets translated towards origin.

Scanned by CamScanner

Q.] Translate an object ABC having end pt. co-ordinates A(3,3), B(6,3), C(4,5) using

i.) $tx = 2, ty = 3$

ii.) $tx = -2, ty = -3$

iii.) $tx = -2, ty = 2$

iv.) $tx = 2, ty = -3$

Soln:-

i.) $tx = 2, ty = 3$

$A'B'C' = T \cdot ABC$

$$= \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3 & 6 & 4 \\ 3 & 3 & 5 \\ 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 5 & 8 & 6 \\ 6 & 6 & 8 \\ 1 & 1 & 1 \end{bmatrix}$$

$A' = (5,6)$, $B' = (8,6)$, $C' = (6,8)$

ii.) $tx = -2, ty - 3$

$A'B'C' = T \cdot ABC$

$$= \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3 & 6 & 4 \\ 3 & 3 & 5 \\ 1 & 1 & 1 \end{bmatrix}$$

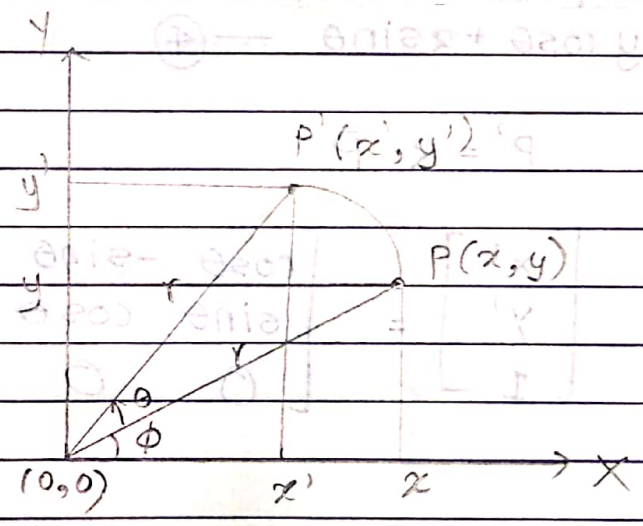
$$= \begin{bmatrix} 1 & 4 & 2 \\ 0 & 0 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$

Scanned by CamScanner

$A' = (1,0)$, $B' = (4,0)$, $C' = (2,2)$

2-D rotation -

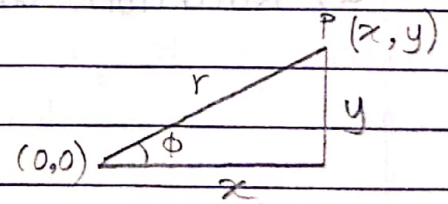
Rotation transformation rotates an object in a circular path from one co-ordinate position to new co-ordinate position by an angle θ.
Input parameter required for 2-D rotation is the end point co-ordinates of an object and angle of rotation θ.



Input - Point $P(x,y)$

θ - Angle of rotation

o/p $P'(x',y')$



$$\cos\phi = \frac{x}{r}$$

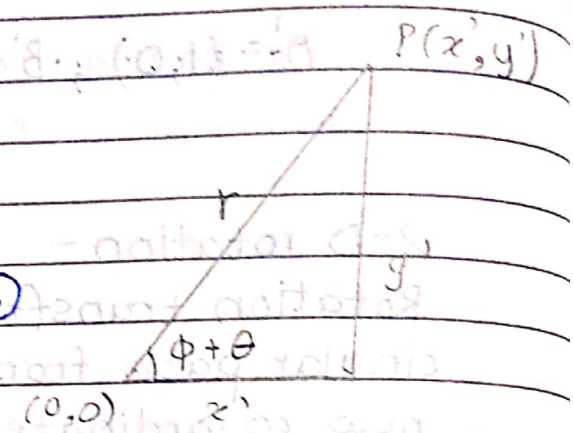
$$x = r\cos\phi \quad -(1)$$

$$\sin\phi = \frac{y}{r}$$

$$y = r\sin\phi \quad -(2)$$

Scanned by CamScanner

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



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

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

$$x' = x\cos\theta - y\sin\theta \quad \text{— (3)}$$

$$\sin(\phi+\theta) = \frac{y'}{r}$$

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

$$= r\sin\phi\cos\theta + r\cos\phi\sin\theta$$

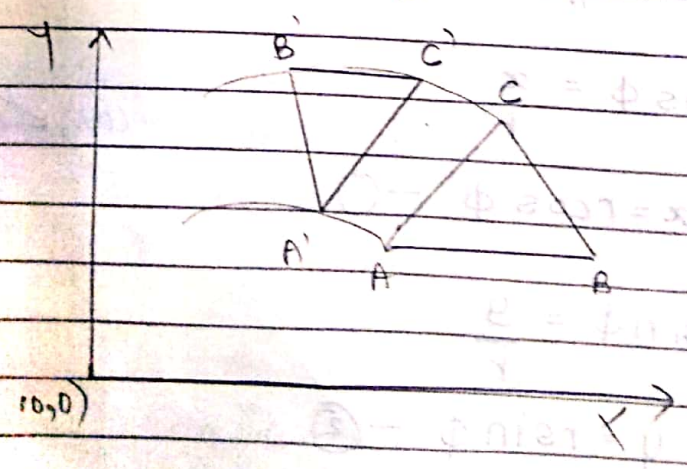
$$y' = y\cos\theta + x\sin\theta \quad \text{— (4)}$$

$$P' = R \cdot P$$

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

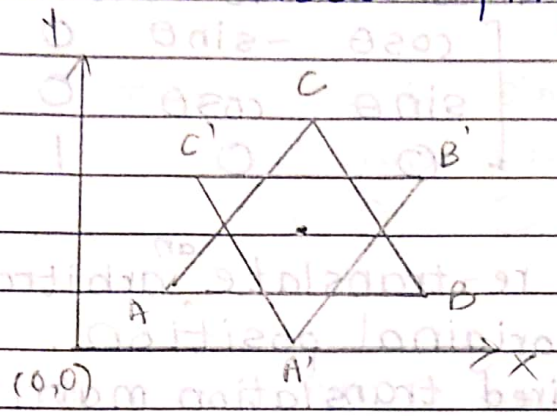
Types of Rotation:-

a.) Rotation about (0,0)

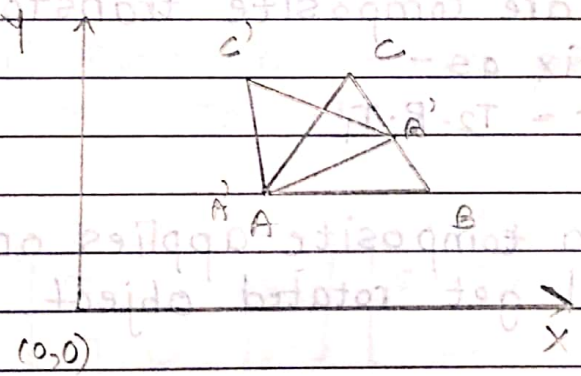


Scanned by CamScanner

b) Rotation about pivot pt. :-



c.) Rotation about arbitrary pt. :-



Algorithm for rotation of an object about arbitrary point:-

Step 1:- Translate an arbitrary pt. to origin. The required translation transformation matrix is -

$$T_1 = \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix}$$

Step 2:- Now rotate the object about origin by an angle θ

Scanned by CamScanner

The required rotation matrix is -

$$R = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Step 3:- Now re-translate an arbitrary pt. to its original position.

Required translation matrix is -

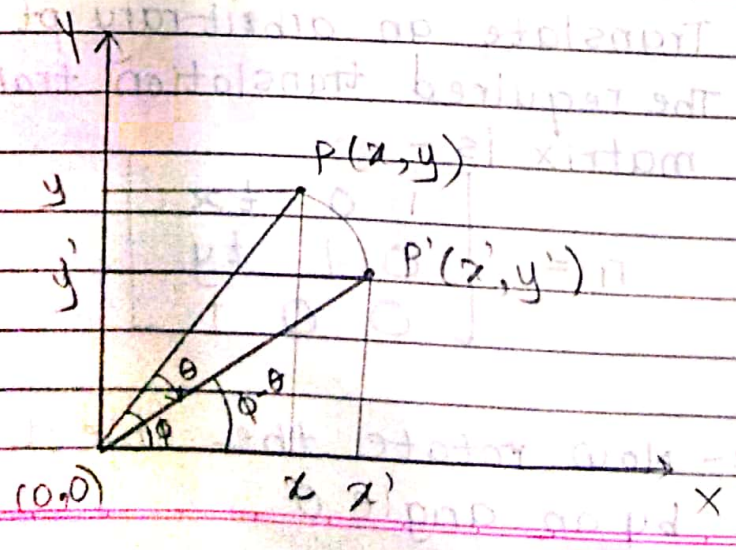
$$T_2 = \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix}$$

Step 4:- Prepare composite transformation matrix as -

$$MT = T_2 \cdot R \cdot T_1$$

when tomposite applies on object we will get rotated object.

Inverse rotation:-
(Clockwise rotation)



Scanned by CamScanner

put $\theta = -\theta$ in R -

$$R^{-1} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Q.] Rotate an object ABCD by angle $\theta = 90°$
i.) about origin
ii.) about pivot point PC(4,5)
iii.) about pt. A(4,2)

Soln:-

iii.) Rotation about pt. A

$M_T = T_2 \cdot R \cdot T_1$

$$= \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 90 & -\sin 90 & 0 \\ \sin 90 & \cos 90 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -4 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & -1 & 4 \\ 1 & 0 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -4 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$M_T = \begin{bmatrix} 0 & -1 & 6 \\ 1 & 0 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

Scanned by CamScanner

$$A'B'C'D' = M_T \cdot ABCD$$

$$= \begin{bmatrix} 0 & -1 & 6 \\ 1 & 0 & -2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 & 7 & 4 & 1 \\ 2 & 6 & 8 & 6 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 4 & 0 & -2 & 0 \\ 2 & 5 & 2 & -1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

(i) Rotation about (0,0)

$$A'B'C'D' = R \cdot ABCD$$

$$= \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 & 7 & 4 & 1 \\ 2 & 6 & 8 & 6 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} -2 & -6 & -8 & -6 \\ 4 & 7 & 4 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

ii) Rotation about pivot point $P_c(4,5)$

$$M_T = T_2 \cdot R \cdot T_1$$

$$= \begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 90 & -\sin 90 & 0 \\ \sin 90 & \cos 90 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -4 \\ 0 & 1 & -5 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -4 \\ 0 & 1 & -5 \\ 0 & 0 & 1 \end{bmatrix}$$

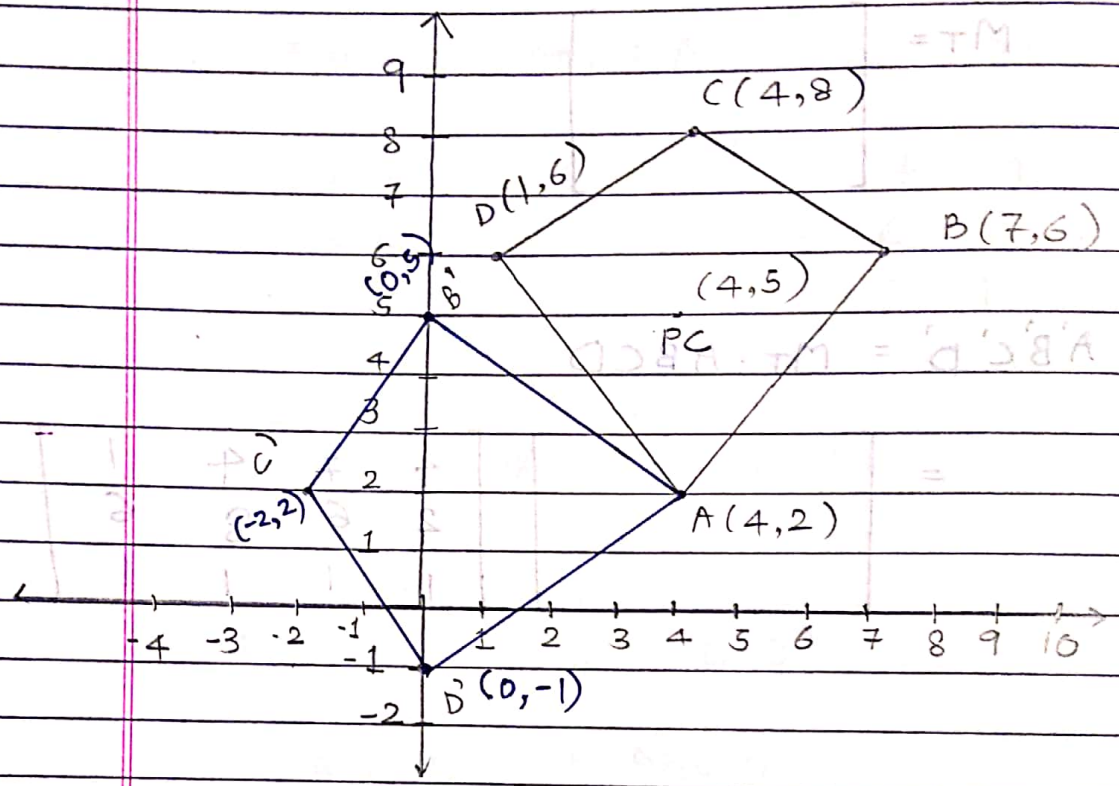
Scanned by CamScanner

$$M_T = \begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}$$

$$A'B'C'D' = M_T \cdot ABCD$$

$$= \begin{bmatrix} & & \\ & & \\ & & \end{bmatrix} \begin{bmatrix} 4 & 7 & 4 & 1 \\ 2 & 6 & 8 & 6 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Scanned by CamScanner



2-D Scaling Transformation:-

- Scaling Transformation alters (either shrink or swell) the size of an object.
- Input parameter required for 2-D scaling are the end pt. co-ordinates of an object and sx and sy i.e. scaling factor along x and y axis respectively.
- The equation for scaling of width and height of an object are

$$x' = Sx \cdot x$$

$$y' = Sy \cdot y$$

$$P' = S \cdot P$$

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

Scanned by CamScanner

where $S = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$ is known as basic 2-D scaling transformation matrix which scales the object about origin only.

Note:-

1. If $S_x = S_y > 1$
then object size will increase and object moves away from origin.

2. If $S_x = S_y < 1$
then object size will decrease and object moves towards origin.

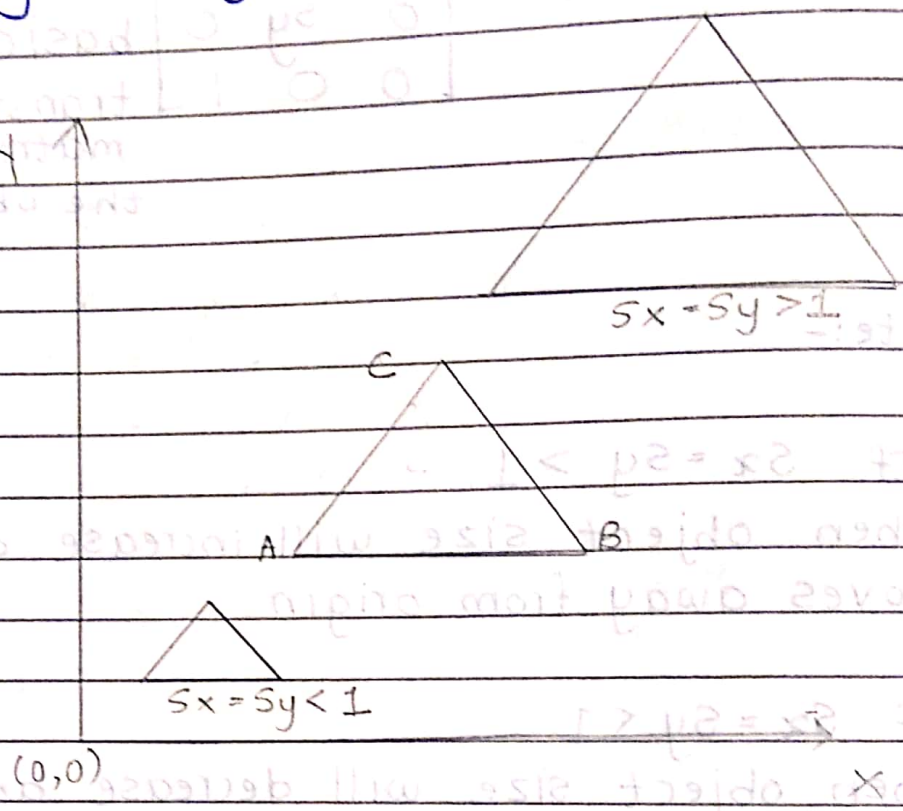
3. If $S_x = S_y = 1$
then there will be no change in the size of the object as well as the position will remain unchanged.

Types of Scaling:-

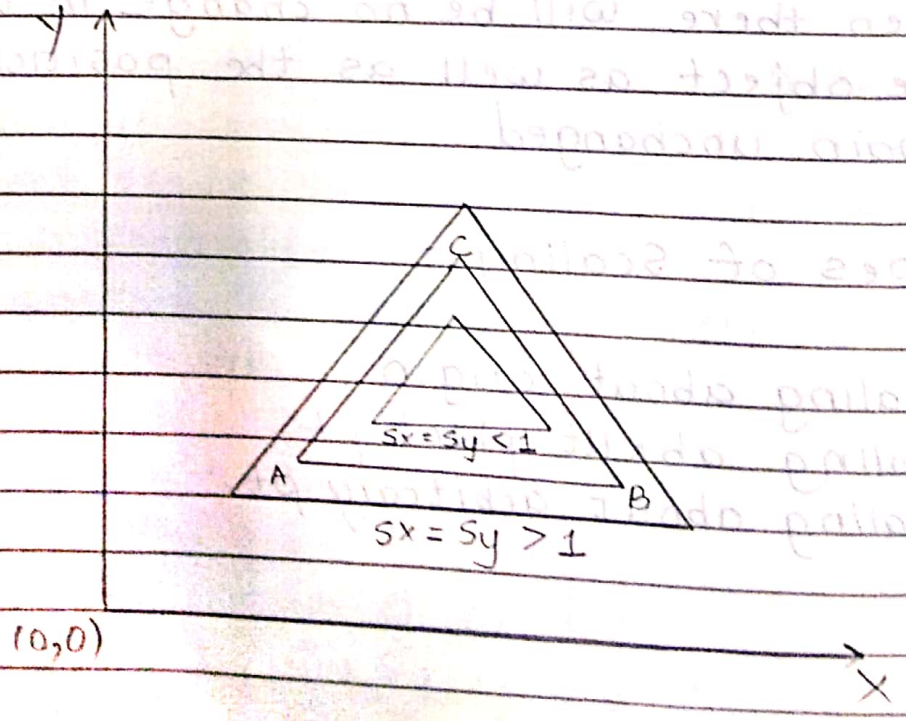
1. Scaling about origin
2. Scaling about pivot pt.
3. Scaling about arbitrary pt.

Scanned by CamScanner

① Scaling @ origin

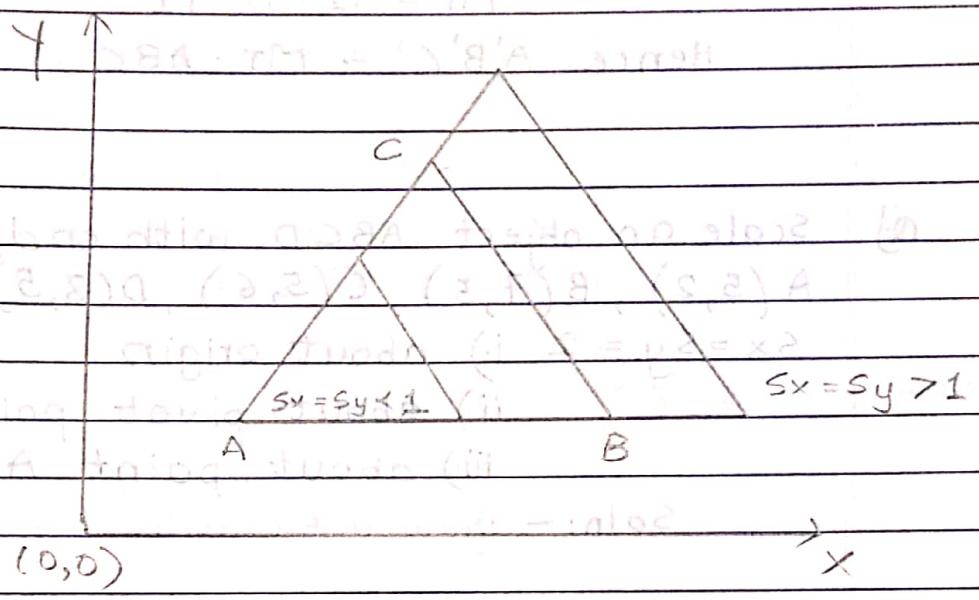


② Scaling @ pivot pt.



Scanned by CamScanner

③ Scaling @ arbitrary pt. @ A



Algorithm for arbitrary point scaling –

Step 1:- Translate an arbitrary pt. to origin.
The required translation transformation matrix is $T_1 = \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix}$

Step 2:- Now scale the object about origin
The required scaling matrix is $S = \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Step 3:- Now retranslate an arbritary pt. to its original position. The required translation matrix is $T_2 = \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix}$

Scanned by CamScanner

Step 4:- Compute composite matrix

$$M_T = T_2 \cdot S \cdot T_1$$

Hence $A'B'C' = M_T \cdot ABC$

Q.] Scale an object ABCD with end pt. co-ordinates A(5,2), B(7,5), C(5,6), D(3,5) with $S_x = S_y = 2$ i.) about origin

ii.) about pivot point (5,4)

iii.) about point A(5,2)

Soln:- Draw diagram in exam

i.) Scaling @ origin

$A'B'C'D' = ABCD$

$$= \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 & 7 & 5 & 3 \\ 2 & 5 & 6 & 5 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 10 & 14 & 10 & 6 \\ 4 & 10 & 12 & 10 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

ii.) Scaling @ pivot pt.

$$M_T = T_2 \cdot S \cdot T_1$$

$$= \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Scanned by CamScanner

$$= \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -5 \\ 0 & 1 & -4 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 0 & 5 \\ 0 & 2 & 4 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -5 \\ 0 & 1 & -4 \\ 0 & 0 & 1 \end{bmatrix}$$

$$M_T = \begin{bmatrix} 2 & 0 & -5 \\ 0 & 2 & -4 \\ 0 & 0 & 1 \end{bmatrix}$$

$A'B'C'D' = M_T \cdot ABCD$

iii.) Scaling about pt. $A(5,2)$

$M_T = T_2 \cdot S \cdot T_1$

$$= \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -5 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}$$

$A'B'C'D' = M_T \cdot ABCD$

$$= \begin{bmatrix} & & \\ & & \\ & & \end{bmatrix} \begin{bmatrix} 5 & 7 & 5 & 3 \\ 2 & 5 & 6 & 5 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Scanned by CamScanner

Q] Prove that 2-D successive rotations are additive.

Soln:-

T.P.T $R(\theta_1) \cdot R(\theta_2) = R(\theta_1 + \theta_2)$

let $R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$

be the 2D Rotation matrix @ origin

$L.H.S = R(\theta_1) \cdot R(\theta_2)$

$$= \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta_1 \cdot \cos\theta_2 - \sin\theta_1 \cdot \sin\theta_2 & -\cos\theta_1 \sin\theta_2 - \sin\theta_1 \cos\theta_2 & 0 \\ \sin\theta_1 \cdot \cos\theta_2 + \cos\theta_1 \cdot \sin\theta_2 & -\sin\theta_1 \cdot \sin\theta_2 + \cos\theta_1 \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) & 0 \\ \sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$= R(\theta_1 + \theta_2)$

$= R.H.S$

$\therefore$ Hence proved

$R(\theta_1) \cdot R(\theta_2) = R(\theta_1 + \theta_2)$

Scanned by CamScanner

Q.] Prove that 2D rotation and scaling comutes if and only if $S_x = S_y$ and $\theta = n\pi$

Soln:-

T.P.T $R \cdot S = S \cdot R$

let $R = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$ be the 2D rotation matrix

$S = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$ be the 2D scaling matrix.

$L.H.S = R \cdot S$

$$= \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} S_x\cos\theta & -S_y\sin\theta & 0 \\ S_x\sin\theta & S_y\cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{——— (1)}$$

$R.H.S = S \cdot R$

$$= \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} S_x\cos\theta & -S_x\sin\theta & 0 \\ S_y\sin\theta & S_y\cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{——— (2)}$$

Scanned by CamScanner

Now put $S_x = S_y$ and $\theta = n\pi$ in ①, ②

$\sin\theta = \sin n\pi = 0$

$\cos n\pi = (-1)^n$

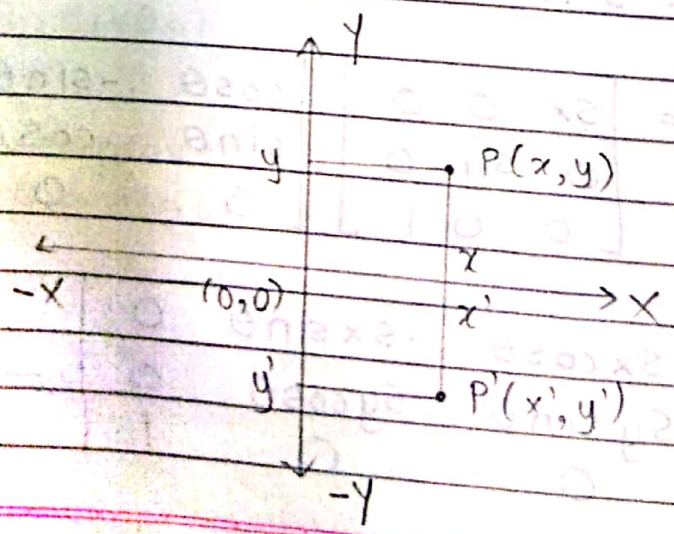
$\therefore R \cdot S = S \cdot R$

$$\begin{bmatrix} S_x(-1)^n & 0 & 0 \\ 0 & S_x(-1)^n & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} S_x(-1)^n & 0 & 0 \\ 0 & S_x(-1)^n & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2-D Reflection :-

- Reflection transformation reflects an object about the axis of reflection.
- The input parameter required for 2D reflection are the end pt. co-ordinates of an object and the axis of reflection (mirror).
- There are 2 types of 2-D reflection.
  i) Reflection @ x axis
  ii) Reflection @ y axis

i) Reflection @ x axis –



Scanned by CamScanner

Input :- point $P(x, y)$
Axis of reflection.

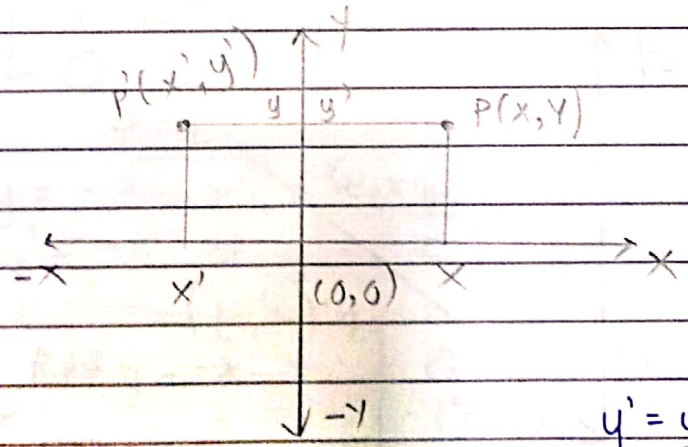
O/p :- $p'(x', y')$

$x' = x$ —— ①
$y' = -y$ —— ②

$p' = Ref \cdot P$

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

ii) Reflection @ y axis



$y' = y$
$x' = -x$

$p' = Ref \cdot P$

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

Scanned by CamScanner

Q.] Give matrix representation for following reflection –

i.) Reflection about line $y=0$ (x-axis)

ii.) Reflection about line $x=0$ (y-axis)

iii.) ~~About~~ Reflection about line $y=x$

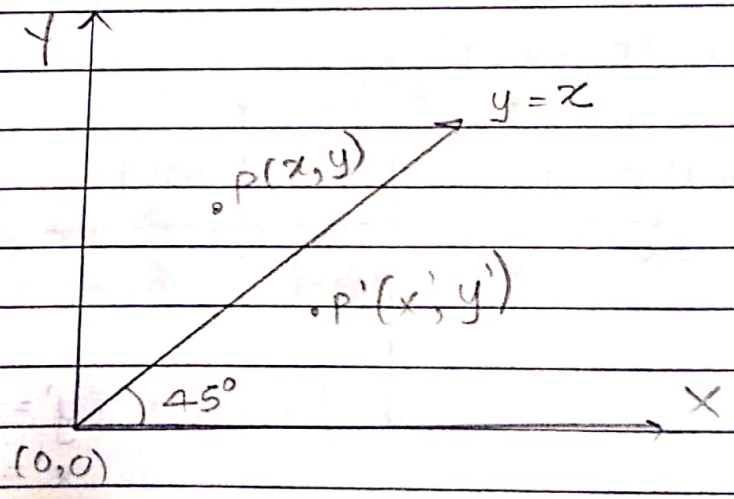
iv.) About line $y=-x$

iii.) $y = x' \Rightarrow x' = y$

$y' = x \Rightarrow y' = x$

$$P' = Ref_{y=x} \cdot P$$

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



$$R_1 = \begin{bmatrix} \cos 45 & \sin 45 & 0 \\ -\sin 45 & \cos 45 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$ Rotate line $x=y$ by $\theta = 45°$ in clockwise direction

$$Ref_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$ Reflect pt. $P(x,y)$ w.r.t x axis

Scanned by CamScanner

$$R_2 = \begin{bmatrix} \cos 45 & -\sin 45 & 0 \\ \sin 45 & \cos 45 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$ Rotate line $y = x$ by $\theta = 45°$ anticlockwise direction

$$M_T = R_2 \cdot Ref_x \cdot R_1$$

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

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

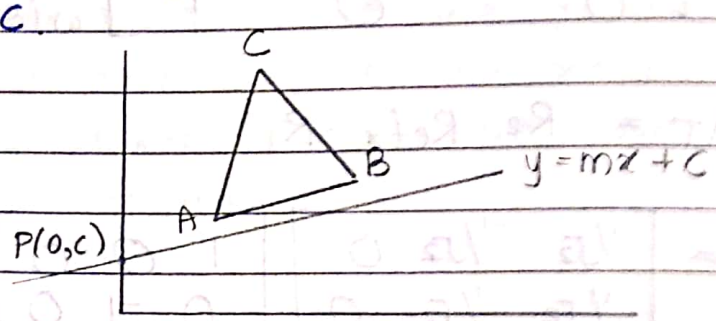
$$= \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

iv.) $y = -x' \Rightarrow x' = -y$
$y' = -x \Rightarrow y' = -x$

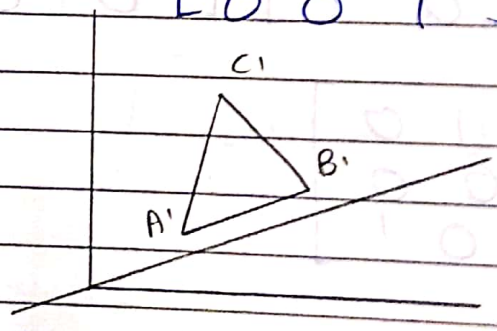
$$Ref_{y=-x} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Scanned by CamScanner

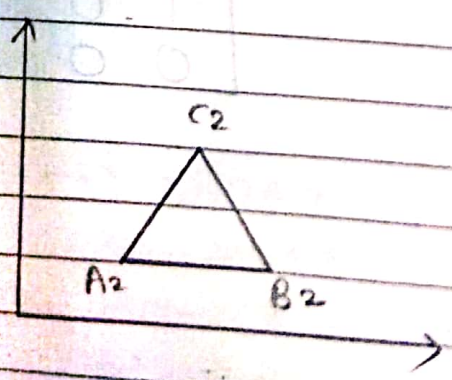
Q.] Give composite transformation matrix to reflect an object ABC above the line $y = mx + c$.



Step 1: Translate point P(0,c) to origin. The required translation transformation matrix is $T_1 = \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -c \\ 0 & 0 & 1 \end{bmatrix}$



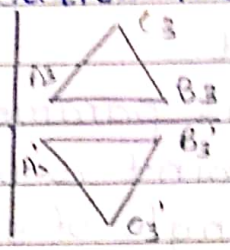
Step 2: Now rotate an object by an angle θ in clockwise direction. The required rotation transformation matrix is $R_1 = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$



Scanned by CamScanner

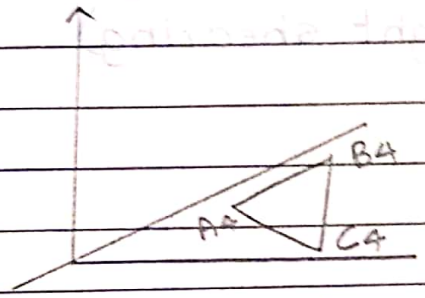
Step3: Now reflect an object x axis.
Required reflection transformation matrix is

$$R_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



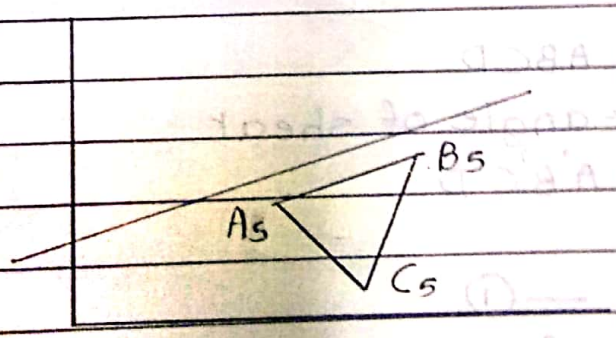
Step4:- Now rotate an object by angle θ in an anticlockwise direction. The required rotation matrix is

$$R_2 = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



step5:- Now translate point P to its original position. The required translation matrix is

$$T_2 = \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & c \\ 0 & 0 & 1 \end{bmatrix}$$



Step6: Composite Transformation matrix

$$M_T = T_2 R_2 Ref_x R_1 T_1$$

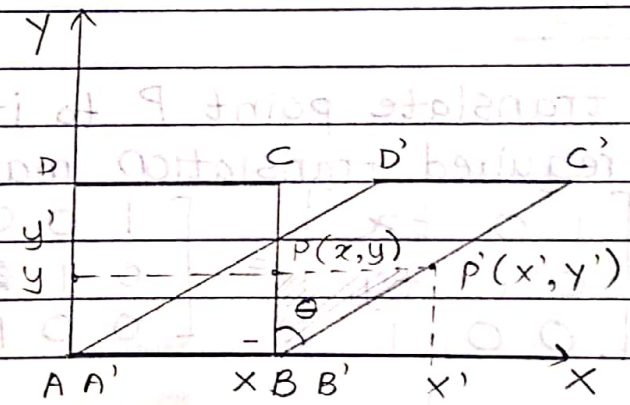
$$A'B'C' = M_T \cdot ABC$$

Scanned by CamScanner

Q.] 2D Shear Transformation

- Shear transformation slants the surface of an object.
- Input parameter required for 2D shear are the end point co-ordinates of an object and θ i.e. angle of shear or (Shx or Shy) i.e. shear constants along x and y direction.
- There are 2 types of shear transformation.
  1.) Shear along x (width shearing)
  2.) Shear along y (height shearing)

1.) Shear along x



I/P : obj ABCD

θ = angle of shear

O/P : obj A'B'C'D'

$$y' = y \quad \text{—} \; (1)$$

$$\tan\theta = \frac{x' - x}{y}$$

$$x' = x + y\tan\theta \quad \text{—} \; (2)$$

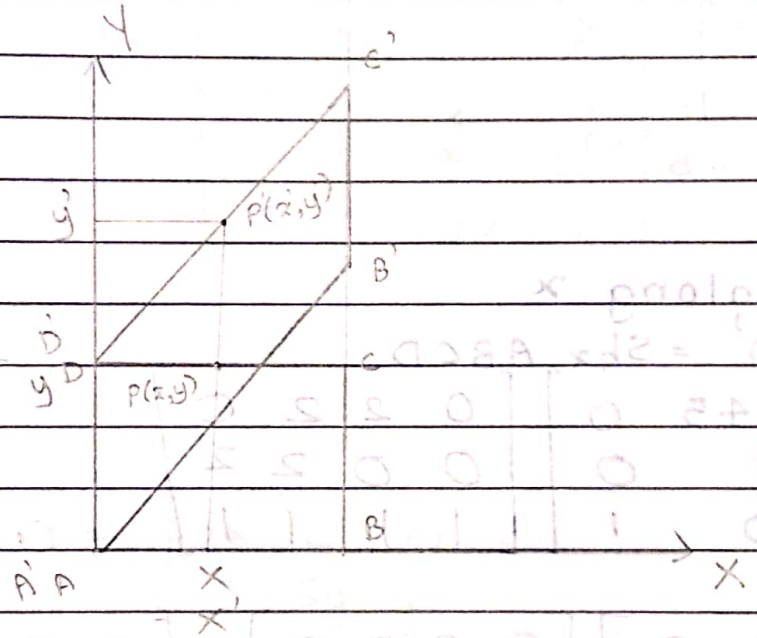
$\tan\theta \Rightarrow Sh_x$

Scanned by CamScanner

$$P' = Sh \cdot P$$

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

2.) Shear along y



$$x' = x \quad \text{—— (1)}$$

$$\tan\theta = \frac{y' - y}{x}$$

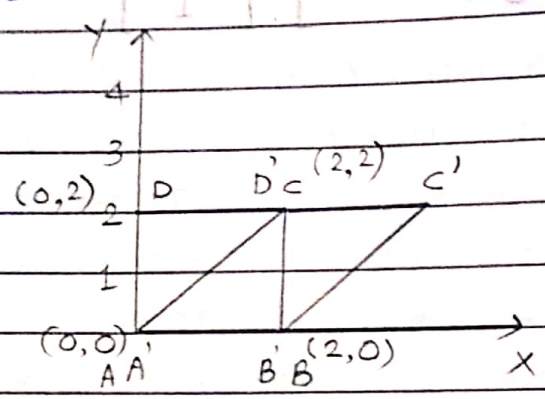
$$y' = y + x\tan\theta \quad \text{—— (2)}$$

$$P' = Shy \cdot P$$

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

Scanned by CamScanner

Q.] Shear an object ABCD having end point co-ordinates A(0,0), B(2,0) C(2,2), D(0,2) along x and y axis by angle $\theta = 45°$



a.) Shear along x

$A'B'C'D' = Sh_x\ ABCD$

$$= \begin{bmatrix} 1 & \tan 45 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 2 & 0 \\ 0 & 0 & 2 & 2 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 2 & 0 \\ 0 & 0 & 2 & 2 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 4 & 2 \\ 0 & 0 & 2 & 2 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

b.) Shear along y

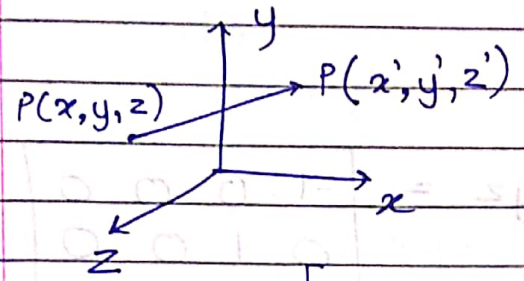
$A'B'C'D' = Sh_y\ ABCD$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 2 & 2 & 0 \\ 0 & 0 & 2 & 2 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 2 & 0 \\ 0 & 2 & 4 & 2 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Scanned by CamScanner

3-d Geometric Transformation:-

1. 3D - Translation



$x' = x + tx$
$y' = y + ty$
$z' = z + tz$

$$T = \begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2. 3D - Scaling

$$S = \begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3. 3D - Reflection

a. Along xy plane

$x' = x$
$y' = y$
$z' = -z$

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

Scanned by CamScanner

(b.) Along yz plane

$x' = -x$
$y' = y$
$z' = z$

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

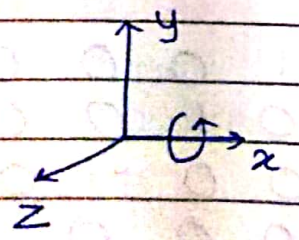
(c.) Along xz plane

$x' = x$
$y' = -y$
$z' = z$

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

(4.) 3D - Rotation

(a.) Rotation @ X axis

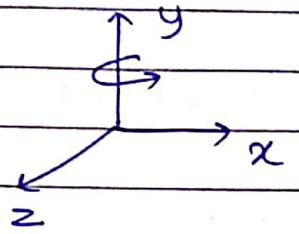


$x' = x$
$y' = y\cos\theta - z\sin\theta$
$z' = y\sin\theta + z\cos\theta$

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Scanned by CamScanner

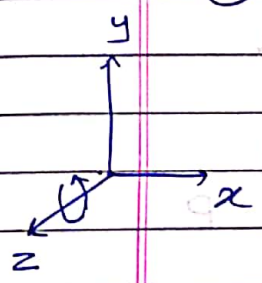
(b) Rotation @ y-axis



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

$$R_y = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(c) Rotation @ z-axis



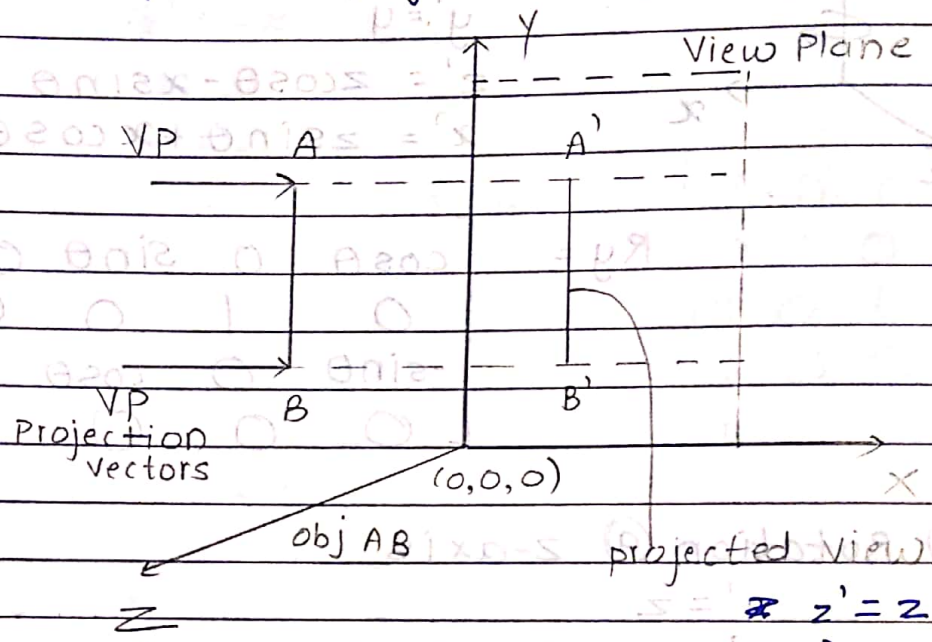
$$z' = z$$
$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$

$$R_z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Scanned by CamScanner

3D-Projection transformations.

(a.) 3D-parallel projection -

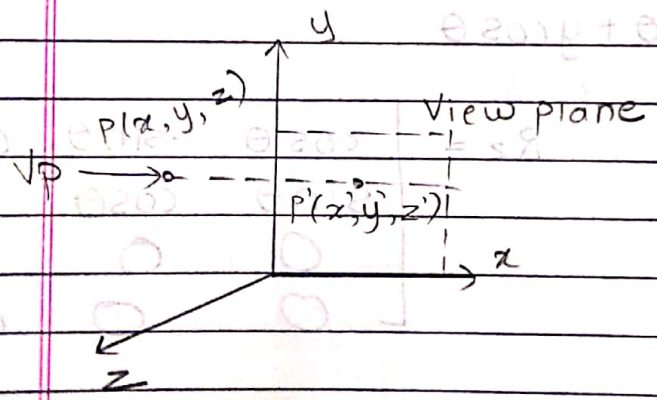


$z' = z$

$x' = x$

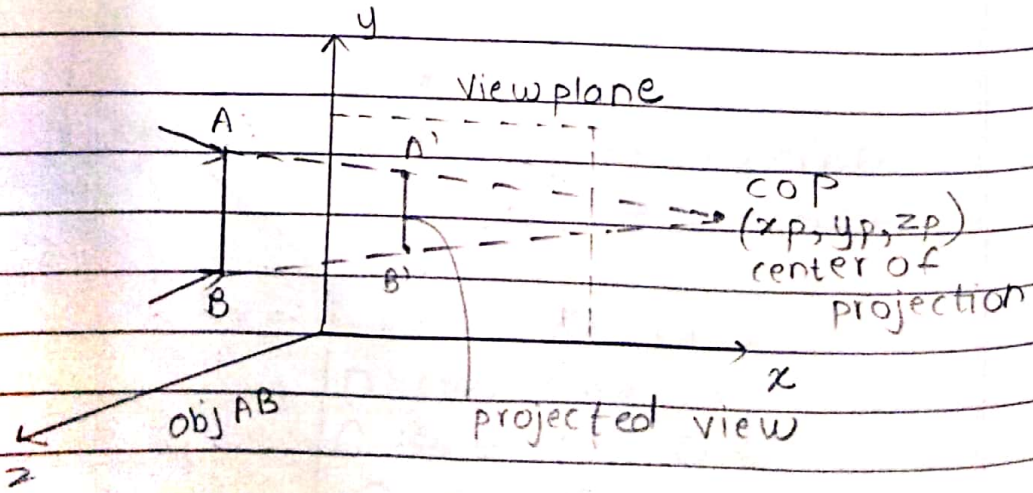
$y' = y$

$P' = M_{or} \cdot P$

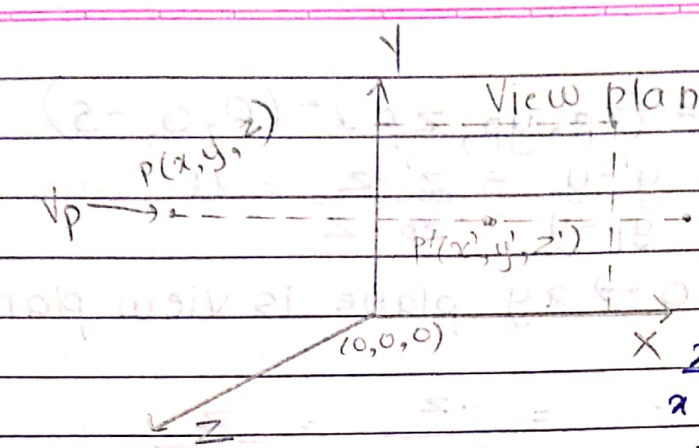


$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

(b.) Perspective projection



Scanned by CamScanner



$0 \le u \le 1$

$$\frac{x'-x}{x_p-x} = \frac{y'-y}{y_p-y} = \frac{z'-z}{z_p-z} = u$$

i/p - point $P(x, y, z)$
COP (x_p, y_p, z_p)

$$\frac{z'-z}{z_p-z} = u \quad [z'=0]$$

VP is XY plane

$$\boxed{u = \frac{-z}{z_p-z}}$$

o/p - point $P'(x', y', z')$

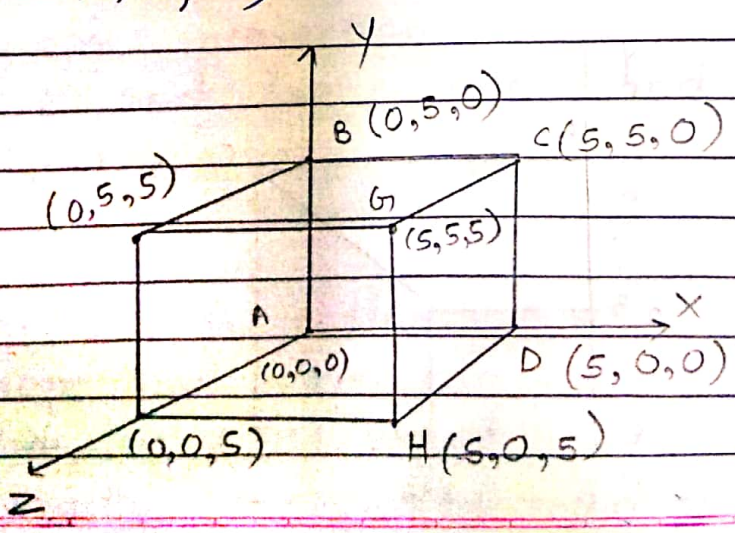
$$\frac{x'-x}{x_p-x} = u$$

$$\boxed{x = x + u(x_p-x)}$$

$$\frac{y'-y}{y_p-y} = u$$

$$\boxed{y' = y + u(y_p-y)}$$

Q] Consider a cube of side 5 unit each resting on the origin and the sides of cubes are alligned to real axis. Find the perspective projection of a cube where centre of projection (COP) = (0, 0, -5)



Scanned by CamScanner

$$COP = (x_p, y_p, z_p) = (0, 0, -5)$$

$$\frac{x' - x}{x_p - x} = \frac{y' - y}{y_p - y} = \frac{z' - z}{z_p - z} = u$$

$z' = 0 \rightarrow xy$ plane is view plane

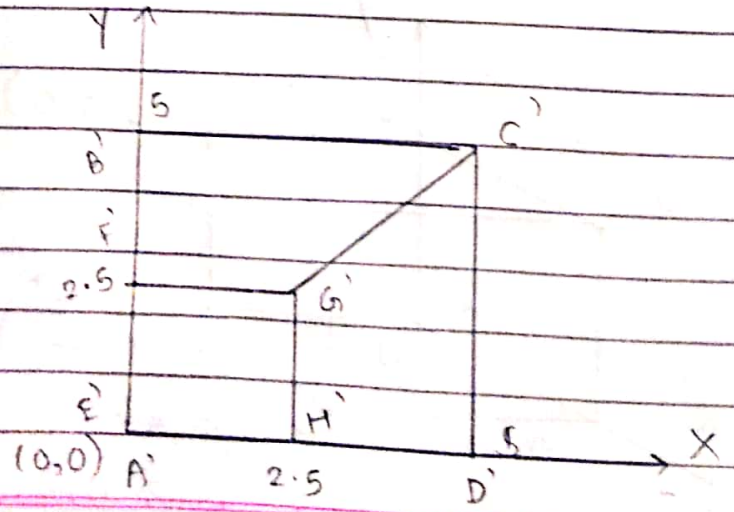
$$u = \frac{-z}{z_p - z} = \frac{-z}{-5 - z} = \frac{z}{5 + z}$$

$$x' = x + u(x_p - x)$$

$$x' = x - ux + ux_p = x - ux$$

$$y' = y - uy$$

points (x, y, z)	$u = \frac{z}{5+z}$	$x' = x - ux$	$y' = y - uy$	$z' = 0$
A (0,0,0)	0	0	0	0
B (0,5,0)	0	0	5	0
C (5,5,0)	0	5	5	0
D (5,0,0)	0	5	0	0
E (0,0,5)	1/2	0	0	0
F (0,5,5)	1/2	0	2.5	0
G (5,5,5)	1/2	2.5	2.5	0
H (5,0,5)	1/2	2.5	0	0



Scanned by CamScanner

- A parallel projection transformation is defined by direction of projection vector Vp and a view plane.

3-D parallel projection

- In parallel projection, object co-ordinate position and the intersection of the plane with projection lines give projected view of an object.
- In parallel projection, object co-ordinate positions are transformed to the view plane along parallel lines and the intersection of a plane with projection line gives projected view of an object.
- A parallel projection transformation is determined by prescribing a direction of projection Vp and view plane.
- In parallel projection, the projection vectors are place ~~in~~ parallel to each other and perpendicular to view plane.
- Parallel projection preserves relative proportions (true length) of objects and this method is used in drafting to produce scale drawings of 3D objects.
- The examples of parallel projections are orthographic projection where we get front elevation and top elevation and side elevation of object.

Perspective projection:-

- In perspective projection object co-ordinate position are transferred on to view plane along lines that converge at common point called as centre of projection.

Scanned by CamScanner

- In perspective projection, the projection vectors are placed in such a way that they are meeting at a common point behind the view plane called centre of projection.
- The intersection of proj. vectors with a view plane gives us the projected view of an obj.
- Perspective proj. produces realistic view but does not preserve the relative proportions i.e. ~~length of and~~ perspective proj. always shows the logical length of an object.

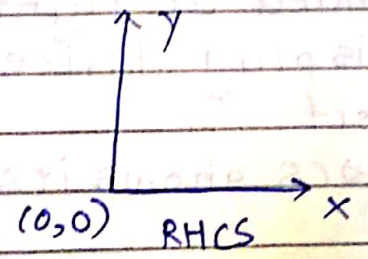
Scanned by CamScanner

2D Viewing and Clipping

Coordinate System
10M M-19 2D viewing transformation
- 2D clipping
- point clipping
- line clipping
M-19 Cohen Sutherland line clipping
Mid point sub division clipping
M-19 LRange Basky line clipping
Polygon clipping
Sutherland Hodgemen poly clipping
Wcilla Asthusron poly clippling
Text clipping

1.) Co-ordinate System

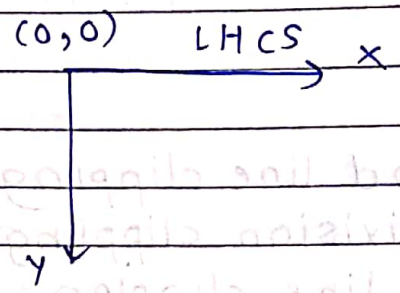
1.) Right Handed Coordinate System (RHCS)
- Any co-ordinate system whose lower left corner is considered as origin is known as RHCS
eg. cartesian co-ordinate system.



- RHCS is infinite in extends
- Any display device which uses concept of RHCS is know as Right Handed Display System
eg. CRO oscilloscope.

Scanned by CamScanner

2.) LHCS (Left Handed co-ordinate system)
- Any co-ordinate system whose upper left corner is considered as origin is known as LHCS.



- LHCS is also infinite in extend.
- Any display device which uses a concept of LHCS is known as left handed display system
eg. CRT monitor, television, notebook.

3.) WCS (World Co-ordinate System)
- WCS is RHCS
- object residing in WCS shows its physical length.

4.) DCS (Device Co-ordinate system)
- DCS can be left handed or right-handed depending upon display device which is used to display object.
- object resides in DCS shows its logical length.

5.) NDCS (Normalized device co-ordinate system)
- NDCS is a DCS where working area is unit square.
eg. Whatsapp profile photo.

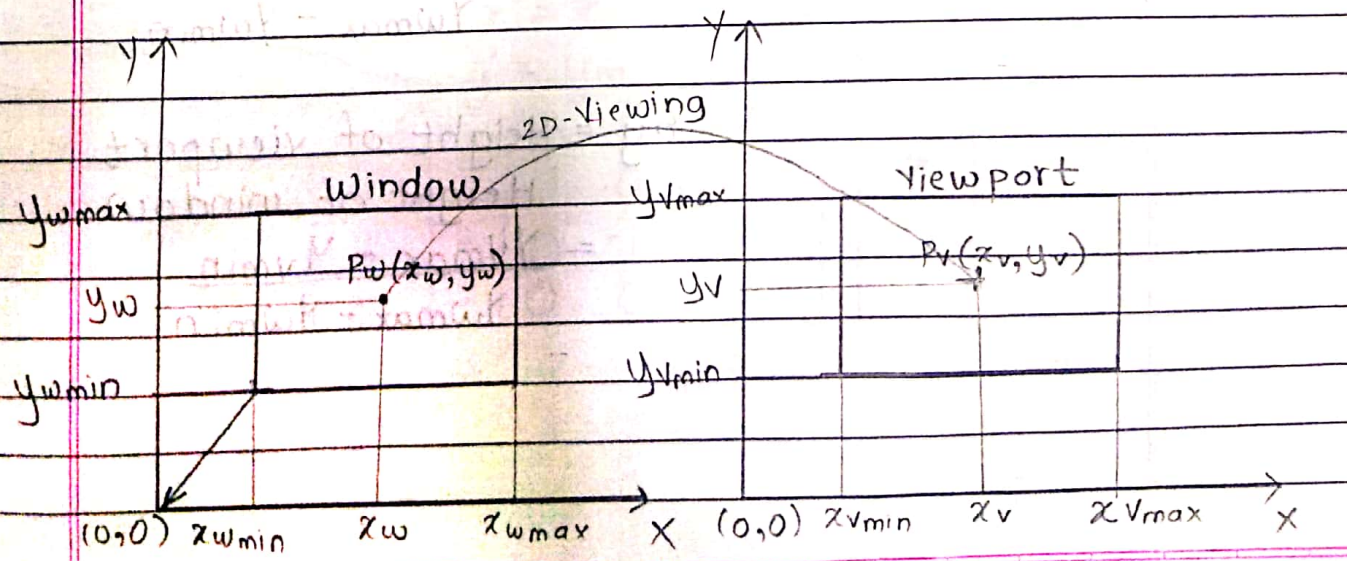
Scanned by CamScanner

6.) Window
- window is WCS
- The dimension of window decides "what is to be displayed" i.e. the proportion of an object selected by window should be visible to the observer.
eg. snipping tool

7.) VIEW PORT
- Viewport is DCS
- Dimension of view port decides where to display the object which is selected by WINDOW.

2D Viewing Transformation -

M-19 Q.) Explain 2D Viewing Transformation OR Explain the transformation to convert window to view port.
A.- A transformation which transforms the WCS to a DCS or simply from window to viewport is known as 2D viewing transformation.



Scanned by CamScanner

Inputs – Window ⇒ (x_{wmin}, y_{wmin}) & (x_{wmax}, y_{wmax})
Viewport ⇒ (X_{Vmin}, Y_{Vmin}) & (X_{Vmax}, Y_{Vmax})
Point on window $P_w(x_w, y_w)$
Output – Point on viewport $P_v(x_v, y_v)$

Step 1:- Translate lower left corner of window to origin. The required translational transformation matrix is –

$$T_1 = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -X_{wmin} \\ 0 & 1 & -Y_{wmin} \\ 0 & 0 & 1 \end{bmatrix}$$

Step 2:- Now Scale the dimension of window so that it fits to the viewport. Required scaling transformation matrix is –

$$S = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$ ~~where Sx is wid~~

where $S_x = \dfrac{\text{width of viewport}}{\text{width of window}}$

$\cancel{S_y =}$ $= \dfrac{X_{Vmax} - X_{Vmin}}{Y_{wmax} - Y_{wmin}}$

$S_y = \dfrac{\text{Height of viewport}}{\text{Height of window}}$

$= \dfrac{Y_{Vmax} - Y_{Vmin}}{Y_{wmax} - Y_{wmin}}$

Scanned by CamScanner

Step 3:- Now re-translate viewport to its original position. Required translational transformation matrix is -

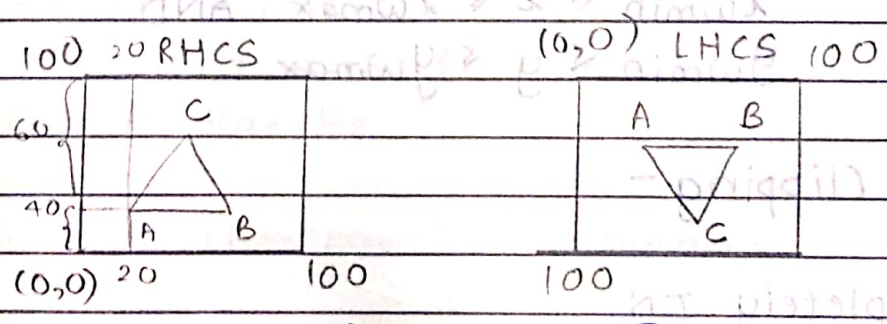
$$T_2 = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & X_{vmin} \\ 0 & 1 & Y_{vmin} \\ 0 & 0 & 1 \end{bmatrix}$$

Step 4:- Composite matrix for 2D viewing transformation.

$$M_T = T_2 \cdot S \cdot T_1$$

$$P_V = M_T \cdot P_W$$

Q.] Transformation of an object from RHCS to LHCS



$$x' = x \quad \text{—— (1)}$$

$$y' = y_{max} - y \quad \text{—— (2)}$$

$$P' = M_{RL} \cdot P$$

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

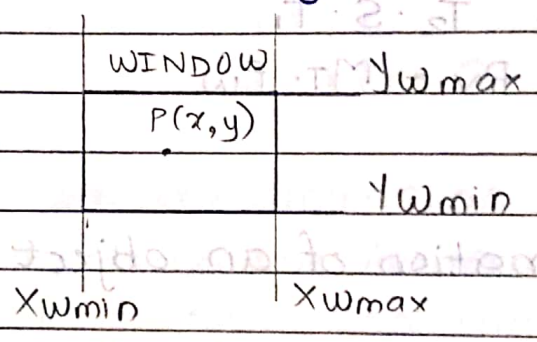
Scanned by CamScanner

2D Clipping:-

Clipping is generally a procedure that identifies which portion of an object are insider or outside of a specific region of a space known as clipping window.

① Point Clipping-

Any point $P(x,y)$ is said to be inside a window if it satifies the following condition



$$x_{wmin} \leq x \leq x_{wmax} \quad \text{AND}$$
$$y_{wmin} \leq y \leq y_{wmax}$$

② Line Clipping-

ⓐ Completely IN

$$X_{wmin} \leq x_1, x_2 \leq X_{wmax}$$
AND
$$Y_{wmin} \leq y_1, y_2 \leq Y_{wmax}$$
Display line AB
Stop.

Scanned by CamScanner

(b.) Completely OUT -

$x_1, x_2 < x_{wmin}$ OR
$x_1, x_2 > x_{wmax}$
OR
$y_1, y_2 < y_{wmin}$ OR
$y_1, y_2 > y_{wmax}$
Discard line AB
Stop.

(c.) Clipping Canditate
Partially IN
Partially OUT
If case (a) and (b) fail then the line is clipping candidate and to clip such line we use following line clipping algorithms.-
(1) cohen Sutherland
(2) Midpoint sub-division
(3) Liange Barske

Cohen Sutherland line clipping -

(1.) Accept end pt. coordinates of line AB i.e. $A(x_1, y_1)$ and $B(x_2, y_2)$ and window coordinates (x_{wmin}, y_{wmin}) (x_{wmax}, y_{wmax})

Scanned by CamScanner

② Assign 4 bit Region code to both end pts. of line AB

$B_4 B_3 B_2 B_1$	$B_4 B_3 B_2 B_1$	$B_4 B_3 B_2 B_1$
1001	1000	1010
y_{wmax}	WINDOW	
0001	0000	0010
y_{wmin}		
0101	0100	0110
	X_{wmin}	X_{wmax}

If $x < X_{wmin}$ then $B_1 = 1$ else 0 (left of left)
If $x > X_{wmax}$ then $B_2 = 1$ else 0 (right of right)
If $y < y_{wmin}$ then $B_3 = 1$ else 0 (bottom of bottom)
If $y > y_{wmax}$ then $B_4 = 1$ else 0 (top of top)

③ Check status of line AB

ⓐ Completely IN

If the region code for both end pts are 0000 then line AB is completely IN

Display line AB
Stop.

ⓑ Completely OUT

If logical AND operation between 2 end pt. codes is NOT 0000 then line AB is completely OUT

Discard line AB
Stop.

Scanned by CamScanner

(c) Clipping candidate-

If case (a) and (b) fails the line AB is clipping candidate.

Go to step (4)

~~5t~~

(4) Determine Intersection boundary.

consider region code of outside pt.

$$P = \underset{T}{B_4}\ \underset{B}{B_3}\ \underset{R}{B_2}\ \underset{L}{B_1}$$

If $B_1 = 1$ line intersect with left boundary

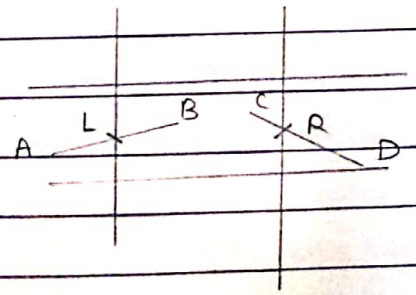
If $B_2 = 1$ line intersect with right boundary

If $B_3 = 1$ line intersect with botton boundary

If $B_4 = 1$ line intersect with top boundary

(5) Determine intersection point

(a) left / Right Boundary



$$x' = x_{wmin}\ (L)$$

OR

$$= x_{wmax}\ (R)$$

$$\frac{x' - x_1}{x_2 - x_1} = \frac{y' - y_1}{y_2 - y_1}$$

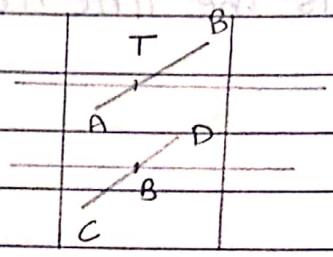
$$y' - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x' - x_1)$$

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

$$I(x', y')$$

Scanned by CamScanner

(b.) Bottom / Up



$y' = y_{wmin}$ (B)

OR

$= y_{wmax}$ (T)

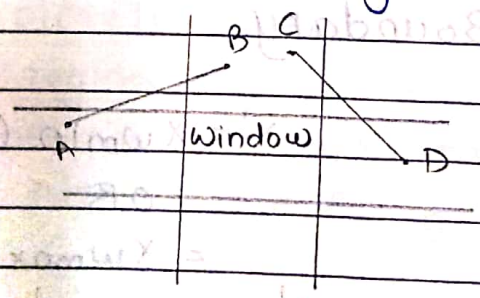
$x' = x_1 + \frac{1}{m}(y' - y_1)$

$I'(x_i', y')$

(6.) To determine region code for I' goto step2

Limitations of Cohen Sutherland line clipping algorithm:-

- Complex mathematical calculations are involved and it is time consuming process.
- Consider the following case -



- In the above senario it is clearly observed that line AB and CD are completely outside the window. But the algorithm requires multiple Iterations to specify that these lines are outside

Drawbacks –

- Algorithm is only applicable w.r.t rectangular window.

Mid-point subdivision line clipping algorithm –

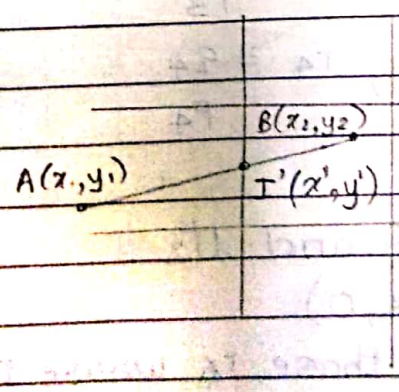
Step ① to Step ③ is same as Cohen Sutherland line clipping algorithm.

Step ④ :- Determine the midpoint of line segment AB using $M = \left(\frac{x_1+x_2}{2}, \frac{y_1+y_2}{2}\right)$

Midpoint M divides the line segment into 2 equal parts.
To find the region code of midpoint go to step ②.

Liang Barsky Algorithm :-

- This algorithm is used to overcome the limitations of Cohen Sutherland line clipping algorithm.
- It works on the principle of parametric equation of line segment.



$$\frac{x'-x_1}{x_2-x_1} = \frac{y'-y_1}{y_2-y_1} = u; \; 0 \le u \le 1$$

$$\therefore x' = x_1 + u\Delta x \Rightarrow I'(x', y')$$
$$y' = y_1 + u\Delta y$$

Scanned by CamScanner

Step 1:- Accept end pt. coordinates of line AB i.e. $A(x_1, y_1)$ and $B(x_2, y_2)$ and window boundary $(x_{wmin}, y_{wmain})(x_{wmax}, y_{wmax})$.

Step 2:- Calculate P_k and q_k for $k = 1, 2, 3, 4$

$P_1 = -\Delta x$ [left Boundary]

$P_2 = \Delta x$ [Right Boundary]

$P_3 = -\Delta y$ [Bottom Boundary]

$P_4 = \Delta y$ [Top Boundary]

$q_1 = x_1 - x_{wmin}$

$q_2 = x_{wmax} - x_1$

$q_3 = y_1 - y_{wmin}$

$q_4 = y_{wmax} - y_1$

Step 3:- If $P_k = 0$ & $q_k < 0$ then the line AB is either horizontal or vertical but AB is completely OUT

Discard line AB

Stop.

Step 4:- Calculate $r_k = \frac{q_k}{P_k}$, $k = 1, 2, 3, 4$

$r_1 = \frac{q_1}{P_1}$, $r_3 = \frac{q_3}{P_3}$

$r_2 = \frac{q_2}{P_2}$, $r_4 = \frac{q_4}{P_4}$

Step 5:- Calculate u_1 and u_2

$u_1 = \max(r_k, 0)$

But consider only those r_k whose P_k is -ve.

Scanned by CamScanner

$u_2 = \min(r_k, 1)$

Consider only those r_k whose $p_k > 0$

Step 6:- If $u_1 \not< u_2$

then line AB is completely OUT.

Discard line AB

stop.

Step 7:- Calculate I' and I''

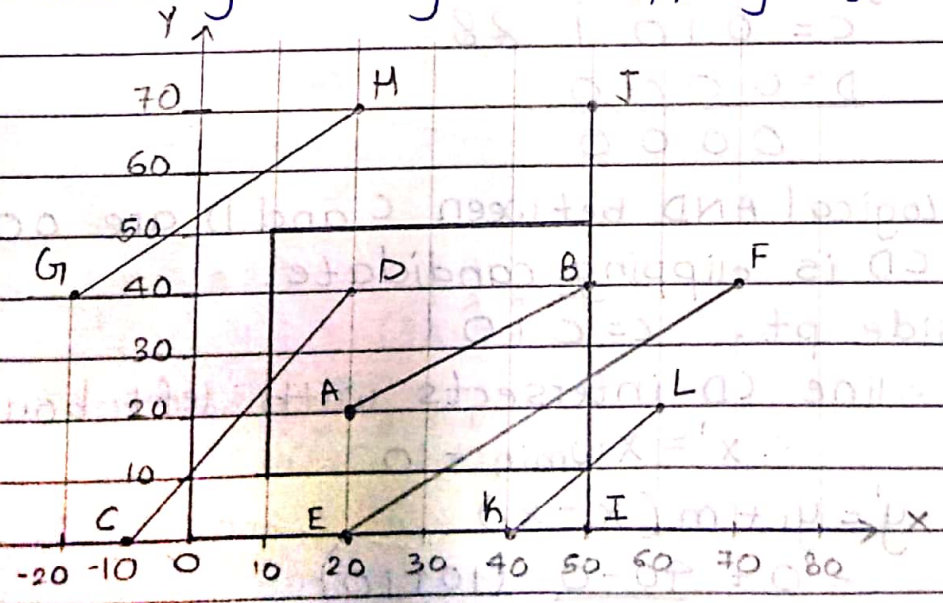
$$I' = \begin{cases} x' = x_1 + u_1 \Delta x \\ y' = y_1 + u_1 \Delta y \end{cases}$$

$$I'' = \begin{cases} x'' = x_1 + u_2 \Delta x \\ y'' = y_1 + u_2 \Delta y \end{cases}$$

Display I', I''

stop.

Clip the following lines against rectangular window boundary by using Cohen Sutherland and Liang Barsky eline clipping algorithm.



Scanned by CamScanner

window – $x_{wmin} = 10$
$y_{wmin} = 10$
$x_{wmax} = 50$
$y_{wmax} = 50$

(1.) Line AB

$A(x_1, y_1) = (20, 20)$
$B(x_2, y_2) = (50, 40)$

	T	B	R	L
	B_4	B_3	B_2	B_1
A =	0	0	0	0
B =	0	0	0	0

∵ Region code for both end pts are 0000
∴ AB is completely IN.
Display line AB
Stop.

(2.) Line CD

$C(x_1, y_1) = (-10, 0)$
$D(x_2, y_2) = (20, 40)$

	T	B	R	L	
C =	0	1	0	1	&&
D =	0	0	0	0	
	0	0	0	0	

∵ Logical AND between C and D are 0000
∴ CD is clipping candidate.

Outside pt. C = 0 1 0 1 (T B R L)
∴ line CD intersects with left boundary.
$\therefore x' = x_{wmin} = 10$

$y' = y_1 + m(x' - x_1)$
$= 0 + \frac{40-0}{20+10}(10+10)$

$y' = \frac{40}{30}(20) = \frac{80}{3} = 26.66$

Scanned by CamScanner

$\therefore I'(x', y') = (10, 26.66)$

New

	T	B	R	L
	B_4	B_3	B_2	B_1
$I' =$	0	0	0	0

$\therefore$ Display line $I'D$

Stop

(3.) Line EF

$E(x_1, y_1) = (20, 0)$

$F(x_2, y_2) = (70, 40)$

	T	B	R	L	
$E =$	0	1	0	0	88
$F =$	0	0	1	0	
	0	0	0	0	

$\therefore$ line EF is clipping candidate

Outside pt. $E = 0100$

$y_1 = y_{wmin} = 10$

$x' = x_1 + \frac{1}{m}(y' - y_1)$

$= 20 + \frac{50}{40}(10)$

$= 20 + \frac{50}{4}$

$x' = 32.5$

$I'(x', y') = (32.5, 10)$

Outside pt.

	T	B	R	L
$F =$	0	0	1	0

$x' = x_{wmax} = 50$

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

$= 0 + \frac{40}{50}(50 - 20) = \frac{40}{50}(30) = \frac{120}{5} = 24$

$y = 24$

Scanned by CamScanner

$I''(50, 24)$

$I' = 0000$

$I'' = 0000$

Display $I'I''$

Stop.

Line GH

$G(x_1, y_1) = (-20, 40)$

$H(x_2, y_2) = (20, 70)$

$$\begin{array}{r} G = 0001 \quad \& \\ H = 1000 \\ \hline 0000 \end{array}$$

Outside pt.

$G = 0001$

$x' = x_{wmin} = 10$

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

$= 40 + \frac{30}{40}(10 + 20)$

$= 40 + \frac{3}{4}(30)$

$y' = 62.5$

$I' = (x_1' y') = (10, 62.5)$

$B_4 \; B_3 \; B_2 \; B_1$

$I' = 1000$

Now $J' = 1000 \; \&$

$$\begin{array}{r} H = 1000 \\ \hline 1000 \end{array}$$

$\Rightarrow$ Outside

Discard line GH

Stop.

Scanned by CamScanner

AIPOPQUIZ

Liang Barsky –

(1) Line AB

$A(x_1, y_1) = (20, 20)$

$B(x_2, y_2) = (50, 40)$

K	P_K	q_K	$r_K = q_K / P_K$
1	$-\Delta x = -30$	$x_1 - x_{wmin} = 10$	$r_1 = -1/3$
2	$\Delta x = 30$	$x_{wmax} - x_1 = 30$	$r_2 = 1$
3	$-\Delta y = -20$	$y_1 - y_{wmin} = 10$	$r_3 = -1/2$
4	$\Delta y = 20$	$y_{wmax} - y_1 = 30$	$r_4 = 3/2$

$$u_1 = \max(r_K, 0)\Big|_{P_K < 0}$$

$$= \max(r_1, r_3, 0)$$

$$= \max\left(\frac{-1}{3}, \frac{-1}{2}, 0\right)$$

$$u_1 = 0$$

$$u_2 = \min(r_K, 1)\Big|_{P_K > 0}$$

$$= \min(r_2, r_4, 1)$$

$$= \min(1, 3/2, 1)$$

$$u_2 = 1$$

$$I' = \begin{cases} x' = x_1 + u_1 \Delta x = x_1 \\ y' = y_1 + u_1 \Delta y = y_1 \end{cases} \quad (20, 20)$$

$$I'' \begin{cases} x'' = x_1 + u_2 \Delta x = x_2 \\ y'' = y_1 + u_2 \Delta y = y_2 \end{cases} \quad (50, 40)$$

Display I' & I''

Stop.

Scanned by CamScanner

② Line CD

$C(x_1, y_1) = (-10, 0)$

$D(x_2, y_2) = (20, 40)$

K	P_K	q_K	$r_K = q_K / P_K$
1	-30	-20	$r_1 = 2/3$
2	30	60	$r_2 = 2$
3	-40	-10	$r_3 = 1/4$
4	40	50	$r_4 = 5/4$

$$u_1 = \max(r_K, 0) \,|\, P_K < 0$$

$$= \max\left(\frac{2}{3}, \frac{1}{4}, 0\right)$$

$$u_1 = \frac{2}{3}$$

$$u_2 = \min(r_K, 1) \,|_{P_K > 0}$$

$$= \min\left(2, \frac{5}{4}, 1\right)$$

$$u_2 = 1$$

$$I' = \begin{cases} x' = x_1 + u_1 \Delta x = -10 + \frac{2}{3}(30) = 10 \\ y' = y_1 + u_1 \Delta y = 0 + \frac{2}{3}(40) = 80/3 = 26.67 \end{cases}$$

$$I'' \begin{cases} x'' = x_1 + u_2 \Delta x = 20 \\ y'' = y_1 + u_2 \Delta y = 40 \end{cases}$$

Display I', I''

Stop.

Scanned by CamScanner

③ Line EF

$E(x_1, y_1) = (20, 0)$

$F(x_2, y_2) = (70, 40)$

k	p_k	q_k	r_k
1	-50	10	$-1/5$
2	50	30	$3/5$
3	-40	-10	$1/4$
4	40	50	$5/4$

$$u_1 = \max(r_k, 0)\big|_{p_k<0}$$

$$= \max\left(\frac{-1}{5}, \frac{1}{4}, 0\right)$$

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

$$u_2 = \min(r_k, 1)\big|_{p_k>0}$$

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

$$= \frac{3}{5}$$

But $u_1 < u_2$ as $0.25 < 0.6$

$$I' = \begin{cases} x' = x_1 + u_1 \Delta x = 20 + \frac{1}{4}(50) = 32.5 \\ y' = y_1 + u_1 \Delta y = 0 + \frac{1}{4}(40) = 10 \end{cases}$$

$$I'' = \begin{cases} x'' = x_1 + u_2 \Delta x = 20 + \frac{3}{5}(50) = 50 \\ y'' = y_1 + u_2 \Delta y = 0 + \frac{3}{5}(40) = 24 \end{cases}$$

Scanned by CamScanner

(a) Line GH

$G(-20, 40) = (x_1, y_1)$

$H(20, 70) = (x_2, y_2)$

K	p_K	q_K	r_K
1	-40	-30	3/4
2	40	70	7/4
3	-30	30	-1
4	30	10	1/3

$$u_1 = \max(r_K, 0)\Big|_{p_K<0}$$

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

$$= \frac{3}{4} = 0.75$$

$$u_2 = \min(r_K, 1)\Big|_{p_K>0}$$

$$= \min\left(\frac{7}{4}, \frac{1}{3}, 1\right)$$

$$= \frac{1}{3} = 0.33$$

But $u_1 > u_2$ ($u_1 \nless u_2$)

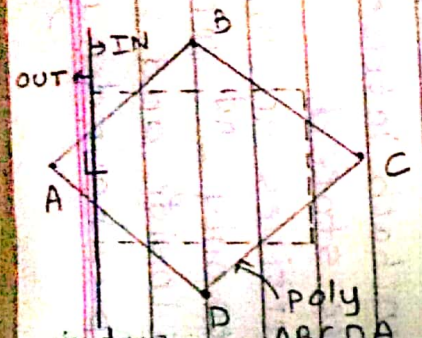
$\therefore$ line is completely out

Discard & stop

Scanned by CamScanner

Polygon clipping algorithm:- (Sutherland Hodgeman)

Initial set of vertices of poly and window boundary → LEFT CLIPPER → RIGHT CLIPPER → BOTTOM CLIPPER → TOP CLIPPER → Clipped Polygon



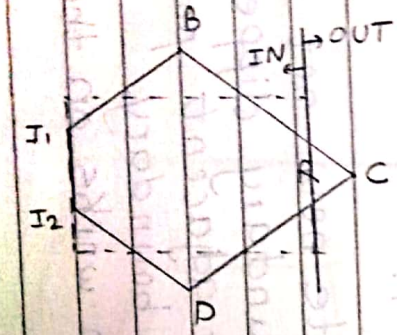
window boundary
(x_{wmin}, y_{wmin})
(x_{wmax}, y_{wmax})

Vertex list

A
B
C
D

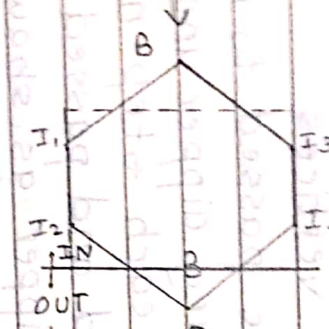
Edge list

AB
BC
CD
DA



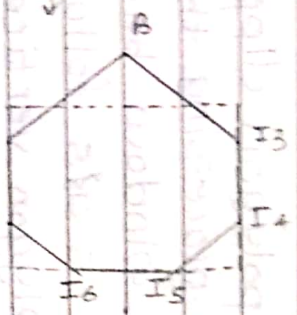
A → B	I_1, B
B → C	C
C → D	D
D → A	I_2

I_1
B
C
D
I_2



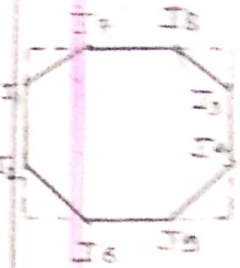
I_1 → B	B
B → C	I_3
C → D	I_4, D
D → I_2	I_2
I_2 → I_1	I_1

B
I_3
I_4
D
I_2
I_1



B → I_3	I_3
I_3 → I_4	I_4
I_4 → D	I_5
D → I_2	I_5, I_2
I_2 → I_1	I_1
I_1 → B	B

I_3
I_4
I_5
I_6
I_2
I_1
B



I_3 → I_4	I_4
I_4 → I_5	I_5
I_5 → I_6	I_6
I_6 → I_2	I_2
I_2 → I_1	I_1
I_1 → B	I_7
B → I_8	I_8, I_3

I_4
I_5
I_6
I_2
I_1
I_3
I_8
I_3

1st vertex	2nd vertex	Save
OUT	IN	Intersection pt 2nd vertex
IN	IN	2nd vertex
IN	OUT	only intersection
OUT	OUT	—

Scanned by CamScanner

Limitations:-

- This algorithm is only applicable for convex polygon.
- If polygon is concave this algorithm fails.

① In Sutherland Hodgeman polygon clipping algorithm
→ A polygon is clipped by processing the entire figure against each window boundary.

② The clipping sequence is as follows -

a.) The initial set of polygon vertices are clipped against the left window boundary, this produces a new sequence of vertices

b.) The set of vertices are passed to right boundary clipper, bottom boundary clipper and a top boundary clipper. At each step new o/p vertices are generated and passed to the next window boundary clipper as show in diagram above.

③ Algorithm to generate new set of o/p vertices at each boundary clipper we need to process each pair of adjacent polygon vertices w.r.t clipper boundary.

Each clipper algorithm works on the following principle -

a.) If the first vertex is outside the window boundary and 2nd vertex is inside the window boundary then both intersection point and second vertex are stored.

b.) If both the i/p vertices are inside the window boundary then only 2nd vertex is stored

c.) If the 1st vertex is inside and 2nd vertex is outside the window boundary then only

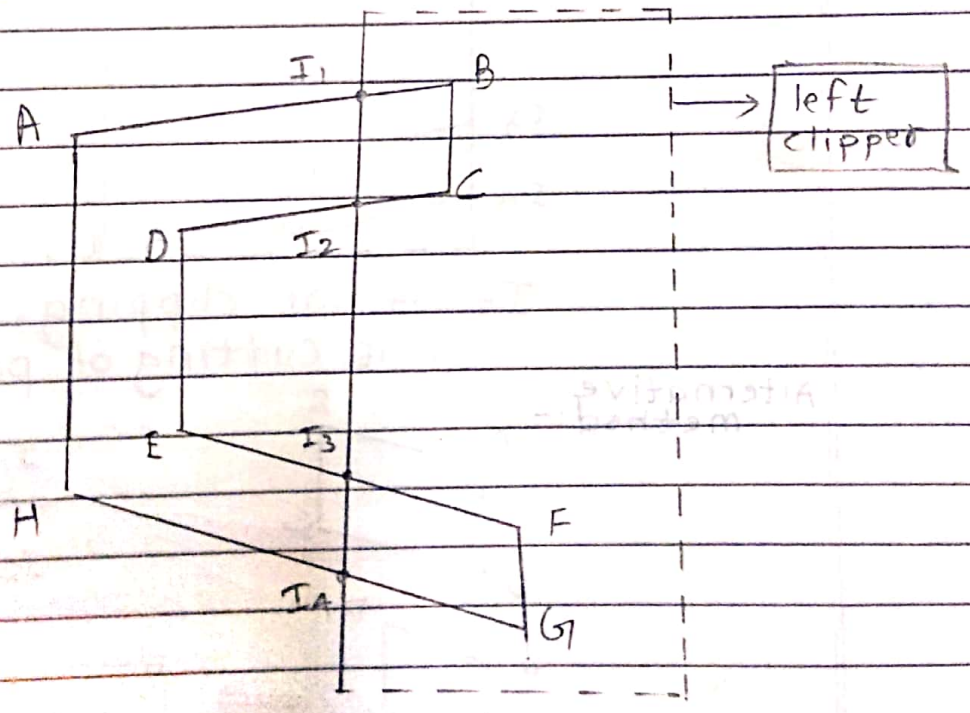
Scanned by CamScanner

edge intersection point is stored.

d.) If both i/p vertices are outside the window boundary then nothing is stored.

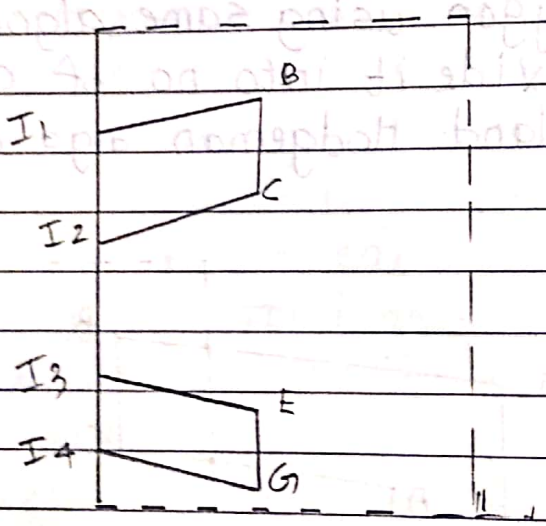
Limitations –

- Convex polygons are correctly clipped by the Sutherland Hodgeman polygon clipping algo.
- But concave polygons cannot be clipped by same algorithm.
- Remedy solution/Alternative method to clip concave polygon using same algorithm is that we need to divide it into no. of convex polygon and apply Sutherland Hodgeman algo on each polygon individually.



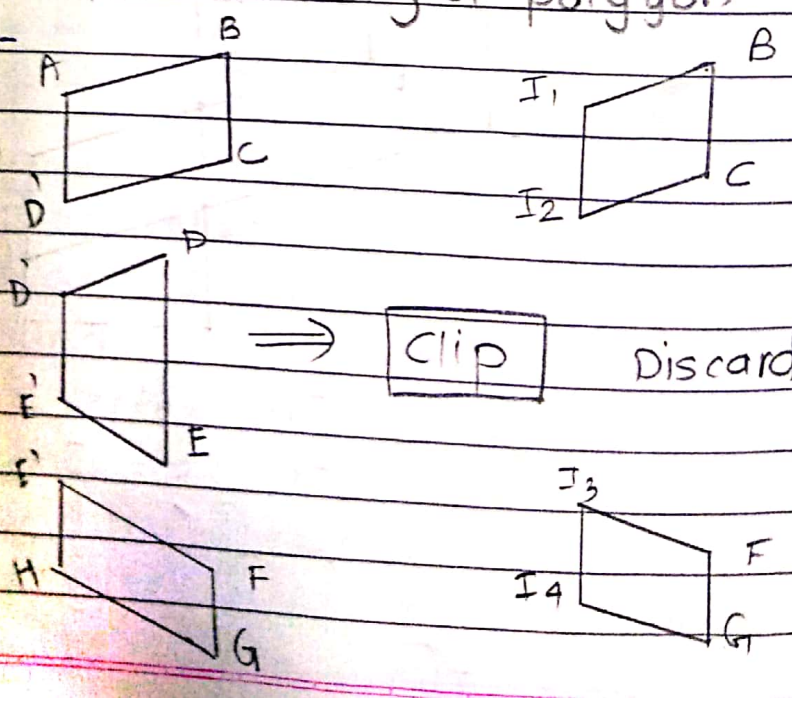
Scanned by CamScanner

Vertex list	1st → 2nd	Save	
A	A → B	I_1, B	
B	B → C	C	I_1
C	C → D	I_2	B
D	D → E	—	C
E	E → F	I_3, F	I_2
F	F → G	G	I_3
G	G → H	I_4	F
H	H → A	—	G
			I_4



It is not clipping,
it is cutting of polygon

Alternative method:-



Scanned by CamScanner

6. Visible surface Detection

1. Curves and fractals –
- Bezier Curve
- Hidden surface elimination algorithm / backface removal.
- Painter's algorithm
- z buffer / depth buffer algorithm
- Warnock algorithm
- Scanline fill algorithm.

Bezier Curve –

1) For any set of control points an approximate curve is formed by adding a sequence of polynomial function from the co-ordinates of control points is called as Bezier Curve.

2) The parametric equation of Bezier curve is given as $p(u) = \sum_{k=0}^{n} P_k \cdot B_{k,n}(u)$, $0 \leq u \leq 1$

where $B_{k,n}(u)$ is called as blending function.

$$B_{k,n}(u) = {}^nC_k \, u^k (1-u)^{n-k}$$

$${}^nC_k = \frac{n!}{k!(n-k)!}$$

n = deg. of Bezier Curve

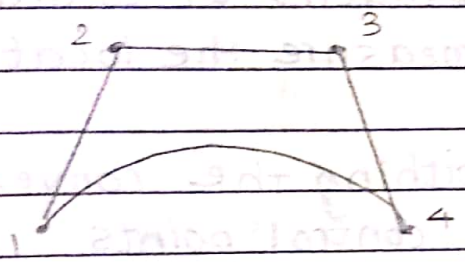
$n+1$ = set of control pts.

3) The degree of bezier curve is 1 less than the no. of control points.
Say n is the no. of control points then the degree of curve is $n-1$ or suppose a curve is having $n+1$ control points then the degree of curve should be n.

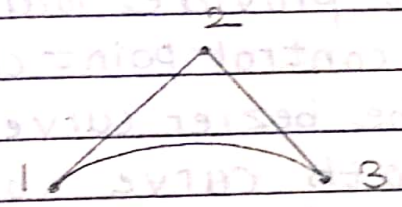
Scanned by CamScanner

If the degree of bezier curve is 3, then it is called as cubic bezier curve.

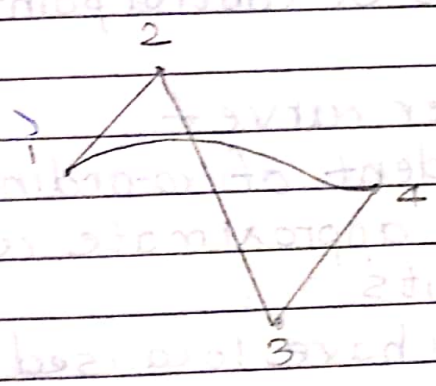
Example –



control pts. = n+1 = 4
degree = n = 3



control pts. = n+1 = 3
degree = n = 2



control pts. = n+1 = 4
degree = n = 3

Applications of Bezier curve –

1.) Bezier curve is used in painting and drawing packages such as CADCAM, engineering and architectural drawing.
2.) Bezier curve is used in game theory to design various games.
3.) It is also used in engineering drawing to draw mathematical shapes such as ellipse, parabola, hyperbola, etc.

Scanned by CamScanner

Properties of Bezier curve :-

a) Bezier curve always passes through the first and last control points i.e. $P(0) = 0$
$P(1) = P_n$

b) Bezier curve is independent of co-ordinate geometry used to measure the location of control points.

c) Bezier curve lies withing the convex polygon boundary formed by control points.

d) Bezier curve does not provide localised control. i.e. moving any control point changes the entire shape of the bezier curve.

e) Bezier curve is a smooth curve to draw against the given set of control points.

Drawbacks of Bezier curve —

a) Since it is independent of co-ordinate geometry, it gives an approximate curve formed by control points.

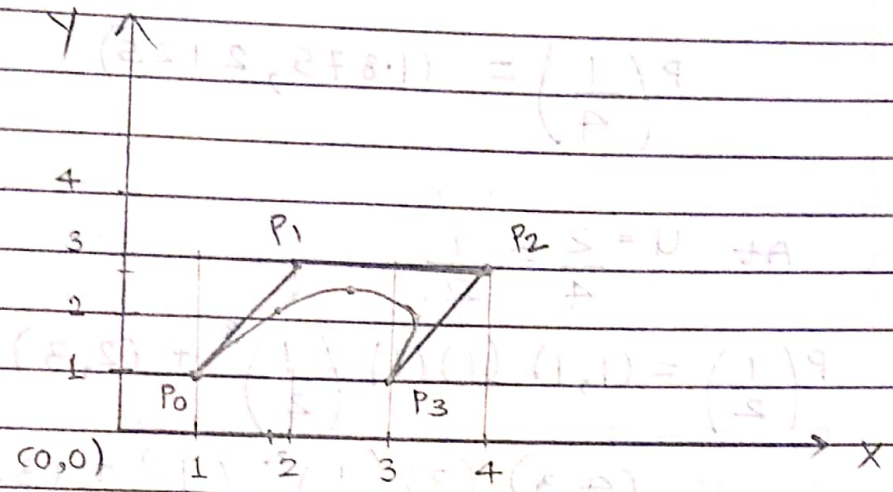
b) Bezier curve does not have localised control.

c) At the joints, complex geometric calculations are required to guarantee the smoothness and continuity.

d) Bezier curve provides less continuity and smoothness.

Q) Give properties and mathematical equation of Bezier curve. Given the vertices of Bezier polygon as (1,1), (2,3), (4,3), (3,1). Determine 5 points on Bezier curve.

Scanned by CamScanner



$$P(u) = \sum_{K=0}^{n} P_K \cdot x_K \cdot u^K (1-u)^{n-K} \; ; \quad 0 \leq u \leq 1 \quad \text{— (1)}$$

Here control pts. $= n+1 = 4$

deg $= n = 3$

we take 5 pts on bezier curve.

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

$P_0(1,1)$, $P_1(2,3)$, $P_2(4,3)$, $P_4(3,1)$

$$P(u) = \sum_{K=0}^{3} P_K \, {}^3C_K \, u^K (1-u)^{3-K}$$

$$P(u) = P_0 \, {}^3C_0 \, u^0 (1-u)^3 + P_1 \, {}^3C_1 \, u^1 (1-u)^2 + P_2 \, {}^3C_2 \, u^2 (1-u)^1 + P_3 \, {}^3C_3 \, u^3 (1-u)^0 \quad \text{— (2)}$$

At $u = 0$

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

$$P(0) = (1,1)$$

At $u = 1/4$

$$P\left(\frac{1}{4}\right) = (1,1)(1)(1)\left(\frac{3}{4}\right)^3 + (2,3)(3)\left(\frac{1}{4}\right)\left(\frac{3}{4}\right)^2 + (4,3)(3)\left(\frac{1}{4}\right)^3\left(\frac{3}{4}\right)^1 + (3,1)(1)\left(\frac{1}{4}\right)^3(1)$$

$$= \left[\frac{27}{64} + \frac{54}{64} + \frac{36}{64} + \frac{3}{64}, \frac{27}{64}, \frac{81}{64} + \frac{27}{64} + \frac{1}{64}\right]$$

Scanned by CamScanner

$$P\left(\frac{1}{4}\right) = (1.875, 2.125)$$

At $u = \frac{2}{4} = \frac{1}{2}$

$$P\left(\frac{1}{2}\right) = (1,1)(1)(1)\left(\frac{1}{2}\right)^3 + (2,3)(3)\left(\frac{1}{2}\right)\left(\frac{1}{2}\right)^2 + (4,3)(3)\left(\frac{1}{2}\right)^2\left(\frac{1}{2}\right) + (3,1)(1)\left(\frac{1}{2}\right)^3(1)$$

$$P\left(\frac{1}{2}\right) = \left[\frac{1}{8} + \frac{6}{8} + \frac{12}{8} + \frac{3}{8}, \frac{1}{8} + \frac{9}{8} + \frac{9}{8} + \frac{1}{8}\right]$$

$$P\left(\frac{1}{2}\right) = (2.75, 2.5)$$

At $u = \frac{3}{4}$

$$P\left(\frac{3}{4}\right) = (1,1)(1)(1)\left(\frac{1}{4}\right)^3 + (2,3)(3)\left(\frac{3}{4}\right)\left(\frac{1}{4}\right)^2 + (4,3)(3)\left(\frac{3}{4}\right)^2\left(\frac{1}{4}\right) + (3,1)(1)\left(\frac{3}{4}\right)^3(1)$$

$$= \left[\frac{1}{64} + \frac{18}{64} + \frac{108}{64} + \frac{81}{64}, \frac{1}{64} + \frac{27}{64} + \frac{81}{64} + \frac{27}{64}\right]$$

$$P\left(\frac{3}{4}\right) = (3.25, 2.125)$$

At $u = 1$

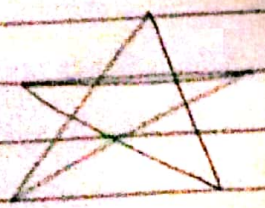
$$P(1) = (0) + (0) + (0) + (3,1)(1)(1)^3(1)$$

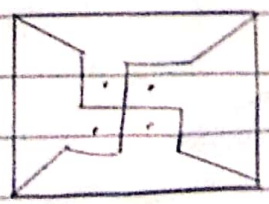
$$P(1) = (3,1)$$

Scanned by CamScanner

Q) Write a short note on fractals-

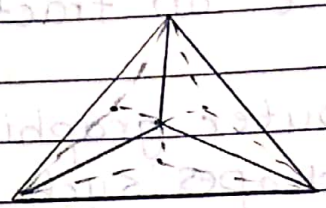
- Till now in computer graphics we have studied basic shapes such as lines, circles, ellipse, rectangle, arc, etc., which can be defined by using their mathematical equations.
- But consider the example of drawing of nature scene where we require some natural objects such as mountains, trees, ocean waves clouds etc. which are irregular in shape.
- It is very difficult to draw these shapes by using normal mathematical equations.
- There are many methods of modelling these natural objects but one of the most effective and interesting method with mathematical perspective is fractal.
- The fractal is nothing but the process of transformation of regular shape of object into irregular shape.
- The basic property of fractal is that they are similar in shape (self similarity).
- The self similarity property of an object can take different forms depending on the choice of fractal representation.
- Consider the example of fractal of line





Scanned by CamScanner

Triangle-



Applications:-

- Fractals can be found in almost every part of the universe starting from bacteria to every real as well as non real part of the universe.
- Fractals have more and more applications in science. The main reason is that they describe very often the real world than the traditional mathematics and physics.
- Fractals find their good applications in astronomy, computer science, fluid mechanics, telecommunication, surface physics & medicine.
- Fractals are also used to draw coast line or to draw a lightning in animation and movies.
- To draw any object we use 3-D effect in application so a concept of fractal surface is used to display 3-D object in a 2-D system.

Scanned by CamScanner

Hidden Surface elimination algo:-

① Painter algorithm

- The basic idea of painter algorithm is developed by Newan at Sancha is to paint the polygon into the frame buffer in order of decrease in distance from the view port.
- This process involves following basic functions -

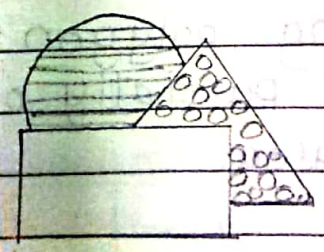
ⓐ Sorting the polygons in order of decreasing depth.

ⓑ Resolving any ambiguity, this may be caused when the polygons z extend overlaps i.e. splitting the polygons if necessary.

ⓒ Scanned conversion of polygons in order starting from the polygon of the greatest depth.

ⓓ The algorithm gets its name from the manner in which an oil painting is created. The artist begins with the background, he then adds the most distant object and then the nearer object and so on.

ⓔ There is no need to erase the portion of background, the artist simply paints over them. The new paint covers the old one so that only newest layer of the paint will be visible for the observer.



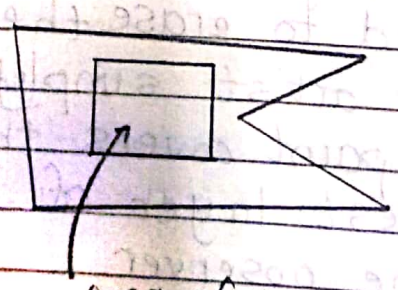
Scanned by CamScanner

In the fig. depth of circle is maximum (Minimum z value) and depth of rectangle is minimum (max z value) then the painter will first paint the circle, then triangle and finally rectangle.

Warnock algorithm -
(Area sub-division algo)

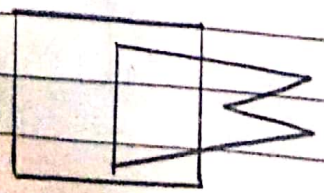
- It is an intersecting approach to the hidden surface problem. It was developed by Warnock.
- He developed the area sub division algorithm which subdivides each area into 4 parts, at each stage in the recursive sub-division process.
- The relation between projection of each polygon and area of interest is checked for following 4 possible relationships -

1. Surrounding polygon - One that completely encloses the area of interest



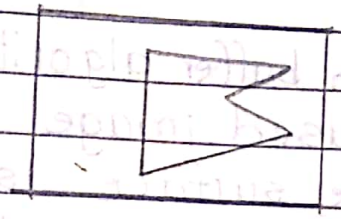
Area of interest / View port

2. Overlapping polygon - One that is partially inside and partially outside the given specified area.

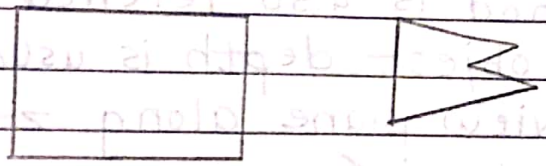


Scanned by CamScanner

③ Inside / Contained polygon - One that is completely inside the area.



④ Outside / Disjoint polygon - One that is completely outside the area.



- After checking 4 relationships we can handle each relationship as follows -

ⓐ If all the polygons are disjoint from the area then the background colour is displayed in the area.

ⓑ If there is only one intersecting or only one contained polygon then area is filled first with the background colour then the part of the polygon contained in the area is filled with the colour of polygon.

ⓒ If a single surrounding polygon but no intersectiong polygons then the area is filled with the surrounding polygon colour.

ⓓ If there are more than one polygons intersecting, contained in or surrounding the area then we need to determine the depth of a polygon and proceed the filling of a polygon either using z buffer or

Scanned by CamScanner

Scan line filling algorithm.

Z buffer / Depth buffer algorithm -

- A commonly used image space approach to detect visible surface is the depth buffer method, which compares the surface depth of each pixel position on the projection plane.
- This method is also referred as z buffer since the object depth is usually measured from the view plane along z-axis of the viewing system.
- Each surface of scene is processed separately, one point at a time across the surface.
- This method is usually applied to the scenes containing only one polygon surface because the depth value can be computed very quickly and the method is easy to implement.
- As the name implies this method requires 2 buffers, a depth buffer is used to store the depth value for each (x, y) position as the surfaces are processed and refreshed buffer stores the intensity value at ~~po~~ each position.
- Initially all positions in the depth buffer are set to 0 (because initially no polygon is considered and it represents only the background) and the refreshe buffer is initialized to background intensity.

Scanned by CamScanner

- Each surface listed in the polygon table is then processed by one scan line at a time, calculating the depth (z values) at each (x,y) pixel position.
- Each z value is compared to the value of previously stored depth buffer value at each position.
- If the calculated depth value is greater than the stored depth value then new value is stored and surface intensity at that position is determined.

z buffer algorithm -

Step ① :- Initialise the depth buffer / z buffer and refresh buffer / intensity buffer so that all the buffer position at (x,y) are depth(x,y) = 0
refresh(x,y) = I background.

Step ② :- For each position on polygon surface compare depth values to the previously stored values in the depth buffer to determine visibility.
~~case~~ ⓐ Calculate the depth for each (x,y) position of a polygon.
ⓑ If current z is greater than the depth of (x,y) then set depth(x,y) = $z_{current}$ and refresh (x,y) = I polygon colour
~~ⓒ Repeat ⓑ~~

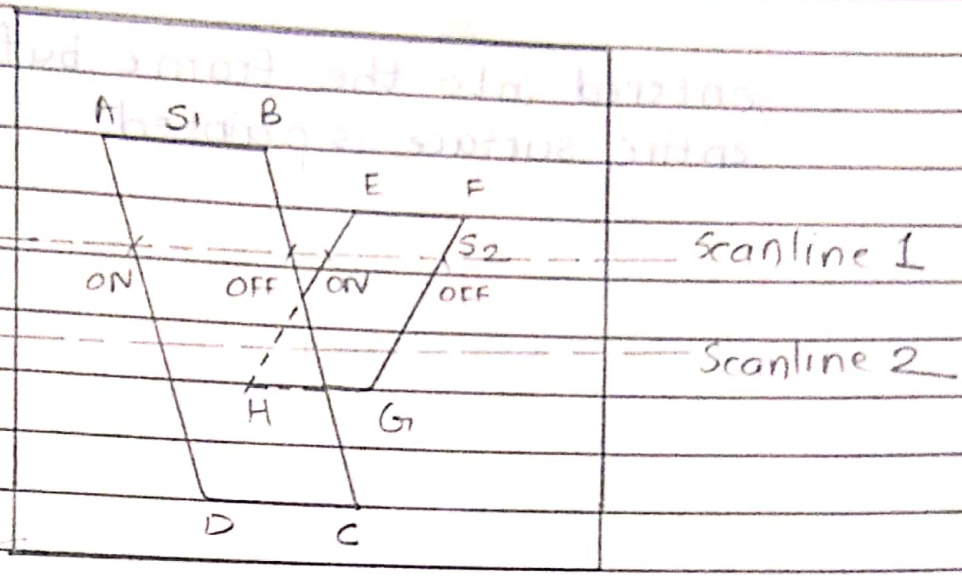
Step ③ :- Repeat step ② for each surface on the view plane.
Step ④ :- Stop.

Scanned by CamScanner

Scan line algorithm-

- Scanline method of hidden surface removal is another approach of image space. It is an extension of scanline algorithm for filling a polygon.
- Here the algorithm operates on more than one surfaces or polygons on a viewplane. Each Scanline is processed, it examines all polygon surfaces intersecting ~~a~~ that scanline to determine which are visible.
- It ~~d~~ then does depth calculations and finds which polygon is nearest to the view plane.
- Finally it enters the intensity value of the nearest polygon at that position into the frame buffer. Scanline algorithm maintains the active edge list which contains active edges for that particular scanline, then sorts them according to increasing order of their x values.
- The scanline method for hidden surface removal also stores a flag for each surface that is set to ON or OFF.
- To indicate whether the position along a scanline is inside or outside to the surface.
- Scanlines are passed from left to right of the surface, the flag is ON and when scanline intersects with the polygon first time it is set to ON and the flag is OFF when scanline intersects with the polygon second time.

Scanned by CamScanner



The figure above explains the scanline method for hidden surface removal

(a) As shown in the figure active edgelist for scanline 1 contains information about edges AD, BC, EH, FG. For the position along this scanline between edge AD and BC the flag of surface S_1 is ON. Hence no depth calculations are necessary and intensity information for surface S_1 is entered into the buffer. Simillarly between edge EH and FG only flag of surface S_2 is ON.

(b) For scanline 2 Active Edge list contains edges AD, EH, BC, FG. For scanline 2 along the edge AD to EH only flag of surface S_1 is ON. However between edge EH and BC the flags of both the surfaces are ON. So in this portion scanline 2 depth calculations are necessary. Here we assume that the depth of S_1 is less than the depth of S_2 and hence the intensity of surface S_1 are loaded into the frame buffer. Hence for edge BC to FG portion of scanline 2 the intensity of surface S_2 are

Scanned by CamScanner

entered into the frame buffer and the entire surface is painted.

Scanned by CamScanner